2019年度　文部科学省委託事業

「専修学校による地域産業中核的人材養成事業」

～Society5.0等対応カリキュラムの開発・実証～

ＡＩ基礎テキスト

学校法人誠和学院　日本工科大学校

【目　次】

AI基礎

第１章　コンピュータ・ネットワーク概論

1. コンピュータと次世代エンジニアリング概要 …… 4

2. 基本入出力インターフェース …… 23

3. ネットワーク概論 …… 40

第２章　ＡＩ基礎

1. ＡＩインターフェイス …… 54

2. ＡＩの概要 …… 70

3. 機械学習 …… 72

4. 大人のＡＩと子供のＡＩ …… 76

5. ディープラーニング（深層学習） …… 78

6. ＡＩ要素技術 …… 80

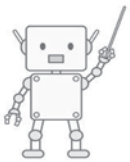
7. ＡＩの情報処理方法 …… 83

参考資料 …… 91

第１章
コンピュータ・ネットワーク概論

1．コンピュータと次世代エンジアリング概要

（1）Society 5.0 とは



ポイント

◇Society 5.0 社会とは
◇社会の進歩
　狩猟社会→農耕社会→工業社会（蒸気→石油）→
　情報社会（コンピュータ）→Society 5.0 社会

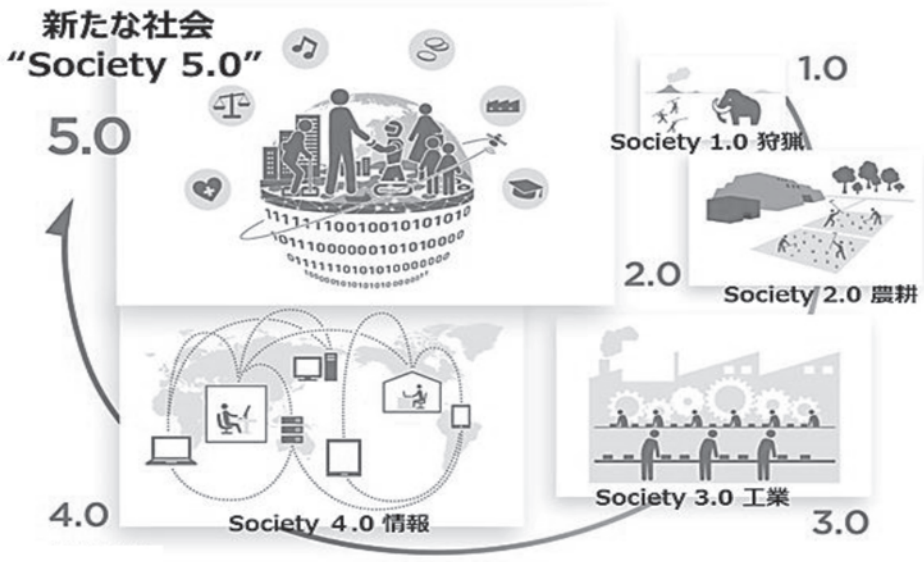
① Society 5.0 とは

「Society 5.0」とは日本政府による科学技術政策の基本指針のひとつで、科学技術基本法に基づき 2016 年から 5 年ごとに改定されている「第 5 期科学技術基本計画」で登場したキャッチフレーズである。

内閣府によると「サイバー空間（仮想空間）とフィジカル空間（現実空間）を高度に融合させたシステムにより、経済発展と社会的課題の解決を両立する、人間中心の社会（Society)」と定義されている。

また、この高度に融合させたシステムにより、経済発展と社会的課題の解決を両立する、人間中心の社会（Society）を目指したもの。これより内閣府の広報にて示された内容をベースに解説する。

狩猟社会（Society 1.0)、農耕社会（Society 2.0)、工業社会（Society 3.0)、情報社会（Society 4.0）に続く新たな社会を指すもので、我が国が目指すべき未来社会の姿として初めて提唱された。



出典：内閣府「Society5.0」

② Society 5.0 で実現する社会

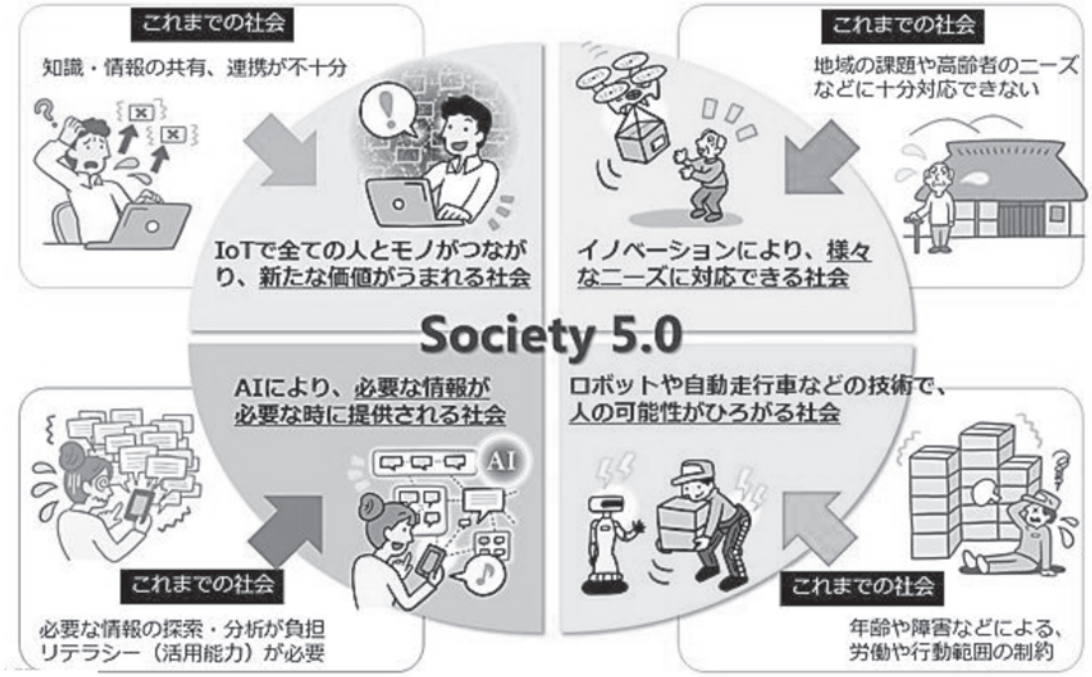
コンピュータの発展に伴い情報社会といわれて久しいが、これまでの情報社会では知識や情報が共有されず、分野横断的な連携が不十分であるという問題があった。

情報社会の実現により社会全体の情報が膨大になり多様性が高まったが、あふれる情報から必要な情報を見つけて分析する作業が負担であったり、年齢や障害などによる労働や行動範囲に制約があった。

Society 5.0 で実現する社会は、IoT（Internet of Things）で全ての「人とモノ」がつながり、様々な知識や情報が共有され、今までにない新たな価値を生み出すことで、これらの課題や困難を克服していく社会である。

また人工知能（AI）により、必要な情報が必要な時に提供されるようになり、ロボットや自動走行車などの技術で、少子高齢化、地方の過疎化、貧富の格差などの課題が克服される。

※注：以後の文章では、人工知能のことを「ＡＩ」と表す場合がある。



出典：内閣府「Society5.0」

【AI 事例：多言語翻訳多言語音声翻訳システムの活用事例（AI 技術を活用した多言語により外国人の駅案内や問い合わせ対応）～ 駅コンシェル ～（京浜急行電鉄株式会社）2018 年度 7 月より本格的に導入】

ア）対話型の逐次翻訳

音声入力した内容を相手の言語に翻訳し、対話型の画面に文章と音声で分かりやすく表示。文章（視覚）と音声（聴覚）による逐次翻訳で円滑なコミュニケーションをサポート。

イ）よく使うフレーズの登録

ご案内でよく使用するフレーズを、ジャンルごとに予め用意するとともに、タブレット端末上で自由に登録・編集することも可能。登録したフレーズを呼び出すことで、翻訳された文章と音声ですばやくご案内。

ウ）電話通訳サービスへのワンタッチ接続

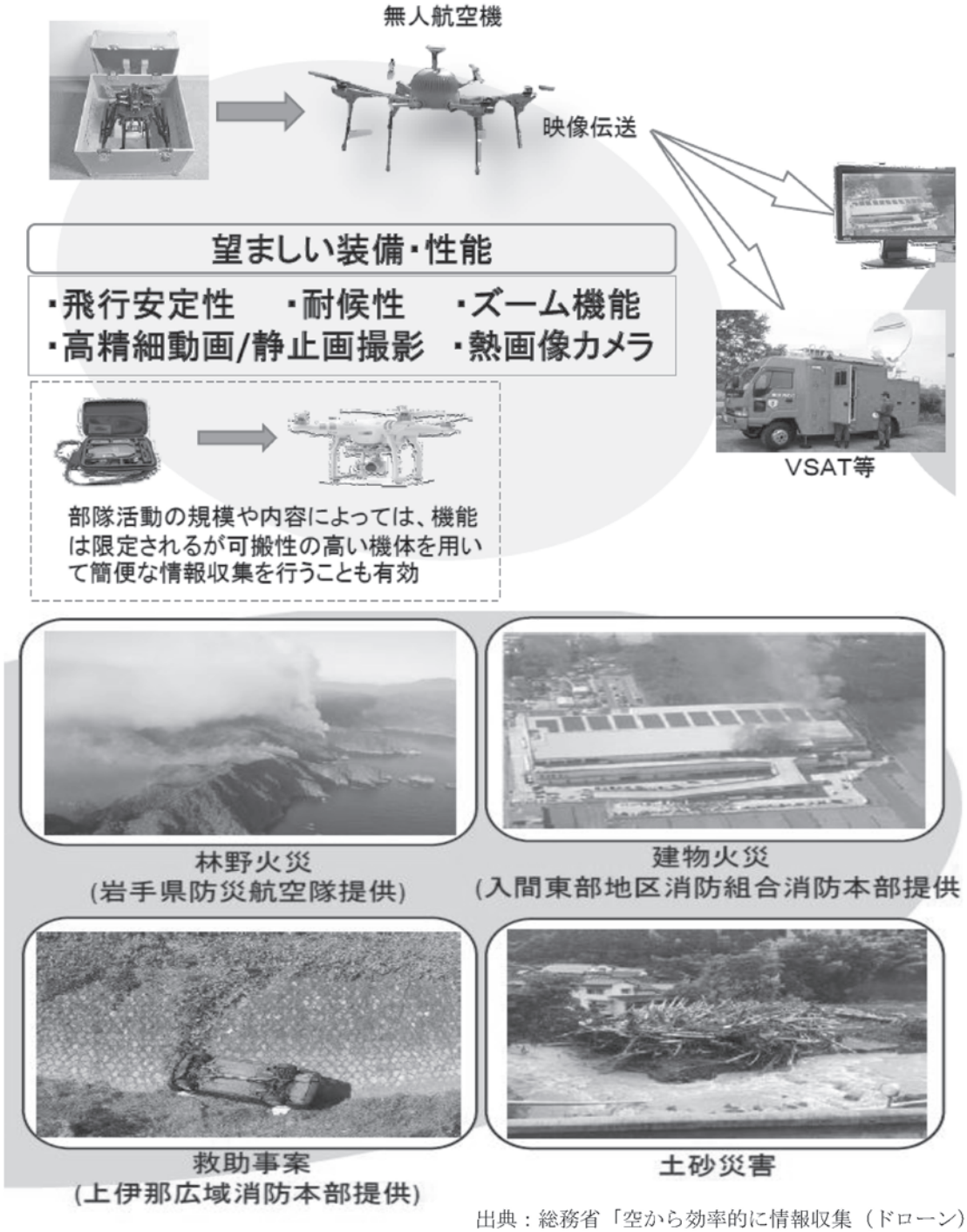
複雑な内容のやりとりが必要な場合には、簡単な操作により、通訳オペレーターを介した電話通訳サービスを利用することが可能。

エ）タッチパネル操作によるご案内

忘れものについて、タブレット端末に表示される多言語のアイコンをタッチすることによる簡単かつ直感的な操作で、いつ・どこで・何を忘れたのかをすばやく確認し、的確に対応することが可能。

【AI 事例：ドローン】

AI 技術の進展により、ドローンが自律的に飛行する開発されている。例えば空から効率的に建設現場において「ドローンによる測量等により作業日数の削減等に貢献」、災害時の「人が近づけない過酷な環境をドローンで情報収集」、山林での「ドローンで森林資源の分布を把握してデータ管理、現地調査の省力化」等の情報収集に活用されている。



出典：総務省「空から効率的に情報収集（ドローン）」

【AI 事例：センサを使った鳥獣被害対策】

センサを活用して獣の追い払いや捕獲の状況を自動で通知し、見回りの効率化を図っている。

罠捕獲センサー

獣検知センサー

〔自動通知イメージ図〕

出典：総務省情報流通行政局「人より早く、詳しく感知（センサ）」

【AI 事例：道路管理業務の削減】

従来は、職員が毎週１回、千葉市内約 3,300 ㎞のうち約 400 ㎞を、約４人の職員で３時間程度パトロール。（別途、毎月１回、夜間に 2人の職員で２時間程度実施）帰庁後は、道路損傷の発見、損傷程度の判定・補修の優先順位付けの作業を、約２時間かけて実施していた。

AI 技術の活用し、従来の市民協働での道路管理に加え、車載カメラで撮影した

画像から道路舗装の損傷を機械学習により、自動抽出する機能を追加した。
その結果、１回あたり 20 時間程度の業務量削減できた。

出典：総務省情報流通行政局「千葉県千葉市、道路管理業務」

③ Society 5.0 の仕組み

これまでの情報社会の仕組みでは、人間がサイバー空間に存在するクラウドサービス（データベース）にインターネットを経由してアクセスして、情報やデータを入手し分析を行ってきた。

◇クラウドサービスとは、従来は利用者が手元のコンピュータで利用していたデータやソフトウェアを、ネットワーク経由で、サービスとして利用者に提供するもの。

Society 5.0 では、フィジカル空間の多様なセンサからの膨大な情報がサイバー空間に集積される。

サイバー空間では、このビッグデータを「人工知能（AI）」が解析し、その解析結果がフィジカル空間の人間に様々な形でフィードバックされる。

◇ビッグデータとは、どの程度のデータ規模かという量的側面だけでなく、どのようなデータから構成されるか、あるいはそのデータがどのように利用されるかという質的側面において、「高解像（事象を構成する個々の要素に分解し、把握・対応することを可能とするデータ）」、「高頻度（リアルタイムデータ等、取得・生成頻度の時間的な解像度が高いデータ）」、「多様性（各種センサ

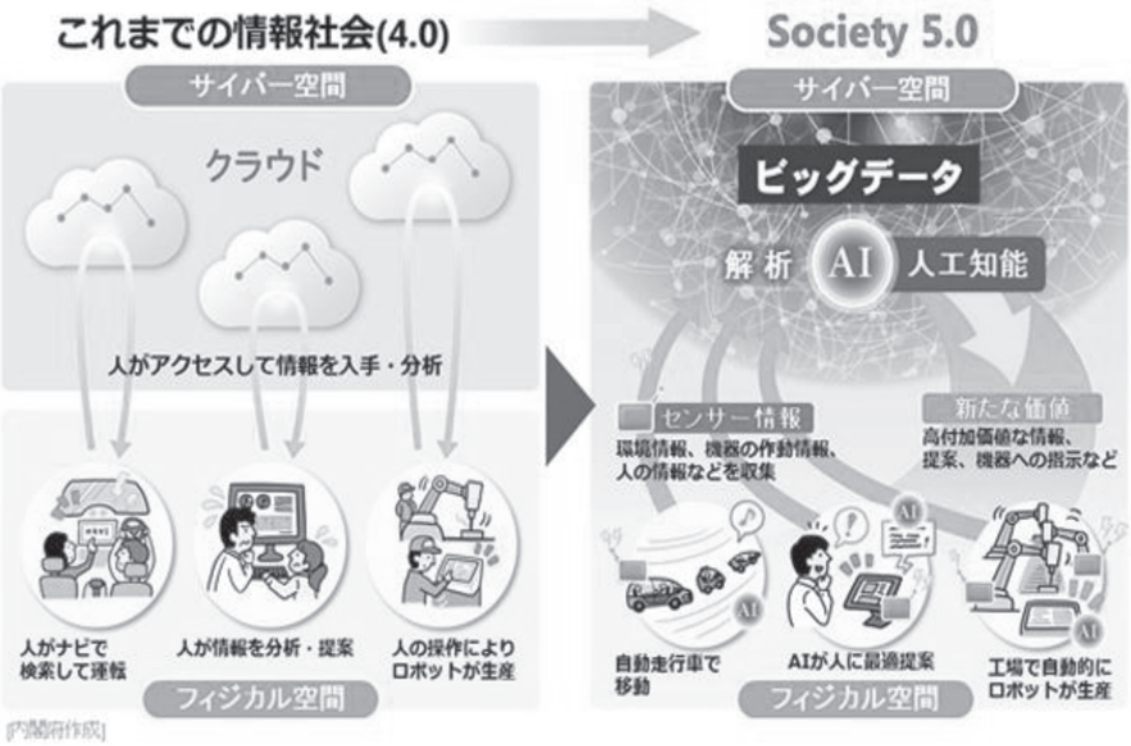
からのデータ等、非構造なものも含む多種多様なデータ)」の３つの特徴を持つ多量性に加え、「多源性（複数のデータソースにも対応可能)」、「高速度（ストリーミング処理が低いレイテンシーで対応可能)」、「多種別（構造化データに加え、非構造化データにも対応可能)」があり、特徴としては、多量性、多種性、リアルタイム性等のデータが生成・収集・蓄積等されることが可能・容易になってきており、異変の察知や近未来の予測等を通じ、利用者個々のニーズに即したサービスの提供や業務運営の効率化、新産業の創出等が可能となること。

〔ビッグデータ各種データ事例〕

出典：平成 24 年度版情報通信白書「情報通信審議会 ICT 基本戦略ボード『ビッグデータの活用に関するアドホックグループ』資料」

今までの情報社会では人間が情報を解析することで価値が生まれてきたが、これからの Society 5.0 の仕組みでは膨大なビッグデータを人間の能力を超えた AI が解析し、支援することになる。その結果、ロボットなどを通して人間にフィードバックされることで、これまでには出来なかった新たな価値が産業や社会にもたらされることになる。

つまり、センサやクラウド環境に蓄積された詳細な情報を人間の能力を超えた処理能力を持ったシステムが高速に解析し、実環境に反映させるというのがこの仕組みである。



出典：内閣府「Society5.0」

④ Society 5.0を支えるAI技術

Society 5.0を支えるAI技術とは、「設備や施設において人間の知覚をはるかに超えて高度化最適されたセンサや計測器から得た環境情報や膨大かつ多様なビッグデータなどから収集したデータを高速に処理し、分類・判定を行い最適な反応を創出する技術を基にした人間への支援システム」であり、かつ継続的に分類・判定や反応・制御の学習を行う仕組み」である。

情報技術があらゆるところに浸透した Society 5.0 に変貌を遂げようとしているが、これらの情報インフラは、巨大なクラウド環境と無数のエッジ端末から成ると考えられている。多くの場合、エッジ端末にはセンサやアクチュエータ（作動機・駆動機あるいは実反応器）が接続され、情報を制御する超小型高性能コンピュータとネットワークインタフェースが備えられている。

クラウドは物理的には分散された多数のデータベースとなるが、論理的にはさまざまなサービスの総体として抽象化されている。

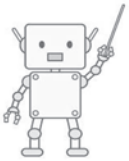
このためエッジ、クラウドのいずれにおいても、大量かつ多様なデータを扱うことになるため従来の情報処理技術の高度化などに加えて、人工知能（深層学習など）、量子計算などがキーテクノロジとなる。現実の諸問題に一定の時間内で反応するリアルタイム技術も高度化させなければならない。このため以下のような技術が「Society 5.0を支える技術」として必要とされる。

ア）情報処理を質的に大転換させる新たなコンピューティング環境の創出

イ） アルゴリズム、アーキテクチャ等を連携・協調させた高く効率化された情報技術

◇エッジとは、端末と端末側のネットワークで収集したデータを回線に送り出すポイント（ネットワークの端末）のこと

（２）第４次産業革命



ポイント

◇Society 5.0 社会は第４次産業革命の進展により実現する社会
◇情報のネットワーク化によって分散していた技術が相互作用的に繋がり、多くの産業で急速に新しい技術の開発と普及

ネットワーク化によって人やモノに留まらず今まで分散していたキー技術がつながり、今後お互いに影響を及ぼし合うことが予想される。

具体的には、ロボティックス（ロボット）、ナノテクノロジ、3D プリンタ、遺伝子工学、バイオ技術など、ネットワークを介すことで相互作用する技術的な進化が新たな産業革命を具現化する。

このような点からも第４次産業革命は、単に ICT 産業に関連するだけではなく様々な既存の産業に及ぶものである。

◇ICT とは、「Information and Communication Technology（情報通信技術）」の略で、通信技術を活用したコミュニケーションを指します。情報処理だけではなく、インターネットのような通信技術を利用した産業やサービスなどの総称のこと。

例えば自動運転技術の革新は、自動運転車の普及と交通事故の減少をもたらし、自動車産業の構造変革や保険の概念をも変革する可能性を秘めている。

また、ドローン（無人航空機）の空撮による三次元計測データは、農林水産業や建設業、鉱業の生産性に飛躍的な向上をもたらす可能性を秘めている。さらに 3D プリンタの普及は製造業における生産管理大学校だけではなく、設計思想や物流政策自体に再考を迫るものである。（参考：総務省「つながる経済」）

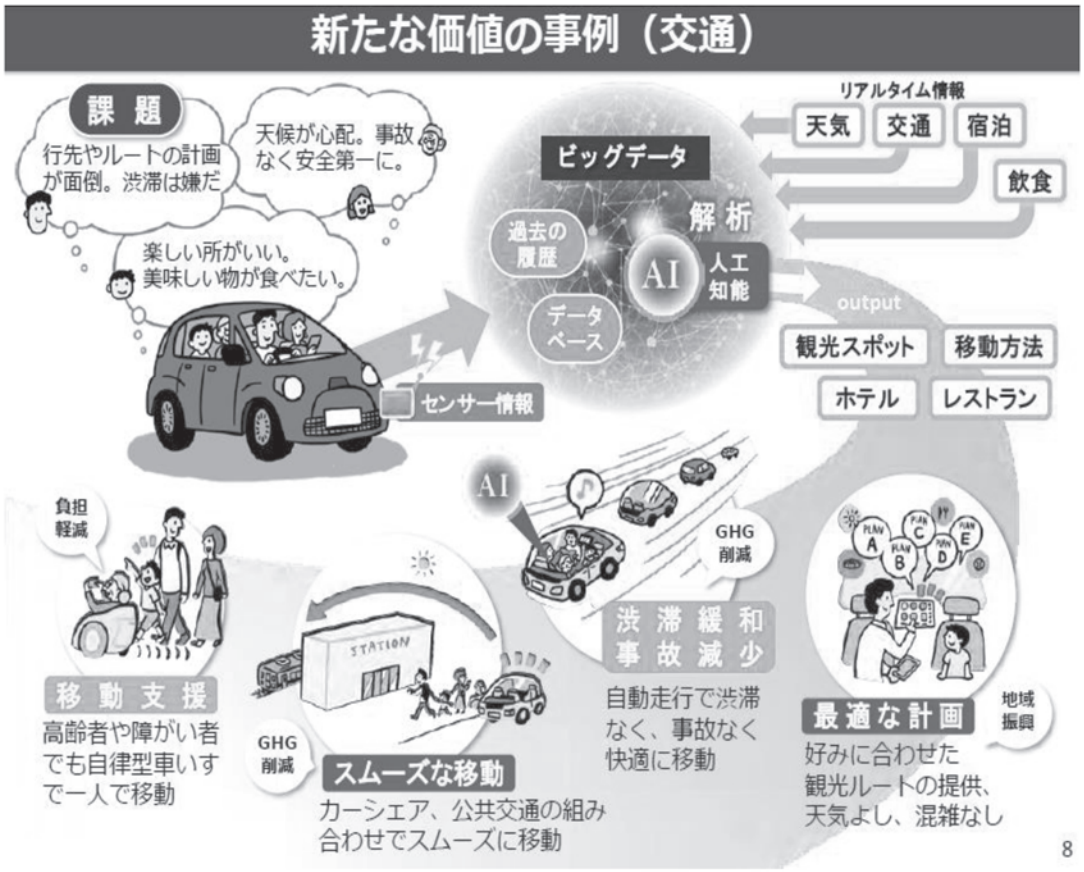
第一次産業革命	第二次産業革命	第三次産業革命	第四次産業革命
18～19世紀初頭 蒸気機関、紡績機など軽工業の機械化	19世紀後半 石油、電力、重化学工業	20世紀後半 インターネットの出現、ICTの急速な普及	21世紀 極端な自動化、コネクティビティによる産業革新※

総務省「平成 29 年度情報通信白書」各産業革命の特徴

① 自動車産業の事例

人と道路と自動車の間で情報の受発信を行い、道路交通が抱える事故や渋滞、環境対策などの様々な課題を解決する ITS※高度道路交通システムの開発が進められており、最先端の情報通信や制御技術を活用して、社会システムを大きく変える取組み。

◇ITS とは、Intelligent Transport Systems の略称で日本語では、高度道路交通システムと呼ばれています。最新の IT 技術を用いて、道路と車両と人間をネットワークで結ぶことで、交通渋滞や事故、輸送効率や快適性など様々な問題を解決していくことを目的とするシステムのこと。

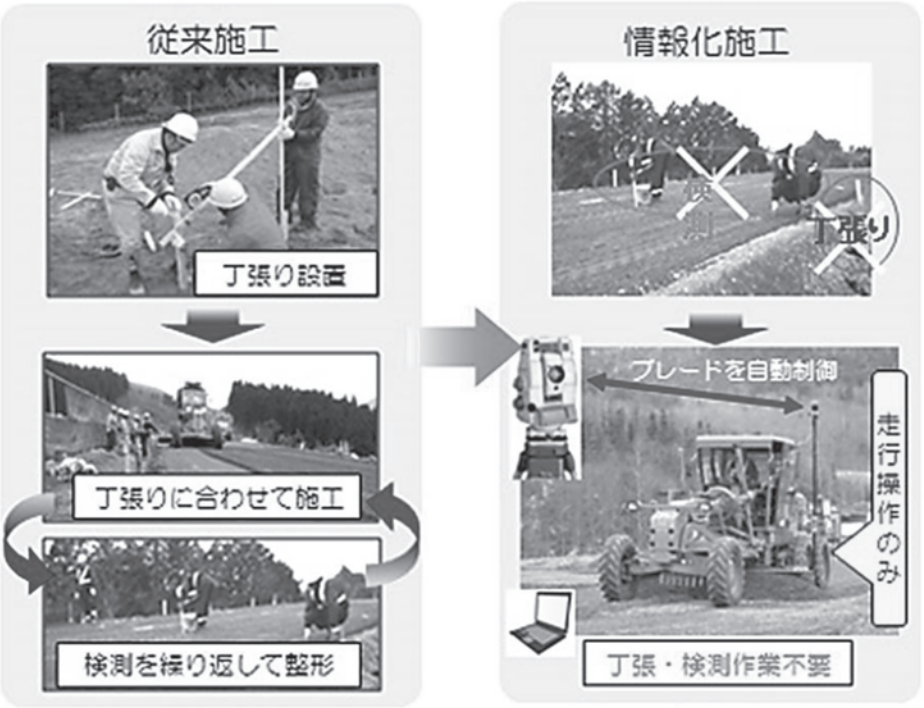


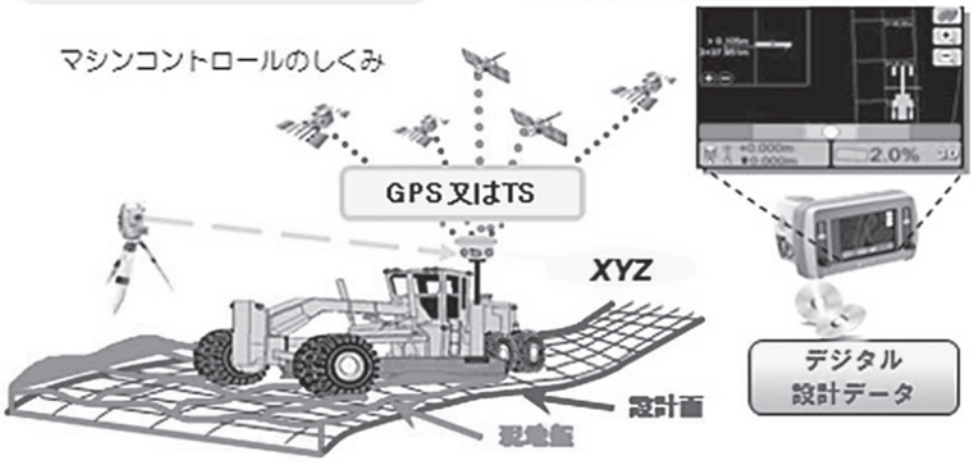
出典：内閣府「Society5.0」

② 建設業の事例

「i-Construction ～建設業の生産性向上～」が進められている。現在の建設業が置かれている状況から建設産業界と取り組まなければならない課題を解決し、生産性の向上を図ろうとする取組で。

◇i-Construction とは、「ICT の全面的な活用（ICT 土工）」等の施策を建設現場に導入することによって、建設生産システム全体の生産性向上を図り、もっと魅力ある建設現場を目指す取組みのことや建設現場に携わる人の賃金の水準の向上を図る建設現場となること、建設現場での死亡事故ゼロにすること、3K「きつい、危険、きたない」から新 3K「給与、休暇、希望」を目指した取組みのこと。





出典：国土交通省

③ AI と雇用の未来

人口減少時代を迎え、AI（人工知能）による生産性の向上に大きな期待が集まっているが、実際にこれまで人間が担ってきた業務を AI に肩代わりさせる事例は枚挙にいとまがなく、その範囲は単純作業から判断業務にまで及びつつある。

しかし AI の進化は人間の「サポート」に留まらず、雇用を奪い、わずかに残った人間の仕事における主導権すら奪ってしまうのではと懸念を示す向きもある。

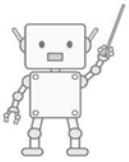
AI によって私たち人間の働き方はどのように変わるのか、また本当に人間の仕事は奪われてしまうのか。

出典：国土交通省「i-Costruction の推進」

ある調査によると、自動化される可能性が 70%を超える職業に就いている労働人口は、全体の 47%もいると示されている。

だが、これまでの産業革命でもわかる通り、新しい技術が生まれると雇用が減少するとその度言われてきたが、新しい技術が生まれる度に新たな職業が発生してきている。AI の検査などは、人が行う必要があるため、ただ単に仕事が減るということではなく、仕事に対する役割が変化していくのである。

（３）コンピュータ概論



ポイント

◇コンピュータの歴史
◇ハードウェアの構成
◇パソコンの構成
◇マルチメディア技術

① コンピュータの歴史

年代	特徴	使用者
第一世代（1945年頃）	コンピュータは、戦争をするために作られました。つまりミサイルの弾道の計算をする目的で作られました。当時のコンピュータは、とても巨大で消費電力が大きく、計算速度があげられます。そのうえ信頼性もあまり高くありませんでした。 そのときの代表的なコンピュータとして「ENIAC」というアメリカ軍のコンピュータがありますが、これには真空管という部品が使われていて、その数はなんと18000本も使われていました。 また、体積８０立方メートル（２５メートルのプール半分くらい）、重量２７トン、消費電力は１４０キロワットも必要でした。	軍隊を中心にした限られた専門家が科学技術計算のために用いました。
第二世代（1965年頃）	この頃のコンピュータの特徴としては、真空管の代わりにトランジスタが使われるようになり、真空管に比べて扱いやすく、小型化・省電力化が進み、信頼性も向上しました。 トランジスタを使った家庭用のラジオが普及したのも、この頃からです。	科学技術者や専門のコンピュータ技師から、一般の技術者にも使用範囲が広がり、大企業や大学にも普及しました。
第三世代（1975年頃）	ＩＣ（集積回路）により、さらに小型化・信頼性の向上・計算の高速化しました。このため、「メカトロニクス」という言葉も生まれました。メカトロニクスとは、機械と電気がドッキングしてできた新しい機械装置をメカトロニクスといいます。 これまでのコンピュータは基本的に人間が機械を扱っていましたが、第三世代からは電子が機械を扱うようになりました。	専門分野で限定されていたものが、一般家庭にも広がりはじめました。

第四世代（1985年頃）	この頃の特徴としては、ＬＳＩ（大規模集積回路）の登場があげられます。LSI は IC の集積率が高くなったものです。また、ワープロや自動車、エアコン、炊飯器、ビデオカメラなど、身近なありとあらゆる家電製品に、ＬＳＩが組み込まれるようになりました。	どの家庭でも、コンピュータを使った製品があるようになりました。
第五世代（1995年以降）	現代は、マルチメディアの時代です。電子によって文字、音、映像を統合的に扱うことができます。ある情報を文字、音声、得絵、写真、映像、ビデオなどのメディアを複数用いて構成するメディアをマルチメディアといいます。 また、パソコンを中心に、インターネットも爆発的に増加してきました。インターネットとは、 世界的な規模で複数のコンピュータネットワークが接続されたひとつの大きなネットワークです。これにより情報が共有され、世界で起きている事が瞬時にわかるようになりました。 これからは、人工知能（ＡＩ）やファジイ、ニューロ理論を使って新しいコンピュータが模索されています。	日常生活をしていく上で、なくてはならないものとなりました。

出典：兵庫教育大学

② ハードウェアの構成とコンピュータの形態

現在実用化されている計算機システムは、1945 年にフォン・ノイマンの考案したプログラム記憶方式であり、計算機の記憶装置内に記憶されたプログラムを逐次読み出し、解析し、実行する過程を繰り返すものである。コンピュータの基本構成は下図に示す。この構成は、計算機の規模にかかわらず本質的には変わらない。算術、論理計算を行う演算装置、プログラムやデータを記憶する記憶装置（メモリ）、外部から情報を受け取る入力装置、外部へ情報を伝える出力装置、及び、これらの装置間の信号の制御を行う制御装置がある。演算装置と制御装置はコンピュータのいわば心臓部で、両者をまとめて中央処理装置（CPU）と呼んでいる。

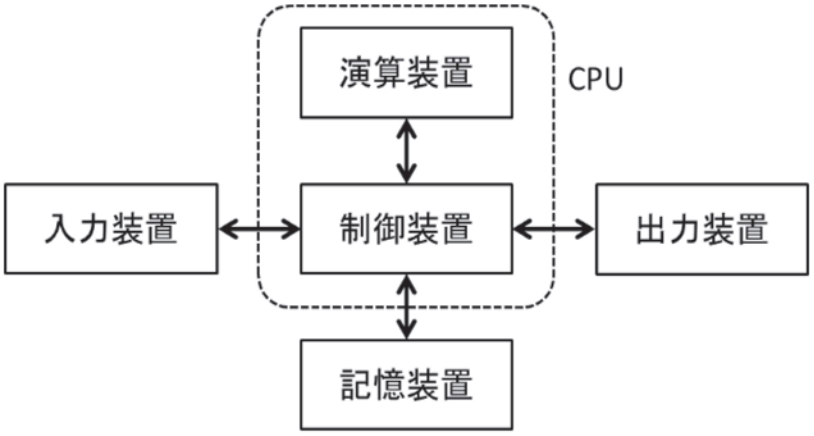


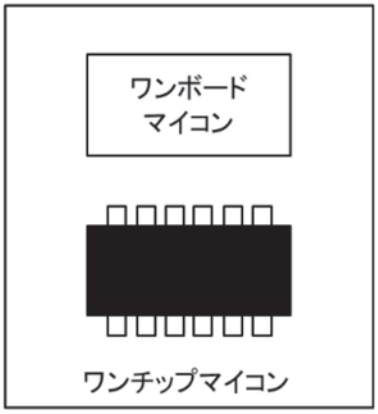
図1.1 コンピュータの基本構造

通常の PC では、本体は演算、制御、記憶装置までを含むケースに収納され、キーボード、マウス等の入力装置、及びディスプレイ、プリンタ等の出力装置を用いる。ハードディスクや DVD などは外部記憶装置の一種である。

入出力が単純で複雑な制御を要しない仕事にはマイクロコンピュータが適しており、多くの製品に実際に組み込まれている。マイクロコンピュータにも様々な形態があり、代表的なマイクロコンピュータでは CPU 単独の LSI や、記憶装置、入出力装置とのインターフェイスまでを持ったワンチップマイコンがある。これらは CPU ボードとして産業機械に組み込まれたり、家電製品の操作性を向上させるために部品として組み込まれたり、用途に応じて使い分けられている。

ワンチップ構成のマイコンとして最も一般的なのは、CPU、メモリ、I/O を 1 個の LSI に統合した 1 チップ構成のものである。あとは電源とクロック、リセット回路、必要に応じてスイッチや表示器、インターフェイスなどを付けるだけで使用できる。バスが外部に出ていないので、パッケージもコンパクトでプリント基板上の配線も削減できる。

一体型



組込型

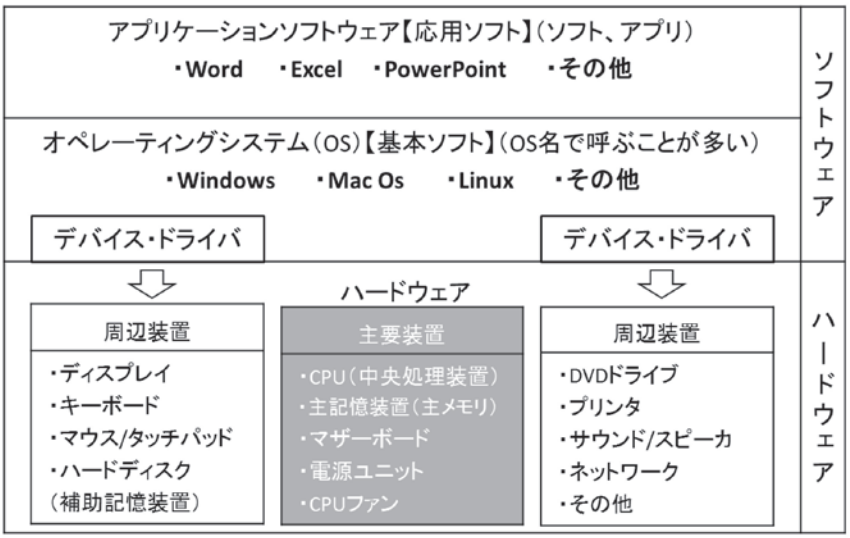
◇インターフェイスとは、コンピュータ本体と各種周辺装置やコンピュータどうしを接続し、電気信号の大きさの調整やデータの形式を変換したりして、両者間のデータのやりとりを仲介する回路や装置のこと。
◇ワンチップマイコンとは、マイコン（マイクロコンピュータ）の一種で、ひとつの IC チップ上に CPU から RAM、ROM、各種入出力装置などを搭載した処理装置のこと。
◇パスとは、その「ファイル」を保存している「フォルダー」に辿り着くまでに階層を重ねたフォルダーの「経路（path)」の意味。

③ ソフトウェアの構成と仕組み

コンピュータを動かす仕組みには前述までの「ハードウェア」に伴って「ソフトウェア」が必要である。ソフトウェアはコンピュータ分野でハードウェア（物理的な機械）と対比される用語で、何らかの処理を行うコンピュータ・プログラムや、更には関連する文書などを指す。

ソフトウェアは一般的にはワープロソフトなど特定の作業や業務を目的としたアプリケーションソフトウェア（応用ソフトウェア、アプリ）と、ハードウェアの管理や基本的な処理をアプリケーションソフトウェアやユーザーに提供するオペレーティングシステム（OS）などのシステムソフトウェアに分類される。

つまりソフトウェアには大まかに 2 種類あって、オペレーティングシステムとアプリケーションソフトウェア（応用ソフト）とになる。またソフトウェアを開発するには「言語」と呼ばれるプログラム言語の環境が必要である。



◇デバイス・ドライバとは、パソコンに接続されている周辺装置（device）をオペレーティングシステム（OS）によって制御するために用意されたプログラムのこと

「パソコンの構成」としては上図に示すように、「ハードウェア」と2種類の「ソフトウェア」の3層構造と理解すればよい。

第1層：「主要装置」および「周辺装置」から成る「ハードウェア」層

第2層：「オペレーティングシステム」という「ソフトウェア」層

第3層：「アプリケーションソフトウェア」という「ソフトウェア」層

オペレーティングシステム（OS）は、パソコン用としてはWindows, MacOS, Linuxなどがあり、モバイル用としてはAndroid, iOSなどがある。その他にも組み込み用のOSなど、様々な種類のOSがある。

④ マルチメディア技術

マルチメディアとは文字情報に加えて音声や映像などさまざまなデータを扱うことであり、これらをデジタルデータとして扱うための技術をマルチメディア技術という。

AI（人工知能）に関するコンピュータ技術では、それまでの文字・数値情報の計算、分類、仕訳、記録などの業務管理・情報処理に使われる手法だけではなく、人間の知覚処理のような音声や映像といったマルチメディアデータの認知・分類・判定などを行うことが多い。

マルチメディアの利用目的は、言うまでもなく「コンテンツ」（情報の意味内　容）を人から人へ伝達することである。情報伝達の手段としてのマルチメディアを考えたとき、各メディアの特質を十分に理解し伝えたい意味内容を効率よく伝えるために、 誰を対象に伝えるのかどのようなメディアを関連させ組み合わせて利用するのかが、重要なポイントとなる。

また、マルチメディア情報を具体的に扱う上で、以下のような技術が必要である。

ア）情報のデジタル化

文字、音声、静止画、動画などの順に情報量はしだいに大きくなってゆく。さらに、音声と動画は時間的な変化を伴うため、長い時間のコンテンツでの情報量は膨大である。これらのデータを劣化なく処理し一元的に管理するためマルチメディア情報を2進数に変換するデジタル化。

イ）情報圧縮技術

マルチメディア情報をデジタルデータとして効率よく蓄積・伝送するため、品質の劣化を抑えた情報圧縮技術。

これらの圧縮技術によって、インターネットの動画コンテンツなどが容易に利用できるようになった。

ウ）コンピュータ処理技術

膨大なマルチメディア情報を、コンピュータを利用して高速に処理・蓄積・伝送するためのソフトウェアおよびハードウェア技術。

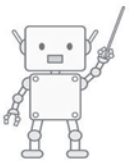
エ）ヒューマンインターフェイス技術

各種のマルチメディアシステムを、あらゆる利用者にとって使いやすく便利にするための利用技術。

なお、音とは、空気の密度の高い部分と低い部分が交互に繰り返して伝播していく波（粗密波あるいは縦波）である。

２．基本入出力インターフェイス

（１）入力インターフェイス



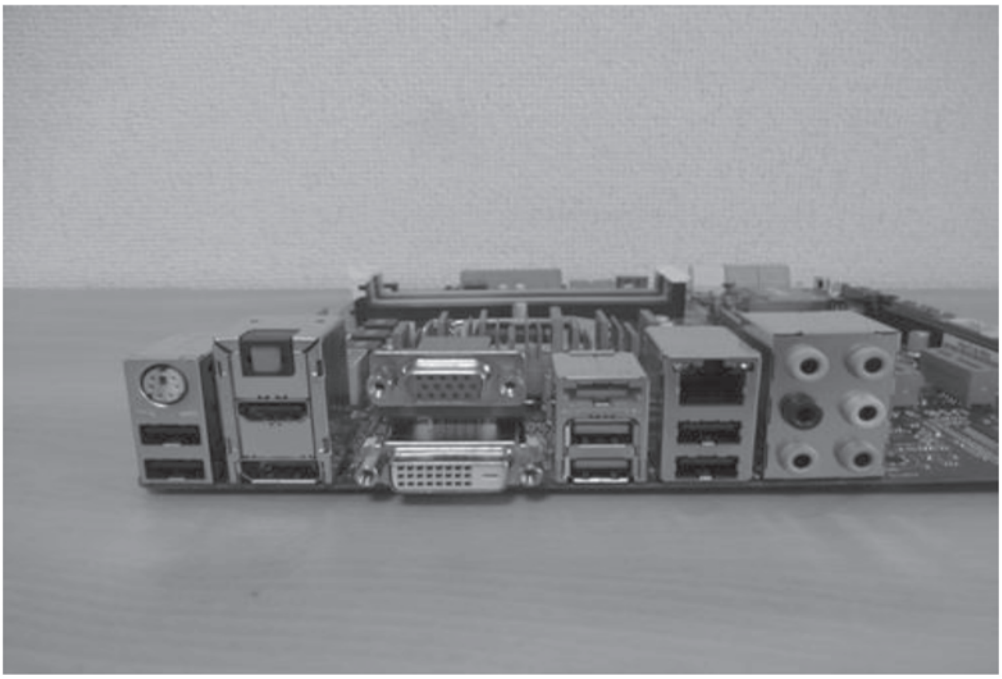
ポイント

◇入力インターフェイスとは、コンピュータシステムに情報を入力するために使用する仕組みの総称で、キーボード、マウス、タッチパネルなどがある。

入力インターフェイスとは、パソコンをはじめとするコンピュータシステムに情報を入力するために使用される仕組みの総称である。入力装置（デバイス）の種類や操作の方式など、システムの構成要素を広く含む。

人間が機器を操作する仕組みとしての入力インターフェイスには、装置としてはキーボード、マウス、タッチパネルなどがある。タッチパネルを利用する入力インターフェイスはタッチインターフェイスと呼ばれるが、マルチタッチインターフェイス、タッチレスインターフェイス、背面入力インターフェイスなど、さらに多種多様な入力方式が研究・開発されている。

入力インターフェイスは出力の方式や方法と共に扱われ、「入出力インターフェイス」と総称されることが多い。特に機器同士の接続方法としての入出力インターフェイスを指す際には「接続インターフェイス」と呼ばれる。



①ＵＳＢ

パソコンや周辺機器によく使われている。

マウス・キーボード、プリンタ、外付けHDD、USBメモリ、外付けTVチューナなど。USB2.0とUSB3.0があり転送速度は異なるが互換性はある。データの送受信だけではなく、バスパワーとして機器の給電・充電用に使われることもある。

写真　https://time-space.kddi.com/ict-keywords/20190116/2549

写真 USBの種類(テックウインド株式会社)

②Micro-USB

USBの小型規格で、スマートフォンやタブレットでデータ通信・充電用として使われている。デジタルカメラでも使われていることがある。

写真 ELECOM極細Micro-USB(A－MicroB)ケーブル

③HDMI

パソコンと液晶ディスプレイを接続する規格。またパソコンとテレビを接続したりすることもある。ひとつのケーブルで映像と音声データを送ることが可能。ハイビジョンテレビや家庭用ゲーム機にもよく使われている。

写真 価格.comマガジン

④DVI

HDMI と同じくパソコンと液晶ディスプレイを接続する規格である。DVI-I、DVI-D など種類がある。

写真 ELECOM 変換ケーブル DVI(29pin)-VGA 1.5m ブラック CAC-DVA15BK

⑤Bluetooth

無線でパソコンと周辺機器を接続する規格。ノートパソコンのほか、iPhone や Android と接続する小型無線ヘッドホンなどに使われている。

⑥SATA

シリアル ATA という。パソコンに内蔵されているハードディスクや光学ドライブをマザーボードと接続する規格である。現在主流の規格である。

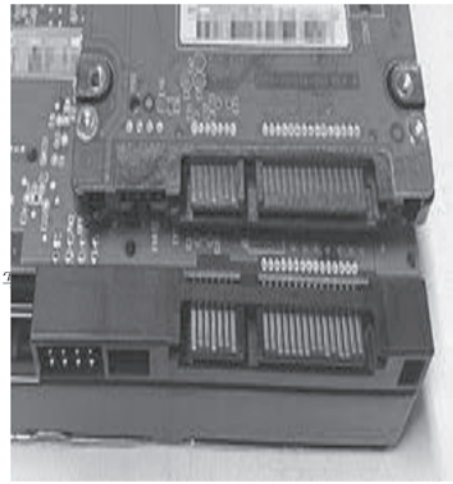


写真 デジモノパーツ.com
SATA 電源拡張ケーブル

⑦PCI-Express

転送速度が高速で、ビデオカード増設などで使われているインターフェイスである。パソコンの内部にあるため拡張スロットともいう。

写真 Advantech PCIE-1674E-AE

写真 StarTech.com PCI Express－PCI 変換

⑧キーボード

パソコンに接続されているキーボードと全く同じ信号を出力するインターフェイスである。

パソコンとキーボードの中間にキーボードインターフェイスタイプのバーコードスキャナを接続することで、読み取ったデータをアプリケーションのカーソル位置に入力できる。

つまりキーボードで入力していたことをバーコードスキャナや磁気リーダによる読み取りで行える簡単に移行できるが、ごく稀に相性が悪く接続ができないケースもある。

写真 ELECOM 有線フルキーボード

⑨マウス

・ボール式：裏面のボールで座標を読み取る。

・光学式：裏面の光の反射光で座標を読み取る。

・トラックボール式：ボール式が反対になったようなもの。このボールを直接指で動かすため本体は固定して使う。

・タッチパッド：ノートパソコンのキーボード手前にあることが多く、指でなぞって操作する。

光学式

写真 ELECOM トラックボールマウス

写真　ヒューレット・パッカード（HP）

⑩タッチスクリーン

指やペンなどをポインティングデバイスとして、ディスプレイなどに触れることで操作することができる。アイコンやボタンといった画面上の操作対象を指差して直接操作（直感的な操作）ができる特徴をもつ。

複数の指で同時に触れて操作できるユーザインターフェイスは、特にマルチタッチインターフェイスと呼ばれる。最近は、タブレットやスマートフォンに実装されており、一般に普及している。

イメージ写真

⑪特殊入力

カメラ

デジタルカメラが登場してから、一般のユーザでも気軽に映像の記録とコンピュータへの取り込みが可能になり、映像系のマルチメディア処理が普及した。デジタルカメラで撮影した画像（写真）は、デジタルカメラとパソコンを接続するか、デジタルカメラのフラッシュメモリ（SD カードやメモリスティック）をパソコンにセットして取り込む。

デジタルカメラとは銀塩フィルムの代わりに撮像素子で撮影した画像をデジタルデータとして記録するカメラであり、世界で初めてコダックが開発した。

一般に「デジタルカメラ」といえば静止画を撮影する「デジタルスチルカメラ」を指し、動画を撮影録画する「デジタルカムコーダ（ビデオカメラ）」は含めない。

現在では静止画撮影が可能なデジタルカムコーダや、動画撮影が可能なデジタルスチルカメラが一般的になっており、双方の性能の向上もあってその境界線が徐々になくなりつつあるが、デジタルカメラはその中でも静止画の撮影に重点を置いたモデルを指す。

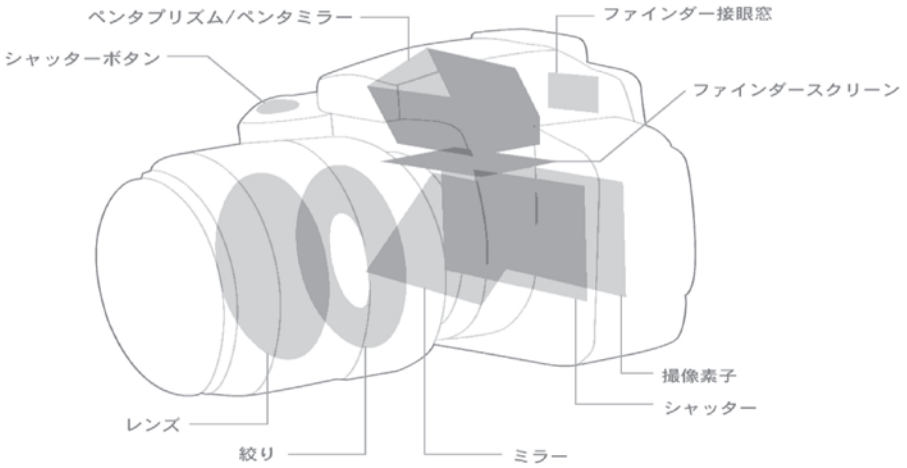
デジタルカメラの全体的な構成は大きく分けて光学系と電子系、そしてそれらを保持する筐体に分類できる。光学系はレンズと絞り機構であり、一眼レフでは光学式ファインダー用のレフレックスミラーとプリズムがこれらに加わる。

その他に、機械式のシャッター機構を備えるものもある。電子系は受光素子とメモリを含む画像演算回路、記録装置、液晶表示器、ストロボ、操作スイッチ、電池などである。

カメラはレンズ、固体撮像素子、デジタル画像処理装置で構成される。使用される固体撮像素子にはCCD（Charge Coupled Device：電荷結合素子）またはCMOS（Complementary Metal Oxide Semiconductor：相補型金属酸化膜半導体）がある。前者は、光情報を電子に変換するフォトダイオードと、その電荷を一時蓄積してバケツリレーのように転送する CCD の配列とを組み合わせた回路で二次元の画像信号を取り出し、後者は、フォトダイオードからの電荷を、CMOS 回路を利用して増幅処理し伝送して二次元の画像信号をつくりだす集積回路で構成される。

いずれもフォトダイオードはシリコンを用いて基板内につくり込まれており、カラー化には、集積されている個々のフォトダイオードにRGB（赤・緑・青）三原色カラーフィルタのいずれかを配置してそれぞれの色を読み取り、これらを組み合わせて画素を構成する。

CCD撮像素子は1970年、アメリカのベル研究所により発表されたもので、高解像・高集積化、高感度、低消費電力、焼き付けがないなどの特徴があり、現在多くの静止画用カメラや動画用カメラに搭載されている。



出典：Nikon ホームページ

⑫センサ

センサは、自然現象や人工物の機械的・電磁気的・熱的・音響的・化学的性質あるいはそれらで示される空間情報・時間情報を、何らかの科学的原理を応用して、人間や機械が扱い易い別媒体の信号に置き換える装置のことをいい、センサを利用した計測・判別を行うことを「センシング」という。検知器とも呼ばれる。

Society 5.0 では、このセンサ類は不可欠なものとして考えられている。また、高速なコンピュータによる情報処理が可能になったことでセンサの意義は増している。

【 センサの種類 】

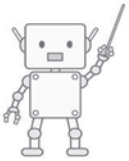
【原理による分類】	
機械料	加速度　力　振動　音波　超音波など
熱	温度
光・放射線	光　赤外線　放射線など
電気	電場　電流　電圧　電力
磁気	磁気センサ
化学	におい　イオン濃度　ガス濃度
【時空間による分類】	
時間：時計	
位置	
距離	超音波距離計　静電容量変位計　光学式測距　電磁波測距
変位	差動トランス　リニアエンコーダ
速度	レーザドップラー振動速度計　レーザドップラー流速計
回転角	ポテンショメータ　回転角センサ
回転数	タコジェネレータ　ロータリエンコーダ
角速度	ジャイロセンサ
一次元画像	リニアイメージセンサ
二次元画像	CCD イメージセンサ　CMOS イメージセンサ
【用途による分類】	
液　漏液センサ（リークセンサ）　液検知センサ（レベルセンサ） 硬度　湿度　流量　傾斜地震センサ	

⑬データグローブ

コンピュータと人間とのインターフェイス用装置のひとつであり、人間の手の単純な動作から情報入力を直感的に行なえるセンシング装置として考案された電気仕掛けの高価な手袋である。指の曲がり具合のような物理データの取得に多様なセンサ技術が用いられ各種形式が存在するが、一般への普及は限られている。限定的ながらコンピュータからの反応出力も行なえるものもある。

出典：日本バイナリー株式会社ホームページ

（２）出力インターフェイス



ポイント

◇標準出力
　コンピュータ上で実施されるプログラムの出力先
◇出力装置
　ディスプレイ・プリンタ・プロジェクタ　など

① 標準出力

標準出力とはコンピュータ上で実行されているプログラムが特に何も指定されていない場合に標準的に利用するデータ出力先であり、コンピュータの出力装置やオペレーションシステム（OS）が提供するデータ出力機能・経路などを指し、多くのシステムではディスプレイ装置による利用者への文字表示が標準出力に設定されている。

コンピュータの出力機器の一つであり、画像を表示する方法には以下のようなものがある。

・ブラウン管(CRT)
・液晶ディスプレイ(LCD)
・プラズマディスプレイ(PDP)
・有機 EL ディスプレイ(OLED)
・マイクロ LED ディスプレイ Micro LED（mLED）
・ビデオプロジェクタ

このうちビデオプロジェクタは、ブラウン管または液晶の表示をレンズで拡大表示するものが多いが、デジタルミラーデバイス(DMD) を使ったものもある。ディスプレイの性能を決定するには、解像度、リフレッシュレート、アスペクト比がある。

ア）解像度

最近の CRT 表示器は非常に柔軟性に富んでおり、おおむね 640×480（通称 VGA）から 1920×1080（Full-HD）で 32 ビットカラーまでの範囲で、様々な表示周波数に対応が可能である。特殊な用途では、さらに情報量の多い 2048×1536（QXGA）や 3940×2160（4K）にも対応できるものもある。

画素数が固定されている液晶ディスプレイの場合は、パネル自体と異なる解像度を表示するために拡大または縮小処理する必要が生じ、文字がぼやけて見づらくなる現象が発生する。液晶パネル自体と同じ解像度を用いるのが望ましい。

◇解像度とは、データの詰め込み具合（画素数の密度）のこと。
◇画素数とは、データの量の多い・少ないということ。写真などで小さな点の集まりの数によって、大きくしたときのきれいさが変わるようなもの。ボケたようなものだと画素数が少ないなど。

イ）リフレッシュレート

CRT モニタで使用できるリフレッシュレートの上限は解像度を上げるほど低くなる。液晶ディスプレイのリフレッシュレートは Windows 等では便宜上 60Hz と表示されることが多いが、原理上は気にする必要はない。

◇リフレッシュレートとは、ディスプレイのチラツキ具合の他、動画やゲームでの滑らかな動きを計るための値のこと。1 秒間にディスプレイがどれくらい書き換わったかを示す。

ウ）アスペクト比

ブラウン管ディスプレイは専ら 4:3 が主流だった。液晶ディスプレイは普及期には 4:3 や 5:4（1280×1024 ドットのパネル）が中心だったものの、2000 年代半ばから 16:10 のワイド画面が特に家庭向けで多くを占めるようになった。

さらに 2008～2009 年にはデジタルハイビジョン放送・薄型テレビと同じアスペクト比である 16:9 の製品が主流になりつつある。現在ではディスプレイも用途によって多様化し、コンピュータの小型化に伴って展示ディスプレイ（デジタルサイネージ）のような機器も使われている。

◇アスペクト比とは、画面や画像の縦と横の長さ（ピクセル数）の比のこと。

② 出力装置

出力装置とはコンピュータ（や実行中のプログラム）からデータを受け取って、人間に認識できる形で外部に物理的に提示する装置。光の像を投影して画面を映し出すディスプレイ（モニタ）やプロジェクタ、紙などに印字・印刷を行うプリンタやプロッタ、音声を発するスピーカやイヤフォンなどがこれに該当する。

主に人間の視覚や聴覚に働きかける原理の機器が多いが、振動で情報を知らせるバイブレータや、ゲームコントローラなどで操作感（押しやすさ、回しやすさなど）を状況に応じて変化させるフォースフィードバック機構など、触覚

を利用する装置もある。映画館や体験型アミューズメント施設などに見られる映像に合わせて霧や風を吹き出す装置なども広義には含まれる。

まだ研究段階ながら、香り（触覚）や味（味覚）を動的に合成してコンピュータからの出力とする装置も構想されている。

これに対し人間や環境、外部の機器から情報を取り込んでデータとしてコンピュータに伝える装置を入力装置（Input Device）といい、キーボードやマウス、タッチパネル、ゲームコントローラ、マイク、イメージスキャナ、各種センサなどが含まれる。出力装置と合わせて入出力装置（I/O device）と総称することもある。イヤホンマイクやプリンタ複合機など、入出力の両方の機能を一体的に提供する装置もある。

◇フォースフィードバックとは、入力装置における機能の一種で、入力操作や画面内のアクションに呼応して入力装置に振動や衝撃を伝える機能のことである。

フォースフィードバックによって、コンピュータ操作の反応を触覚を通じて感じることができ、より現実世界の感覚に近い、バーチャルリアリティを体験することができる。

ア)GPIO（汎用 IO）

GPIO は汎用 I/O ポートとも言われるが、GPIO は I/O のうちデジタル信号に関するピンのことであり、GPIO はユーザ側で制御でき、入力でも出力でも使用できることから汎用の名前がついている。

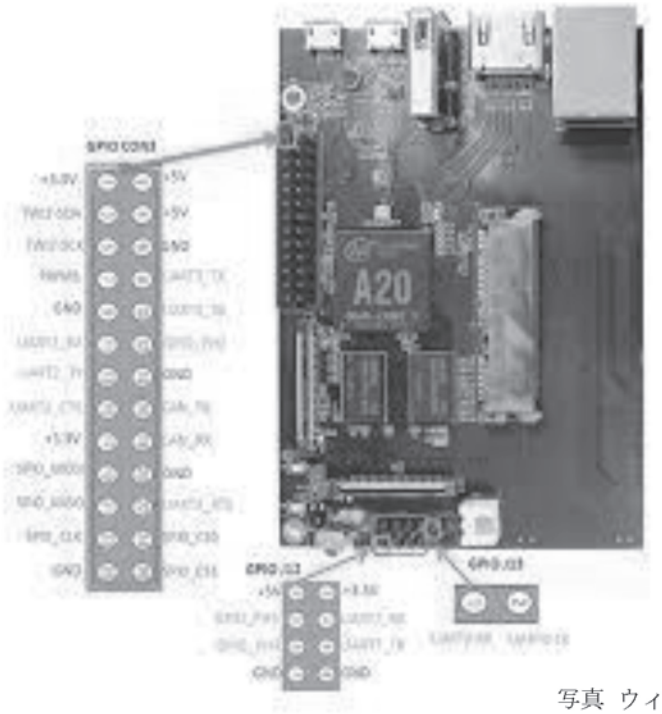


写真 ウィキペディア GPIO のオス側のピン

イ)コンピュータ、マイクロコンピュータに活用されるGPIO

GPIOはピンが少ない集積回路、ビデオカードなどの多機能なチップ、ビデオや液晶ディスプレイなどの組み込み式アプリケーションに活用されている。

つまりテレビや洗濯機、スマートフォン、おもちゃなど、コンピュータ機器に限らず、マイクロコンピュータが使用されているさまざまな機器に使用されているということである。

GPIOを使用することによって、スイッチが入った切られたなどのデジタル信号がマイクロコンピュータに送られ処理が行われる。

出力の場合も同様で、単にLEDなどの出力装置を設置しただけではコンピュータの指示は出力装置にまで届かない、GPIOを活用することによって指示がデジタル信号で届き、処理が実行される。

コンピュータでは、64ピンなどのように複数のGPIOが使用されている。たとえば62ピンが存在するとして、この62ピンはユーザの設定によって自由に入力用、出力用と組み替えることが可能である。ユーザの設定次第では、入力用を多くしたり、出力用を多くしたり、カスタマイズすることができる。

またGPIOは、有効・無効の設定が可能である。たとえばCPUを管理するのにあたって、不要な部分は時と場合に応じて無効にすることができる。必要であればユーザの設定で有効にもできる。

なお、GPIOを使用した入力値は、割り込みが可能である。GPIOコントローラの一部においては両エッジの信号で割り込みをすることができる。こうした割り込み機能は、ボタンイベントなどの機能でよく活用されるものである。

その他、GPIOは出力値で読み書きが可能なほか、入力値においては読み出しが可能となっている。入力、出力と汎用性の高いGPIO。ラズベリーパイ（Raspberry Pi）など、自身でマイクロコンピュータを動かしたい場合の組み込み式プログラミングを作る際にもよく使用される。

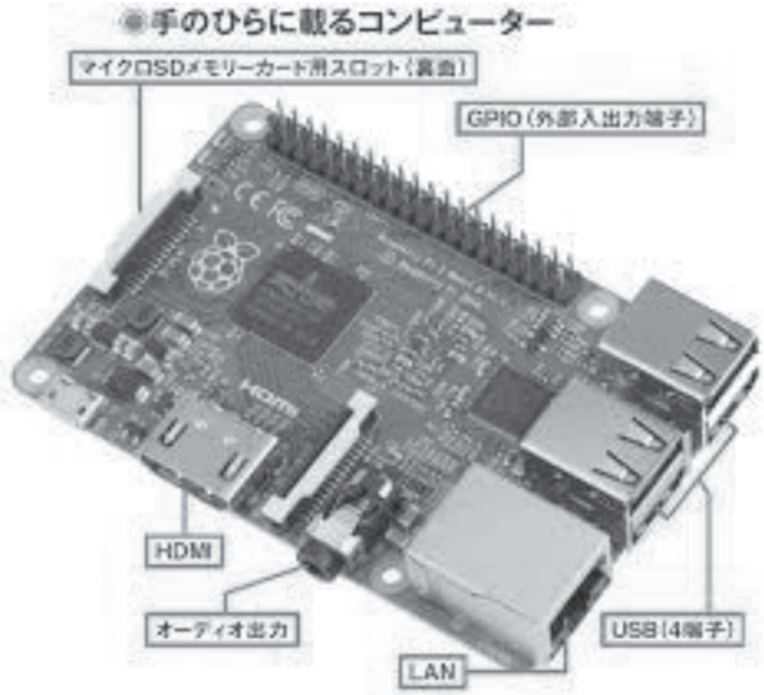
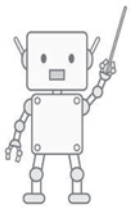


写真　日経 PC21 手のひらに載るパソコン「ラズベリーパイ」

（３）ユーザインターフェイス



ポイント

◇ユーザインターフェイス
利用者が対象を操作するために接する部分
キーボード、ディスプレイなどの操作手順
メニューなどの視覚的要素、警告音などの聴覚的要素
◇ヒューマンユーザインターフェイス
人間が使用するのに必要な安全性、信頼性、利便性などをもった機械システム

① ユーザインターフェイス（UI）

ユーザインターフェイスとは利用者が対象を操作するために接する部分です。パソコンの場合、マウスやキーボード、ディスプレイといった機械的な要素、どのように操作するかという手順、画面に表示されるメニューやアイコン、ウインドウといった視覚的要素、警告音や文字の読み上げといった聴覚的要素などを指す。

機械の性能だけでなく、ユーザインターフェイスの出来不出来も対象の使い勝手に大きく影響する。

Windows のように文字以外の視覚的要素を操作対象として利用するものを、グラフィカル・ユーザ・インターフェイス(GUI)と呼ぶ。

② ヒューマンユーザインターフェイス

人間が使用するのに必要な安全性、信頼性、利便性などをもった機械システムを、人間の知覚、認知、行動を熟知したうえで利用者と機械を含んだ系をシステム的にとらえて構築・研究する分野である。

インターフェイスを人間と外界との相互作用と考えると、機械と人間だけでなく日常の生活空間の設計などあらゆる分野にわたる考えである。なおヒューマンインターフェイスには、ハードウェア技術とソフトウェア技術の２つが存在するが、これらの技術を分離して考えることは不可能である。コンピュータとユーザとの良好なインターフェイスを保つためのソフトウェア技術をユーズウェアと呼ぶ。

ア)グラフィカルユーザインターフェイス（GUI）

入力としてキーボードやマウスといったデバイスを用い、ディスプレイ上にグラフィカルな出力を提示する方式。

マウスを使った入力方式は Windows や MacOS のものが一般的だが、他に

も境界線と交差するマウスポインタの動作で何らかの情報を入力する方式（Crossing Based Interface)、マウスジェスチャで制御する方式などもある。

イ)ウェブユーザインターフェイス（WUI)

ウェブページ生成によって入出力を行い、それをインターネット上で転送し、ウェブブラウザでユーザがそれを表示する。既存のHTMLベースのウェブブラウザを使うことができ、制御はJava・Ajax・Adobe Flash・Microsoft .NETといった比較的新しい技術で実装される。

ウ)キャラクタユーザインターフェイス（CUI)

ユーザがキーボードからコマンドを入力し、ディスプレイ上に文字を表示することで出力とする方式。マウスなどポインティングデバイスを使用しないシステム管理作業などで使われる。

エ)触覚インターフェイス

補助的な出力として触覚フィードバックを用いる方式。コンピュータシミュレーションやバーチャルリアリティで使われる。

オ)タッチインターフェイス

タッチパネルとGUIを入出力に使う方式。工業機械やセルフサービス型機械（ATMなど）またはタブレットなどでよく使われる。

主にAIで利用されるその他のユーザインターフェイスの種類として、以下のものがある。

カ)バッチインターフェイス

バッチ処理で使われる対話型でないユーザインターフェイス。ユーザはバッチジョブとして処理の詳細をまとめて入力し、全ての処理が完了した時点で出力結果を得る。処理が始まると、システムはさらなる入力を求めることはない。

キ)パーセプチュアルユーザインターフェイス（英: perceptual user interface, PUI)

ユーザは従来的なコマンド入力を行わず、身振り手振りや音声を使って意思を伝達し、出力は映像や音声で行われる方式。Kinect やSiri/Cortanaなどが挙げられる。

ク)リフレクシブユーザインターフェイス（英:reflexive user interface）
ユーザインターフェイス全体をユーザが再定義可能な方式。主に非常にリッチなGUIでのみ可能。

ケ)タンジブルユーザインターフェイス（英: tangible user interface, TUI）
物理的な接触を重視したユーザインターフェイス。

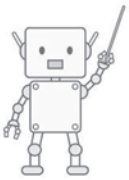
コ)テキストユーザインターフェイス
出力はテキスト形式だが、入力はコマンド入力以外の方式も可能なユーザインターフェイス。テキスト方式のメニュー操作などを指す。
音声ユーザインターフェイス電話において、音声で案内し、ユーザは電話機のプッシュボタンで入力する方式。音声ガイダンス。

サ)ズーミングユーザインターフェイス
GUI の一種で、情報オブジェクト群が異なる詳細さレベルで表示され、ユーザがその中からオブジェクトを選ぶとさらに詳細が表示されるという方式。Microsoft Windows 8 で導入された Modern UI スタイルのアプリケーションでは、「セマンティックズーム」と呼ばれる。AI に関して特化されたヒューマンユーザインターフェイスは、次章で解説する。

３．ネットワーク概論

（１）インターネット原理

ポイント



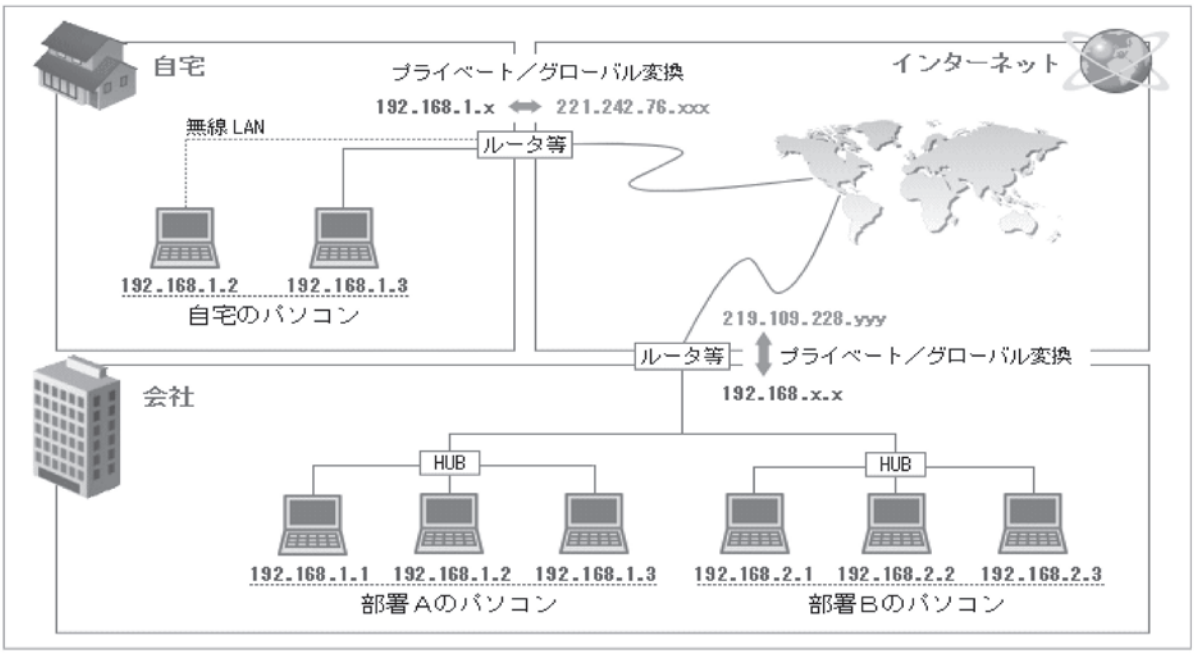
◇インターネットの基礎になっているのはデータ通信
◇デジタルデータに変換して別のコンピュータに送信
◇IP アドレス
　インターネット上の通信端末に割り振られているアドレス

① インターネット通信を支える IP

インターネット上で通信する端末は必ず IP アドレスというインターネット上の住所を示すものを持ち、インターネットで使われているネットワーク機器は、その IP アドレスを手がかりに、目的の IP アドレスを持つ端末までデータを届ける。

例えば、PC-A（IP アドレス=1.1.1.1）がサーバ B（IP アドレス=2.2.2.2）に通信するときは、PC-A が、送りたいデータの先頭に「送信元 IP アドレス=1.1.1.1」「宛先 IP アドレス=2.2.2.2」という情報を付ける。

間の NW 機器は宛先の 2.2.2.2 という IP アドレスがどこに存在するかを知っているので、適した NW 機器に転送する。次の NW 機器でも同様の所作を行う。これをルーティングと言う。このルーティングにより、サーバ B に届く。



青色は、プライベート IP アドレス　　赤色は、グローバル IP アドレス

（出典：サーバ監視／ネットワーク監視サービス）

② IP アドレス

IP アドレスは、32bit で表現され 8bit(0～255)毎にドットで 4 箇所に区切

られ、端末やルータは 0.0.0.0～255.255.255.255 の間のどれか適した値を IP アドレスとして割り振ることができる。適したと表現したのは、この中でも一般的には使えないものが混じっているためである。

IP アドレスには大きく分けて、グローバル IP アドレスとプライベート IP アドレスに分かれる。IP が誕生したときはこのような区分は無く全てがグローバル IP アドレスの位置づけであったが、インターネット利用者が予想を超えて莫大に増え IP アドレスが枯渇してきたため、このような区分ができた。

③ グローバル IP アドレス

グローバル IP アドレスはインターネットに接続するために必須のもので、好き勝手に設定してしまっても通信ができない。このグローバル IP アドレスが重複しないように管理をしているのが IANA という組織で、その下位組織としてアジア圏であれば APNIC、そのさらに下位組織として日本であれば JPNIC がいる。

日本の ISP 事業者は JPNIC から IP アドレスを割り当ててもらい、それを再販する形でインターネットサービスを運営している。

④ プライベート IP アドレス

一方インターネットを使わないような NW、たとえば組織内だけで使う NW であれば、自分たちの好きな IP アドレスを付与してしまって問題がない。そのための IP アドレスが、プライベート IP アドレスである。

⑤ プロトコル

通信での送受信の手順を定めた規格のことである。異なるメーカーのソフトウエアやハードウエア同士でも共通のプロトコルに従うことによって、正しい通信が可能になる。インターネットでは通信相手とのデータの送受信を TCP/IP(TCP と IP)と呼ばれる基本プロトコルで行い、そこで送受信されるデータ内容の表記を別のプロトコルで定めることによって、各種のサービスを実現している。たとえばウェブでは HTTP、ファイル転送では FTP、メール転送では SMTP である。

⑥ OSI 参照モデルと階層化

OSI 参照モデルとは国際標準化機構(ISO)によって策定された、コンピュータの持つべき通信機能を階層構造に分割したモデルである。通信機能（通信プロトコル）を以下の 7 つの階層に分けて定義している。

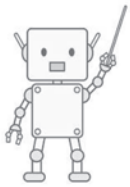
・ 第７層「アプリケーション層」：具体的な通信サービス（ファイルやメールの転送、遠隔データベースアクセスなど）を提供、HTTP や FTP 等
・ 第６層「プレゼンテーション層」：データの表現方法（たとえば EBCDIC コードのテキストファイルを ASCII コードのファイルへ変換）
・ 第５層「セッション層」：通信プログラム間の通信の開始から終了までの手順（接続が途切れた場合、接続の回復を試みる）
・ 第４層「トランスポート層」：ネットワークの端から端までの通信管理（エラー訂正、再送制御等）
・ 第３層「ネットワーク層」：ネットワークにおける通信経路の選択（ルーティング）、データ中継
・ 第２層「データリンク層」：直接的（隣接的）に接続されている通信機器間の信号の受け渡し
・ 第１層「物理層」：物理的な接続。コネクタのピンの数やコネクタ形状の規定等

⑦インターネットが社会にもたらした恩恵、メリット
・情報の流れが非常にスムーズになりどんな情報も比較的簡単に入手できる
・メールやメッセンジャー(業界では IM と呼んでいる)の普及により、簡単に人とコミュニケーションできる
・さまざまなサービスがインターネットに対応することにより、飛行機やホテルの予約、銀行の取引、保険の購入を初めとしてさまざまな取引をインターネットのみで行える

⑧デメリット
・情報やサービスがインターネットで利用できる反面、利用しない、できない人に対する差別は広がっていく
・コミュニケーションが簡単になることにより、より深く人と交わることが少なくなる
・インターネットはグローバル化を強制する

（参考資料 2020 SE の道標「【図解】初心者にも分かるインターネット IP ネットワークの仕組み ～IP アドレスとルーティングについて～」）

（２）インターネットアプリケーション（RIA：リッチインターネットアプリケーション）



ポイント

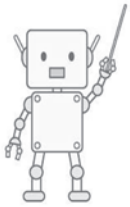
◇リッチインターネットアプリケーション
・Web ブラウザで単純な Web ページを表示する方式を超える表現力や操作性を備えた Web アプリケーション

RIA とは、柔軟で優れた表現力や操作性を持った Web アプリケーションの総称である。RIA は Web ブラウザ上で動作し、DHTML や Java アプレット、Flash、Ajax などの機能を使用して動的・双方向的な Web ページを表現する。

従来の HTML や CSS のみで表現された Web アプリケーションは、もっぱら静的なページをページ遷移によって切り替えていく方法だったが、RIA を導入することによって「動き」や「流れの良さ」があるユーザビリティの高いサイト構築が可能になった。RIA の表現を完全に再現するには、閲覧する側（Web ブラウザ）が RIA の提供する機能に対応している必要がある。

ただし RIA で必要な機能の多くは、主要な Web ブラウザによってサポートされているか、あるいは Web 上に無償で配布されている。

（３）インターネットアプリケーションが成り立つ原理



ポイント

◇クライアントサーバシステム
・コンピュータをサーバとクライアントに分け役割分担
・スマートフォンのアプリケーションで多く採用されている

① クライアントサーバシステム

クライアントサーバシステムとは、コンピュータをサーバとクライアントに分け役割分担をして運用する仕組みのことである。LAN における典型的なクライアントサーバシステムとしては、全員が共有しておきたいデータがおいてある「サーバ」があり、そこに一般のユーザが使う「クライアント」が複数接続されている状態である。

クライアントはサーバに対して「このデータを送ってほしい」という要求をし、それに対してサーバは要求されたデータを送る、といったことを行う。

② インターネットで使われている技術 TCP/IP（ティーシーピーアイピー）

インターネットで使われている標準のプロトコルである。プロトコルとは相互にデータをやり取りするための決まりごとである。現在のオペレーティングシステムでは標準で TCP/IP の機能が組み込まれている。

また、標準で組み込まれているため、インターネット以外のネットワーク、たとえば LAN を構築するときなどにも使われている。

③ IP アドレス（アイピーアドレス）

インターネットに接続されたコンピュータを識別するための番号。通常のインターネット接続では自動的に割り振られる。

しかし、LAN を構築するときなどには、自動的に割り振るよりも、固定して使ったほうが便利な場合が多いため、固定して使うこともある。

④ HTTP

ハイパーテキスト・トランスファ・プロトコルの略で、ホームページのデータを転送するために必要な決まりのひとつである。

⑤ FTP

ファイル・トランスファ・プロトコルの略で、インターネットを通じてファイルをやり取りするための決まりごとである。インターネットの初期（ホームページもない時代）から使われてきたプロトコルで、HTTP などに比べてサーバへの負荷が少ないのが特徴である。

現在ではファイルのダウンロードをするときに使われるほか、自分で作ったホームページをサーバへ転送して公開するときによく使われる。

⑥ P2P

peer-to-peer の略。インターネットにおいて一般的に用いられるクライアント・サーバ型モデルでは、 データを保持し提供するサーバとそれに対してデータを要求・アクセスするクライアントという 2 つの立場が固定されているのに対し、P2P は各ピアがデータを保持し、他のピアに対して対等にデータの提供および要求・アクセスを行う自律分散型のネットワークモデルであり、サーバまたはクライアントのそれぞれの立場に固定されることがない。

⑦ DNS

Domain Name System の略で、その名前が示すようにインターネット上でドメイン名を管理・運用するために開発されたシステムである。DNS はインターネットを利用するうえでなくてはならない存在であり、現在のインターネット

にとって必要不可欠なシステムの一つとなっている。

⑧ SMTP, POP

SMTP とは「メールを送信する仕組み」。POP は「メールを受信する仕組み」ことである。「Simple Mail Transfer Protocol（シンプル・メール・トランスファ・プロトコル)」の略で、あえて訳せば「簡単なメールの送信の手順」となる。

POP は「Post Office Protocol（ポスト・オフィス・プロトコル)」の略で「郵便局の手順」である。なお「SMTP」サーバと「POP」サーバは同じ名前であることが多い。つまり一台二役しているということが多い。

（４）5G の取組みについて

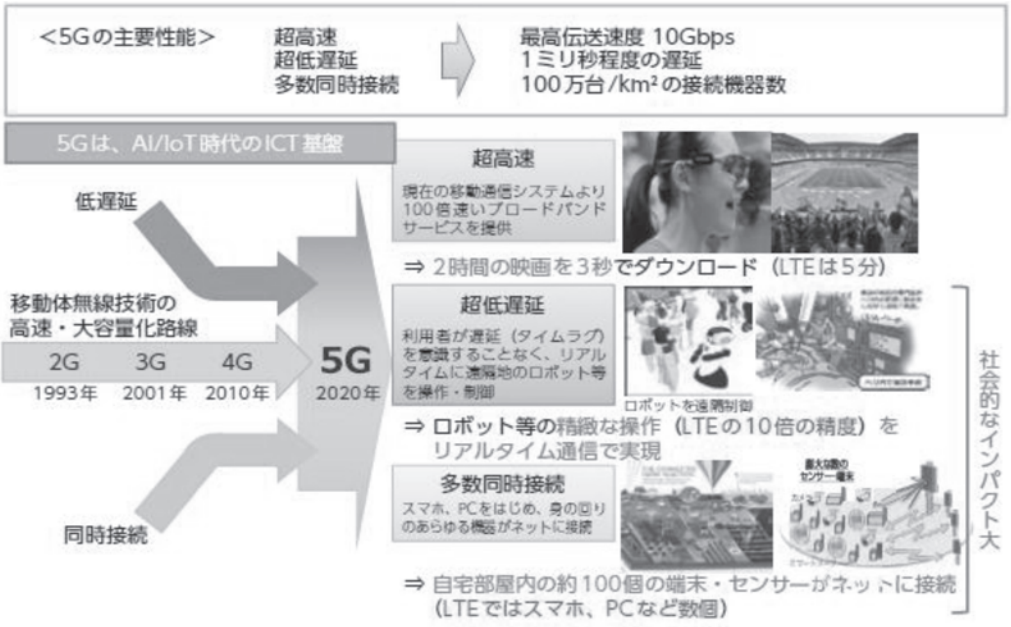
① 5G への期待

無線通信技術の急速な進展と人々のワイヤレスサービスに対する利用ニーズの高度化、多様化に伴い、携帯電話・スマートフォンについては、3.9 世代移動通信システム（LTE）や第 4 世代移動通信システム（4G）の導入による通信速度の高速化と情報量の大容量化が進んでいる。

最近、4G の次の移動通信システムとして、 5G の実現が世界的に期待されている。5G によって、4G を発展させた「超高速通信」だけでなく、多数の機器が同時にネットワークに繋がる「多数接続通信」、遠隔地でもロボット等の操作をスムーズに行える「超低遅延通信」が可能になる予定である。

そのため、5G は、あらゆる「モノ」がインターネットにつながる IoT 社会を実現する上で不可欠なインフラとして大きな期待が寄せられている。

【5G の特徴】



出典：2019 年通信白書

② 移動通信システムの歴史

システム	年代	機能等
１G	1980 年～	・初めて「携帯電話」が登場した。 ・音声を電波に乗る信号に変換して通信するアナログ方式である。 ・そのため、機能は音声通話のみ。
２G	1990 年～	・通信方式はアナログからデジタルへと変わった。 ・データ通信が容易になったことで、メールの利用やインターネット回線への接続が可能になった。 ・モバイルユーザーに対するビジネスも拡大。NTT ドコモに続き、KDDI の前身にあたる DDI セルラーグループは「EZweb」を、ソフトバンクの前身にあたる J-フォンは「J-スカイ」を開始した。 ・データ通信の利用が日常的になったことで、徐々に高速通信へのニーズが高まった。
３G	2000 年～	・通信速度が大幅に向上し、より大容量のコンテンツを楽しめるようになり、「着うた」などのプラットフォーム上のサービスが一気に増加した。 ・初の国際標準の移動通信システムが開始され、日本の携帯電話を海外でも使えるようになった。 ・快適にインターネットに接続できるようになり、高速大容量通信へのニーズが高まった。
４G	2010 年～	・スマートフォンの利用者数が激増した。 ・通信速度が飛躍的に向上したことで、スムーズなインターネット利用のほか、モバイルゲームや動画など大容量コンテンツを楽しめるようになった。
５G	2020 年～	・「高速大容量」「高信頼・低遅延通信」「多数同時接続」という 3 つが特徴である。 ・4G と比べて通信速度は 20 倍、遅延は 10 分の 1、同時接続台数は 10 倍の進化が見込まれる。 ・4G が「スマートフォンのためのモバイルネットワーク技術」とするならば、5G は「社会を支えるモバイルネットワーク技術」といわれている。

③ 5G の実現に向けて

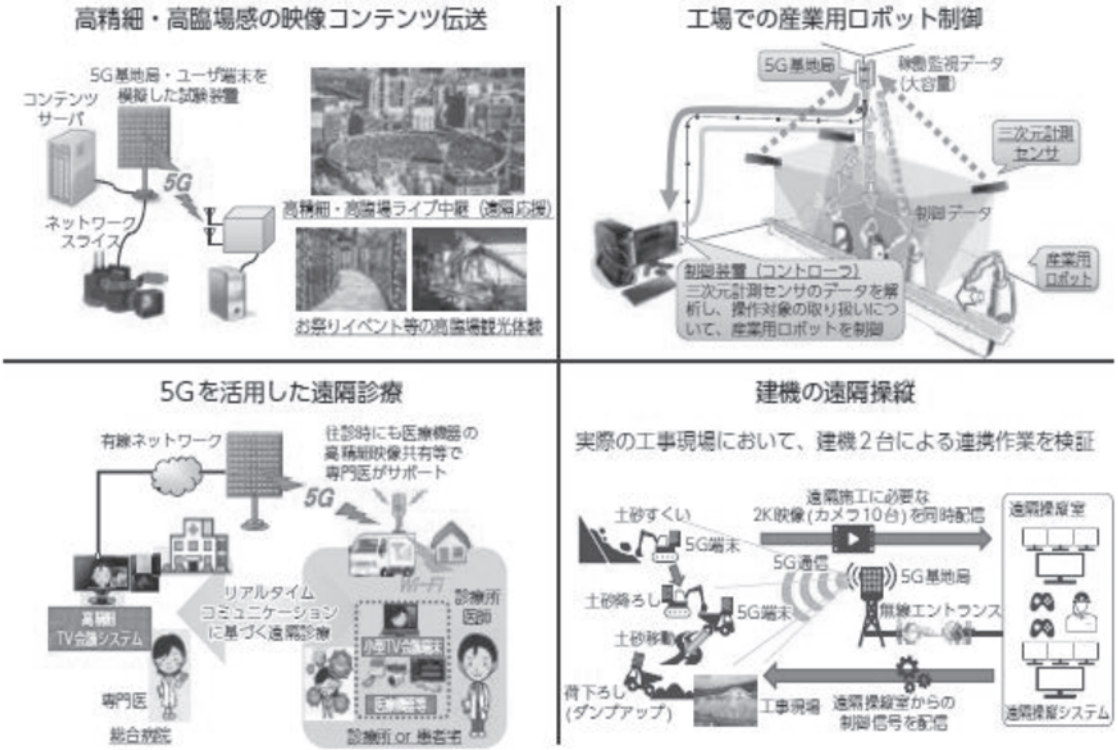
総務省は、5G の 2020 年（令和 2 年）までの実現に向けて、①研究開発・総合実証試験の推進、②国際連携・協調の強化、③5G 用周波数の具体化と技術的条件の策定といった取組を推進している。

5G の実現に不可欠な要素技術の研究開発に 2015 年度（平成 27 年度）から

取り組んでいる。

また、2017 年度（平成 29 年度）からは、新たな市場の創出に向けて、実利用を想定した試験環境を構築し、様々な利活用分野の関係者が参加する 5G の総合的な実証試験を実施している。

【5G 実証試験例】

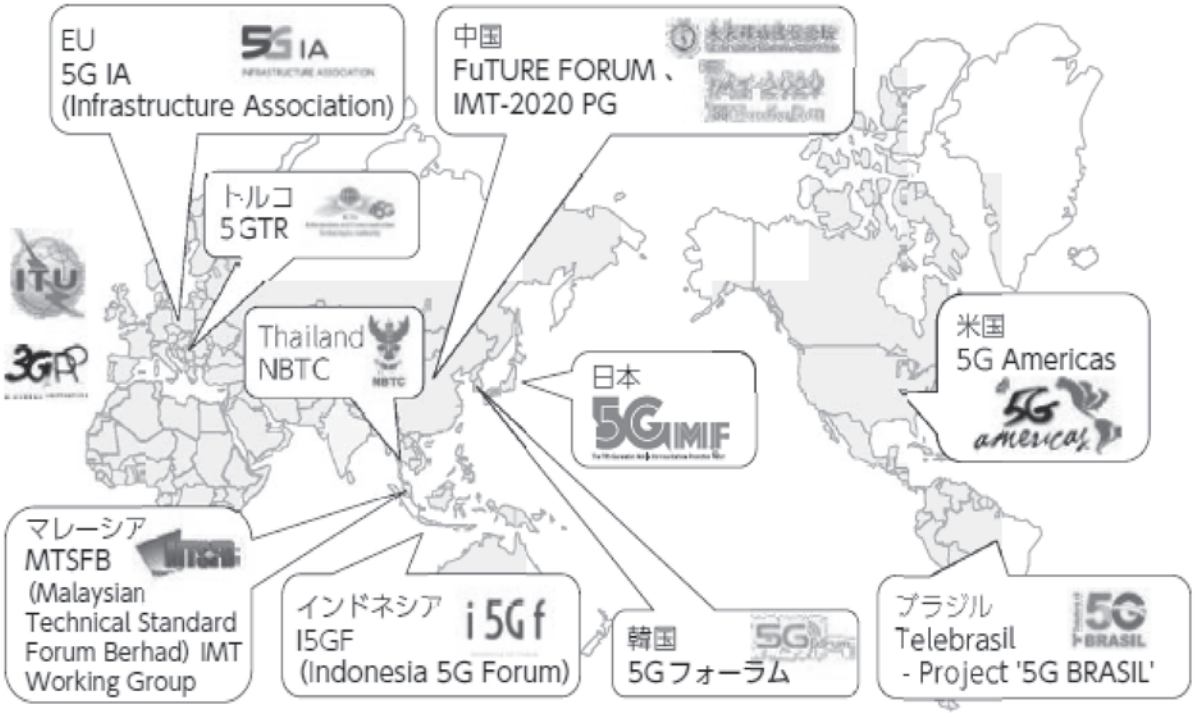


出典：2019 年通信白書

また、5G は経済や社会の世界共通基盤になるとの認識のもと、国際電気通信連合（ITU）における 5G の国際標準化活動に積極的に貢献するとともに、欧米やアジア諸国との国際連携の強化を進めている。

総務省としても、できる限り多くの携帯電話用周波数帯を確保できるよう、既存無線システムとの共用検討等を積極的に推進している。

【各国・地域の推進団体】

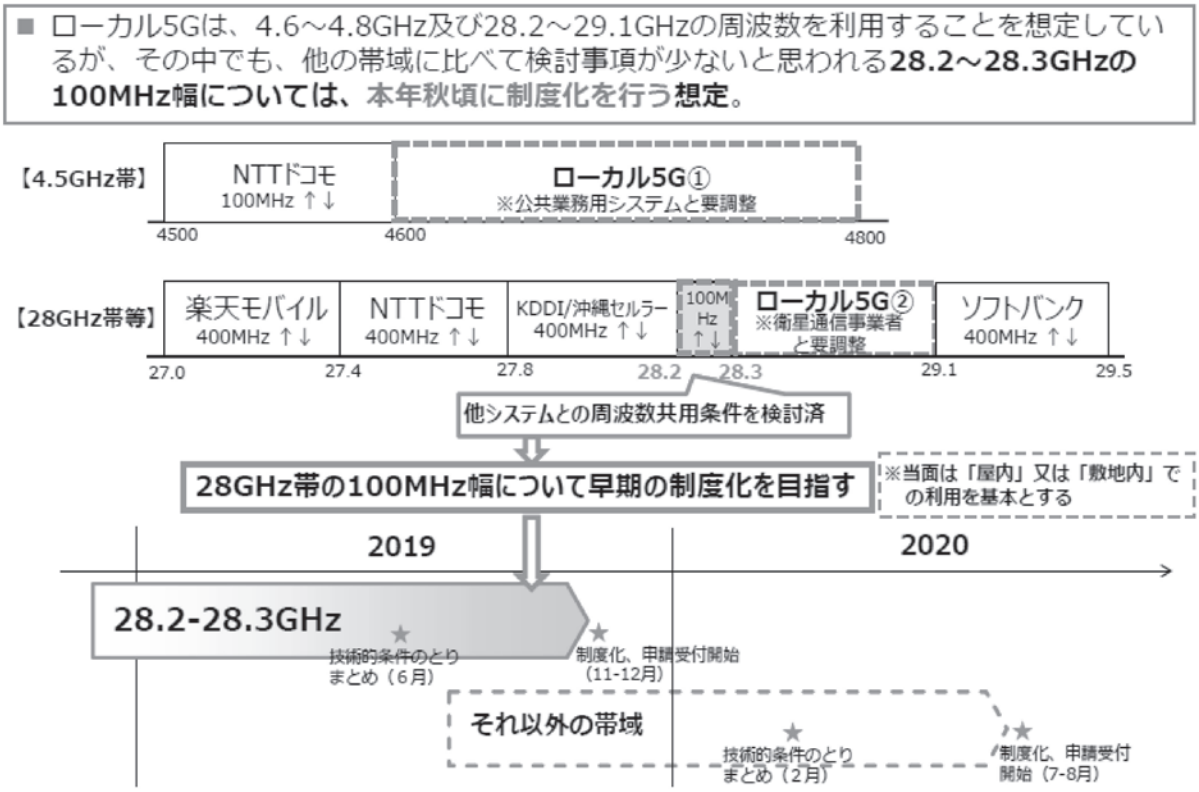


出典：2019 年通信白書

さらに、5G に使用する周波数を速やかに確保するため、国際的な動向等を踏まえつつ、情報通信審議会において、 5G 周波数確保に向けた考え方、既存無線システムとの周波数の共用、5G の技術的条件の策定等に関する検討を進め、2019 年（平成 31 年）4 月に申請のあった携帯電話事業者に対して 5G 用周波数を割り当てた。

周波数の割当てに際しては、2 年以内に全都道府県で 5G サービスを開始することを条件としており、今後順次、全国的に 5G が展開されていくこととなっている。

【ローカル 5G の候補帯域とスケジュール】



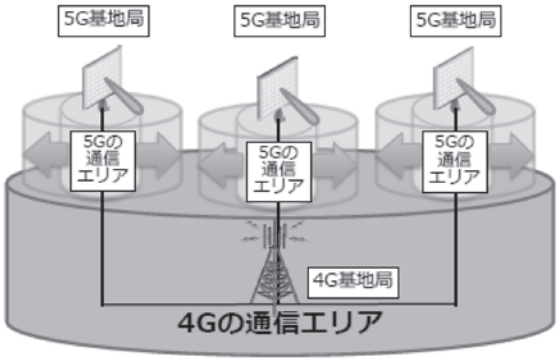
出典：総務省総合通信基盤局「第５世代移動通信システム（５Ｇ）の今と将来展望」

現在、IoT の普及に代表されるように通信ニーズの多様化が進んでおり、5G 時代においては、より一層の多様化が進むと想定されている。そのため、総務省では携帯電話事業者による日本全国でのサービス提供に加え、地域ニーズや個別ニーズに応じて、工場やスタジアムなどの様々な主体が 5G を活用したシステム（ローカル 5G）を導入できる制度の検討を進めている。

【４G のインフラをベースとしたローカル 5G システムの構築】

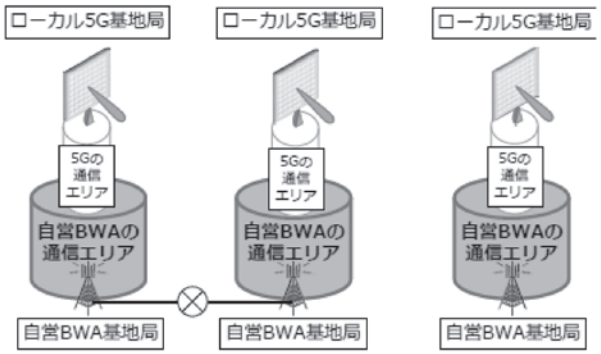
5G全国サービス

- 4Gのインフラをベースにエリアを確保しつつ必要な場所に5G基地局を設置。、ニーズに応じてエリアを拡大。



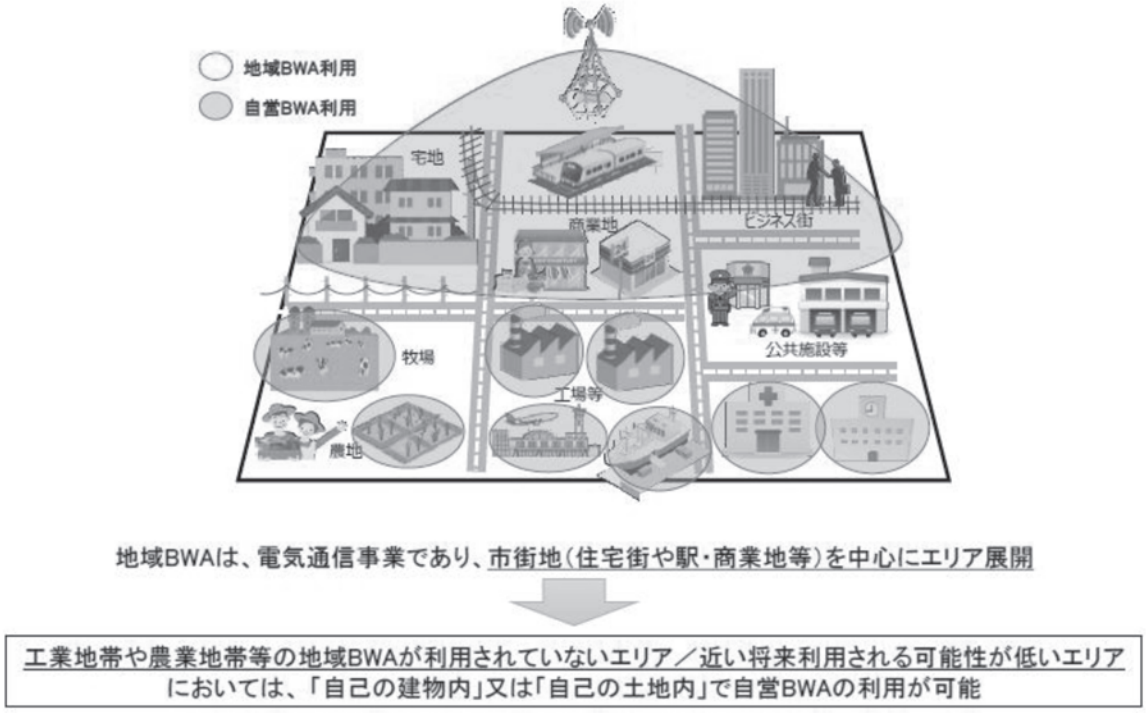
ローカル5G

- 全国サービスと同様に、自営BWA(4G)でエリアを構築しつつ、必要な場所にローカル5G基地局を設置。



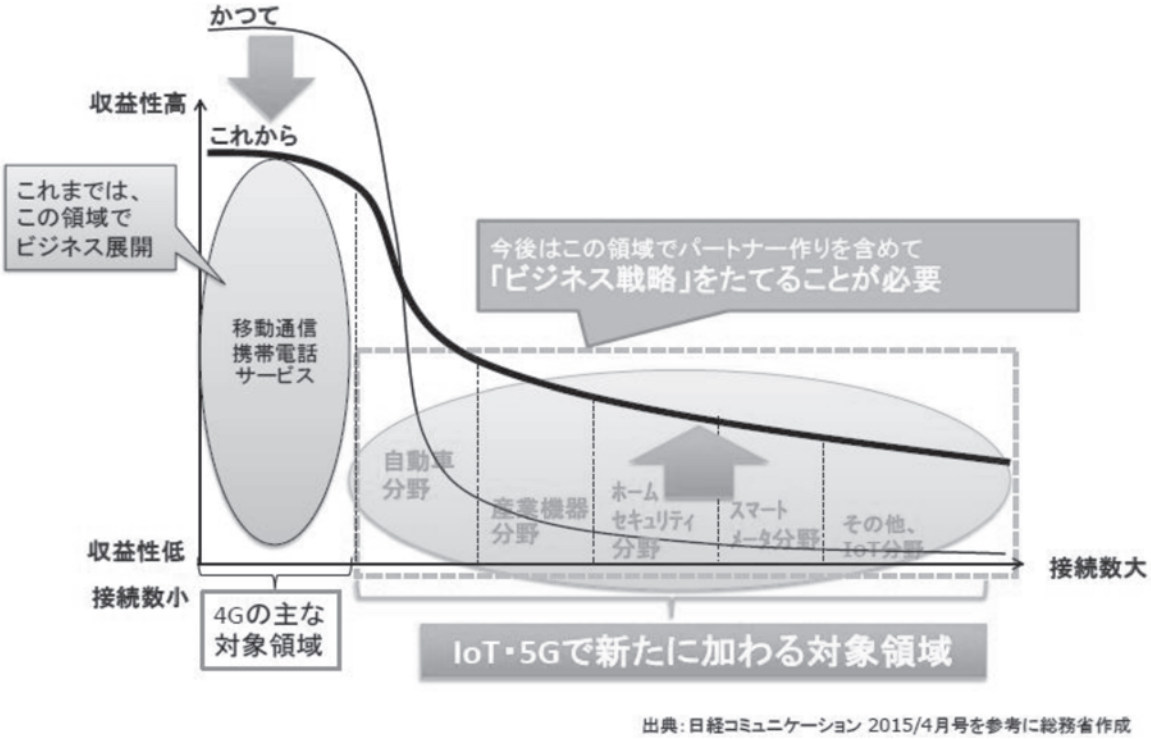
出典：総務省総合通信基盤局「第５世代移動通信システム（５G）の今と将来展望」

【地域 BWA 帯域における自営 BWA の導入イメージ】



出典：総務省総合通信基盤局「第５世代移動通信システム（５Ｇ）の今と将来展望」

【IoT 時代の構造の変化】



出典：総務省総合通信基盤局「第５世代移動通信システム（５Ｇ）の今と将来展望」

④ 5G を支える光ファイバの整備

このような 5G が地方を含む全国各地で早期に利用されるためには、中継回線としても利用される光ファイバなどの整備を促進する必要がある。

これまで我が国の光ファイバは固定系ブロードバンドとして整備が進められ、現在、整備率（世帯カバー率）は 2018 年（平成 30 年）3 月末で 98.3%となっている。しかし、過疎地域や離島などの地理的に条件不利な地域では整備が遅れている。

今後、2020 年（令和 2 年）の 5G の商用化をひかえ、中継回線としてのニーズも高まることが想定され、光ファイバの全国的な整備は、ますます重要になっている。

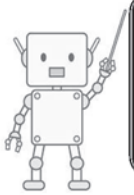
このような背景を踏まえ、総務省は 2019 年度（令和元年度）から、電気通信事業者等が 5G 等の高速・大容量無線局の前提となる光ファイバを整備する場合に、その事業費の一部を補助する「高度無線環境整備推進事業」を実施し、本事業によって光ファイバの整備を促進し、5G の速やかな全国展開を実現する。

— 総務省：2019 年度通信白書抜粋 —

第 2 章

ＡＩ基礎

1　ＡＩインターフェイス



ポイント

◇入力装置と出力装置
◇ユーザインターフェイスとその応用
◇AI とセンサ

（1）入力装置

ほとんどのパソコンには、入力装置としてキーボードが使われており、さらに座標を入力するマウスや音声を入力するマイクロフォンとサウンドボード、 画像を入力するカメラとビデオボードなどが備わっている。このほか、AI の」分野ではパソコンと連携して使用することが可能な入力装置として、イメージスキャナ、デジタルカメラなどがある。

① デジタルカメラ

デジタルカメラとイメージスキャナのほとんどは、画像入力デバイスとして CMOS センサや CCD（電荷結合素子）を使っている。これらは代表的なイメージセンサであり、その表面には、小さな受光素子であるフォトダイオードが数十万個から数百万個以上並んでいる。そして、それぞれの受光素子は入射された光の強さに応じた信号電荷を発生し蓄えることができる。

Nikon ホームページより

② デジタルカメラの原理

CMOS センサでは、各画素のフォトダイオードに付属したスイッチを次々に切り替えて、信号電荷を高速に読み出す仕組みである。

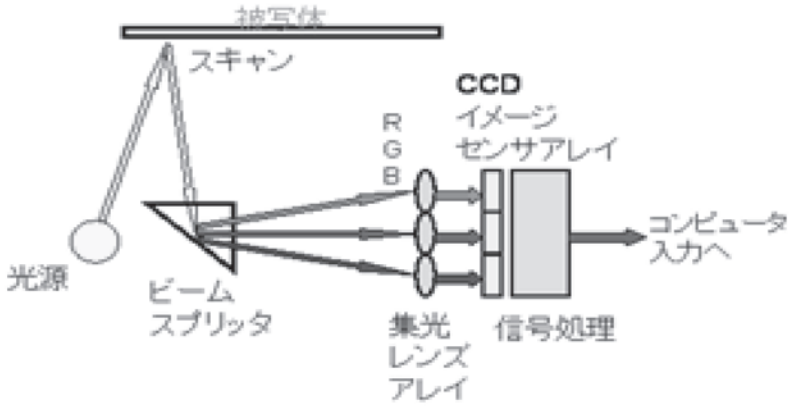
一方 CCD では、受光部の端には、電極が並べられた転送部があり、ここにある転送信号を与えると、全受光素子が蓄えている電荷が順次となりの電極に移動し、電荷を外部に出力する仕組みである。

受光素子の前には、R（赤）、G（緑）、B（青）の3色のカラーフィルタがあり、各色に対応して送り出された電荷は、A-D変換器によってそれぞれ8ビットのデジタル情報に変換される。すなわち、1画素につき24ビットの信号として出力される。

③ イメージスキャナ

イメージスキャナは、紙に書いたり印刷したりした文書や図面を光学的にスキャン（走査）して、このデータをA-D変換して画像としてコンピュータで取り込む装置である。単に、「スキャナ」とも呼ばれる。

イメージスキャナは、出版物や広告デザインの制作やWebページなどのマルチメディア制作に欠かせない機器である。



イメージスキャナの原理

④ 入出力インターフェイス

コンピュータと外部との間のデータを入出力するための入出力インターフェイスには、バス、ビデオボート、ネットワークアダプタなどがある。

◇バス

バスは、パラレル信号線でコンピュータのボードに直接接続されており、いくつかの規格がある。

ハードディスクドライブ（HDD）を接続するためのIDEバスは、フラットケーブルで増設のHDDやCD-ROMドライブを接続する。

ISAバス、PCIバスは、拡張ボードのための規格で、もともとはISAバスが標準だったが、32ビットのデータ幅など仕様が改善されたPCIバスが、今では広く使われている。

また、キーボード、マウスなどのインターフェイスを1本にまとめ、新しい機器も接続できるようにしたUSBや、マルチメディアデータの高速通信が可能なIEEE 1394バスも装備されている。

◇ビデオボード

ディスプレイの表示サイズや表示色数を拡張したり、動画データを取り込んだりする基板を、ビデオボードやビデオキャプチャボードという。

ビデオボードは、パソコンの拡張スロットに差し込んで使用するが、すでに装備されているパソコンが一般的である。さらに、カメラを内蔵したパソコンも増えている。ビデオキャプチャボードには、テレビチューナを内蔵したものや、動画圧縮チップを搭載したボードもある。

◇ネットワークアダプタ

コンピュータをLANに接続するためのハードウェアを、ネットワークアダプタと言う。ほとんどのLANは、国際標準化された「イーサーネット」という規格である。「イーサーネット」には、使用するケーブルによって最大通信速度が決まり、10BASE-2、10BASE-T、100BASE-TXなどが規格化されている。

◇サウンドカード

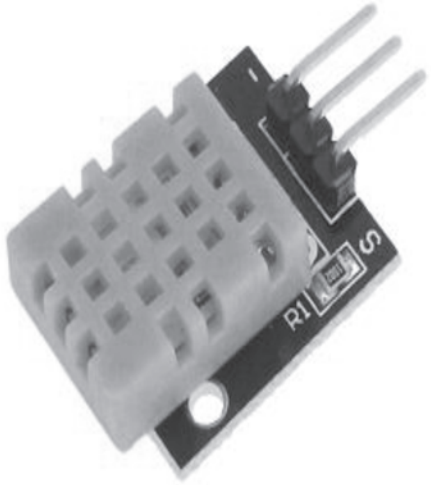
オーディオ信号を入出力するための基板が、サウンドカードまたはサウンドボードである。拡張スロットに差し込んで使用する場合もあるが、最初からパソコンのボードに搭載されている場合がほとんどである。

◇センサ

センサは私たちの日常生活の至るところに存在している。例えば部屋の中にある、エアコンには温度や湿度を計測する温度センサや湿度センサがある。

玄関に設置されているインターホンにはカメラやマイクがあり、イメージセンサや音センサを備えている。夜になると自動的に点灯する外灯は、日照の明るさを光センサで検知している。

また、スマートフォンでは、現在地を確認するためのGPSや手ぶれ防止のためのジャイロセンサがある。



温湿度センサモジュール

USB接続GPSモジュール

【 IoTで使用されるセンサの種類とその用途の一部】

センサ名	用途
温度センサ	温度を測定するセンサ。 エアコン、空調機器関連。
湿度センサ	湿度（空気中の水蒸気の割合）を測定するセンサ。 自動換気扇、空調機器関連。

圧力センサ	気体や液体などの圧力を測定するセンサ。 血圧計、洗濯機や風呂の水位測定。
光センサ	光の強弱や断続を測定するセンサ 受光部だけのものと、発光部と受光部を両方持つものがある 外灯、人やモノの有無検出
地磁気センサ	方角を知るため地磁気を検知するセンサ カーナビ、スマートフォン
GPS	人工衛星を利用した位置情報計測システム カーナビ、スマートフォン
加速度センサ	物体の移動に伴う速度の変化を検知するセンサ カーナビ、エアバッグ（自動車）
ジャイロセンサ	物体の回転を検出するセンサ。 カーナビ、スマートフォン、デジカメ。
CMOS イメージセンサ	画像を２次元の受光面に結像させ、電気信号に変換するセンサ デジカメ、Web カメラ、車載カメラ
音センサ	音の振動を計測するセンサ マイクロホン、携帯電話、録音機器
距離センサ	測定対象物との距離を測定するセンサ。TOF 式等がある 自動運転、自律移動型ロボット

◇データグローブ

データグローブとは、コンピュータと人間とのインターフェイス用装置の１つであり、人間の手の単純な動作から情報入力を直感的に行なえるセンシング装置として考案された電気仕掛けの高価な手袋である。

指の曲がり具合のような物理データの取得に多様なセンサ技術が用いられ各種形式が存在するが、一般への普及は限られている。限定的ながらコンピュータからの反応出力も行なえるものもある。

片手だけものが多いが、両手で使用するものもある[1]。 比較的低価格のものは指の曲がりだけが計られるが、価格に応じて、指同士の角度や手首の動き、外部装置と組み合わされるなどして手の絶対位置や姿勢まで読み取れるものもある。

指の曲がり具合は光ファイバや抵抗素子などで測定され、手の絶対位置や姿勢データは磁気センサや慣性センサといったモーション・トラッカーにより計測される。これらの動きは直ちに専用のソフトウェアによって解釈され、１つの動作から多数のデータが生じる。ソフトウェアによっては、手の特別な動きはジャスチャーとして判別されてコンピュータへは命令として伝えられる。

高機能なデータグローブは、グローブ内の指先を押すことで手が触った感覚を再現する、触覚フィードバック機能まで備え、情報入力だけでなく出力装置としても使用できるものがある。

（２）出力装置

コンピュータのマルチメディア情報の出力装置として代表的なものに、ディスプレイとプリンタがある。このほか、オーディオの出力装置として、スピーカやヘッドホン端子を備えたコンピュータが多くなっている

① ディスプレイ

コンピュータのディスプレイ上の文字や図形は、小さな点（画素、ピクセル）の集合体であり、この点が１インチあたり何個表示されるかで表示の密度がきまる。現在実用化されているディスプレイを、表示方式から分類したものを表に示す。

表 5.1 ディスプレイの種類

分類	表示方式
受光型	透過型液晶(バックライト付き) 反射型液晶
自発光型	有機 EL ディスプレイ、プラズマディスプレイ、CRT ディスプレイ

ディスプレイは、自身で光を出す「自発光型」と、他の光源を利用する「受光型」に分類できる。CRT（Cathode Ray Tube ： ブラウン管）やプラズマディスプレイパネル（PDP ： Plasma Display Panel)、有機 EL（Electro-Luminescence）現象を利用した有機 EL ディスプレイなどは自発光型であり、液晶ディスプレイ（LCD）は受光型である。

◇CRT ディスプレイ

CRT ディスプレイは、テレビのブラウン管と同じで、管面の内側に塗られた赤(R)、緑(G)、青(B)の蛍光体に、細く絞った電子ビームをあて、これを順にスキャンして管面全体に文字や図形を描いている。

画面は、精細で明るく優れているが、奥行きのサイズが大きく重いことが欠点で、消費電力も大きくなります。CRT ディスプレイは、液晶ディスプレイに置き換わっている。

◇液晶ディスプレイ

液晶（LCD:Liquid Crystal Display）という言葉は、「液体でも結晶でもない中間の状態にある物質」をさしている。液晶の分子は、その並び方が電界によって変化する性質を持ち、電圧をかけて電界を与えることにより光の透過率が大きく変わる。

液晶ディスプレイは、この「電圧で透過光の量を調節できる性質」を利用し、電極をつけた液晶をひとつの表示画素として多数並べておき、RGB のカラーフィルタと組み合せて、フルカラー表示を可能としている。

◇有機 EL ディスプレイ

携帯電話や PDA などの小型ディスプレイや、大型テレビのディスプレイで実用化されている有機 EL(Electro-Luminescence)機能を用いたディスプレイが、有機 EL ディスプレイで。ある有機化合物に電圧をかけて発光させる発光ダイオードの一種で、薄型で軽量なメリットがあり、高いコントラストが実現でき視野角も広いという特徴がある。

② プリンタ

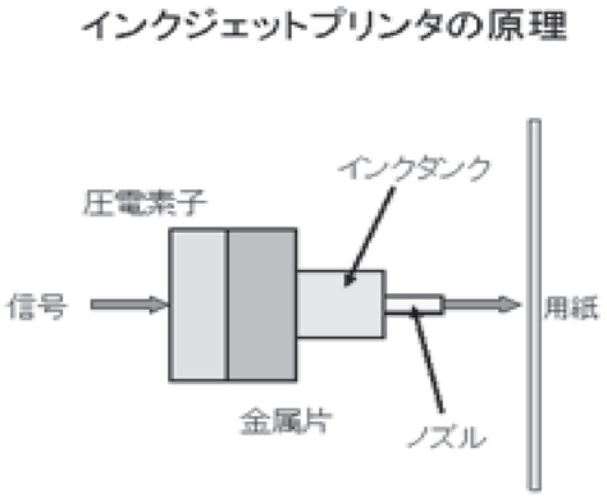
現在、コンピュータのプリンタとして広く使われているものには、レーザプリンタとインクジェットプリンタがある。

まず、レーザプリンタだが、静電気を帯びた回転するドラムに、コンピュータからのプリントイメージ情報に従って、レーザ光を当てて不要な部分の電荷を中和する。

次に、静電気が残っているところにトナー（顔料の粉）を付着させ、できたトナーの画像を紙に転写し、最後にトナーが取れないように熱を加えて定着させる。

レーザプリンタは、従来のタイプライタやインパクトプリンタのように、物理的に文字の型とインキを紙に押し付けて転写するわけではないので、音が静かで高品位で高速な印刷が可能である。しかし、価格が少し高く、消費電力が大きいことが欠点である。

一方、インクジェットプリンタは、ノズルからのインクを紙に吹き付けて印字する方式である。



インクジェットプリンタの原理

③ 特殊出力

◇VR ヘッドセット

「VR」とは「virtual reality（バーチャル＝リアリティ）」の略だが、最近よく言われている VR とは、VR ヘッドセット・VR ゴーグルを使うことで、仮想空間にいるような体験できるもののことを言う。

仮想空間にいるような体験とは、360度見渡せるコンテンツの中に入り込んだような体験や、自分の動作が仮想空間に反映される体験のことを言う。

もともとは高度な処理ができるPCが必要で高価な機器を使わないと体験できないものだったが、スマホとレンズ付きのVRゴーグルを使って体験でき、独立型のVRヘッドセットも安くなってきており、日本でも普及の兆しをみせている。

◇３Ｄプリンタ

製造業で現在注目されている。

3Dプリンタは、以下の様に製造現場のあらゆる場面で活用されている。その理由の一つとして柔らかいゴム素材からアルミなど強度のある金属素材まで、様々な材料で造形物を出力できることが挙げられる。

3Dプリンタの造形方式には数種類の方式があり、方式ごとに使える材料や造形物の特性は異なる。

日本のものづくり現場では、市場の個々のニーズに合わせるため、多品種小ロットの高付加価値製品へのシフトがますます加速している。

そのような中、ものづくりの各工程において、3Dプリンタが活躍する領域がますます拡がってきていることから、製造業での3Dプリンタの活用が急速にすすみ、ものづくりを支える重要な役割を担っている。

・発期間の短縮
・コストの削減
・コミュニケーションの活性化
・作業効率/品質のアップ

（３）ユーザインターフェイス

「HMI」とは「Human Machine Interface（ヒューマン・マシン・インターフェイス）」の略で、人間と機械との間で情報のやりとりを行う境界という意味を持つ。一般的には、人と機械の間に介在し、人への情報提供、機械への指令などをやりとりする部分を指す。簡単にいえば、人と機器をつなぐ部分、つまり機器の表示やユーザの操作を受け付ける部分のこと。

情報というものは、文字や映像によって表現されたものだけを指すのでない。人間が、視覚、聴覚、触覚、嗅覚、味覚という「五感」などを通じて外界から受け取ったり、自分の身体や声を通じて外界に伝えたりするもの、それらもまた情報である。

これらの情報をやりとりして人間と外界の間をつなぐものを、「ヒューマン・インターフェイス」と呼ぶ。

身近にある例では、自動車の各種のスイッチ、ハンドル、レバー、ディスプレイ、カーナビ、パソコンのディスプレイ画面やキーボード、マウスなども、典型的なヒューマン・インターフェイスである。

（４）インターフェイスの応用（各分野）

① 自動運転

「認知」、「判断」、「操作」の３つの処理を連動して行う、というのが自動運転の基本的な仕組みである。

まず、「認知」とは周囲の状況を検知することである。ミリ波レーダセンサと呼ばれる技術は進歩がめざましく、現在すでに衝突予防システム、先行車両との車間を調整するシステム、車線からの逸脱を防止するシステムにも採用されている。

「判断」は、検知によって得た情報から、どう車を動かすか意思決定を行うことである。この判断は人工知能（AI）技術の活用が待たれるところだが、今のところは「ルールベースシステム」という手法が使われている。これは交通ルールとも重なる明確なルールを数多く用意し、それらに基づくアルゴリズムによって判断を下すというものである。

「制御」は、認知と判断の結果を元に、自動で車の運転操作を行うことを指す。現在は、ある程度までの運転走行を制御する「半自動運転」が実現しているが、今後、実現すると考えられる完全自動運転では、車に乗る人はほとんど運転について関与せず、自動運転システムが車を動かすことになると予測されている。

◇自動運転を実現する主な技術

・ダイナミックマップ

高精度の三次元空間情報を持つデジタルマップのことある。カーナビなどに使用される 3D 地図だけでなく、交通事故、渋滞、周辺車両の進行状況、天候といった時間の経過によってリアルタイムに更新される情報を組み込んだマップデータである。

・高度道路交通システム(ITS)

人と道路と自動車という３者間で情報をやり取りして分析・処理を行うことで、事故や渋滞を解消し、さらに省エネや環境対策も実現しようというシステムである。自動運転との関連では、ITS との相互連携によって道路交通の最適化が行われる。

・人工知能（AI）

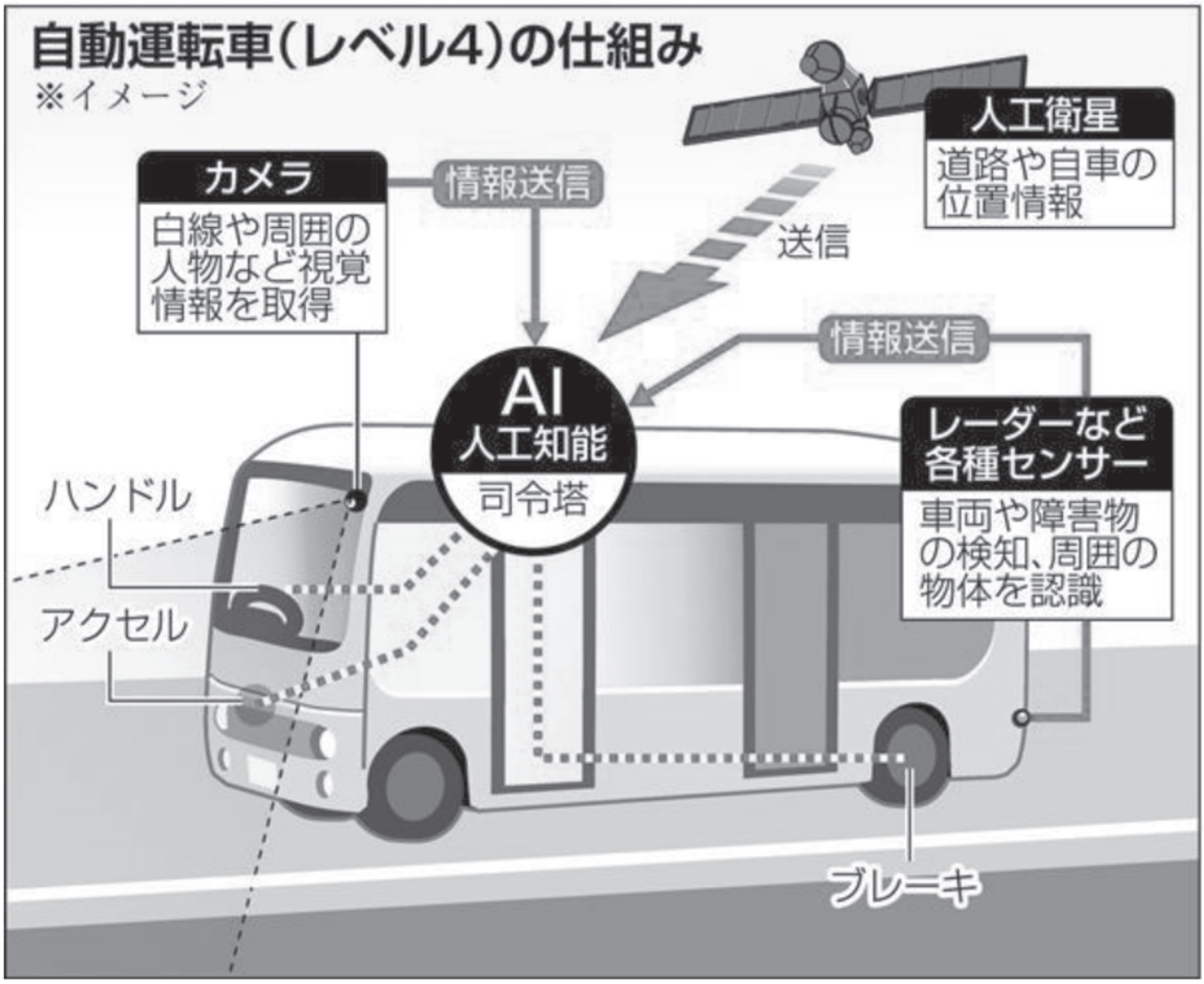
自動車メーカ、IT 企業など世界中の多くの企業が急ピッチで開発を進めている技術である。自動運転システムにおいては、AI はまさにシステムを司る頭脳として機能する。中でも注目されているのは、「ディープラーニング」と呼ばれる技術であり、これは AI が自らネットワークなどから情報を取得し、学習することで判断の精

度を高めていくというもので、自動運転システムにもこの技術が活用される可能性がある。

・バイワイヤ化

バイワイヤとは「by wire」、ワイヤは電線のことを意味し、自動車における機械式制御を電気信号に置き換えて制御する技術を指す。もし、車を電気信号で制御できるようになれば、AI が自動運転における「操作」を行いやすくなる。

つまり、バイワイヤは「認知」、「判断」、「操作」の間をつなぐ「伝達」をスムーズに実現するために必要な技術である。



② ロボットアーム

人間とロボットには、ある共通点がある。機械のロボットと人間……遠い存在のように感じますが、実は骨と関節の構造が一緒である。主にロボットアームで構成される産業用ロボットは、リンク（骨）とジョイント（関節）の組み合わせが基本的な構造で、人間の体で言えば、肘や肩など自由に曲がる部分がジョイント、その間を繋ぐ骨の部分がリンクということになる。ジョイントを動かしてリンクで力を伝えるという原理は、人間もロボットも同じである。

人間の肘や肩にある関節がジョイント、それを繋ぐ骨がリンク

③ チャットボット

「チャットボット（Chatbot）」とは、チャット（会話）とボット（ロボット）を組み合わせた言葉で、人工知能（AI）を活用した「自動会話プログラム」のことである。

LINE（ライン）、Facebook などでこのチャットボットを応用したサービスが提供されており、ユーザはまるで人間と会話するような感覚で AI との会話を通じて情報収集を行える。

◇チャットボットの仕組み

基本的にはアプリケーションと Bot（ボット）といわれるシステムを API で連携し、ボットシステム内で問いかけの解釈・返答生成を行い、API 経由でアプリケーションに戻される、という仕組みになっている。

◇チャットボットの種類

アルゴリズムによって以下の 4 種類に分けられる。

1.選択肢タイプ	データベースに蓄積されたシナリオや、設定された回答を選択して会話するタイプで、09 設定されていない受け答えはできない。
2.ログタイプ	会話を行った記録をログとして蓄積し、これを利用して人間に近づけた会話を行うタイプで、ログが蓄積されることによって、より自然な会話ができるようになるため、ログが少ない場合は会話が続かなくなる。近年では、ログ解析に AI を活用し、より人間の会話に近づける試みがされている。
3.ハッシュタイプ	辞書に登録されたテンプレートを元に、会話を行うタイプで、そのため辞書タイプと呼ばれることもあるが、範囲の限定された利用方法であれば、受け答えには問題はない。

4.Eliza タイプ	「Yes」「No」や相づちで返答しつつ、相手の言葉を要約したり聞き返したりすることによって会話するタイプで、チャットボットの原型ともいわれる Eliza から名付けられており、基本的には聞き役に徹するチャットボットといえる。 そのため辞書タイプと呼ばれることもありますが、範囲の限定された利用方法であれば、受け答えには問題はない。

④ 物流ロボットがピッキング

物流ロボットによるピッキングが自動化される近年、人工頭脳 AI の発達により、物流業界にも大きな波が押しよせている。

今まで人の手によって行って来た倉庫内仕分け作業は、ロボットが肩代わり。早く、正確に、さらに丁寧に商品を吸い取る。

物流の自動化、荷主のいる宅配便の仕分け、某巨大物流センターでは都内や航空貨物などで集められた荷物を全国70か所にあるターミナルに送るため、自動化による仕分けを行っている。

物流ロボットの自動化、ネット通販の物流倉庫ではここ数年で増化を見せている e コマース(ネットショッピング)の宅配も急速にロボットによる自動化が発達している。

⑤ 認識系(画像解析)

X 線、CT、MRI などの人体画像に基づく、重大な症状の診断は、専門医である放射線科の医師が行うことが一般的だが、画像は多い場合 1 人 600 枚ほどになり、診断に時間を要している。また、放射線科医が不足し、一人でも多くの患者を診断するために、効率よく診断できることが重要になってきており、AI(人工知能)技術の活用が注目を集めている。

画像から必要な情報を抽出し、統計的なデータを得る「画像解析」。1990 年代に PC が普及してきた頃から取り組みが本格化し、多種多様な分野において活用されてきた。たとえば、天文写真からの新天体発見、航空写真や衛星写真を用いた観測情報の統計化、医療画像による診断、事件・事故に関する調査および証拠、身近なところでは画像から文字を読み取る「OCR(Optical Character Recognition:光学的文字認識)」、指紋・虹彩・顔認証をはじめとする各種生体認証システムなども、画像解析の活用例といえる。

AI を用いた画像解析では、人間が事前にアルゴリズムを設定しておくだけで、「機械学習」によって AI が前処理から最終的な解析までの各工程を一貫して処理できるようになる。たとえば顔を認識させる場合、目/鼻/口の位置や形など“特徴点”となる部分が抽出できるように設定しておけば、あとは PC 側で背景と顔の判断を行う。

ア）認識系（音声合成・音声認識）

音声合成とは、機械の声を人の声に似せてつくる技術のこと。一方、音声認識は、テキスト（文字）を音声にしたり、音声をテキストにしたりすることをいう。音声合成と音声認識は全然別の技術ではないが、異なる点は多い。

イ）通訳とは

AI 通訳とは、“人工知能によって稼働する通訳・翻訳プログラム”である。

記憶や学習を司る人間の脳神経（ニューロン）の働きを機械で再現した「ニューラルネットワーク」によって、機械自体が人間と同じように事象を学習するという仕組みになっている。

さらに AI 通訳は常に学習するため、何度も翻訳を繰り返すことで、より翻訳の精度が上がっていく。

ウ）「コンシェルジュ」

宿泊施設に代わってお客様からの問い合わせに答えるコンシェルジュサービス。

AI が問い合わせ内容を判別し、チャット形式で 24 時間、即時に自動で応答する。宿泊施設は、お客様からの質問が集中するフロント業務を軽減し、その分、目の前のお客様に接する密度を高めることができるようになる。

（５）AI とセンサ

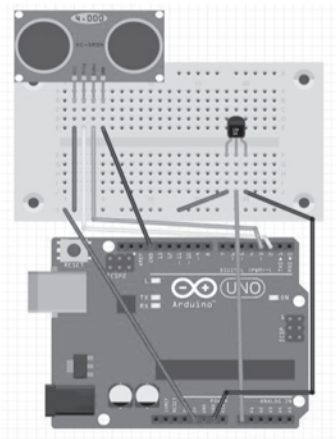
① 温度センサ

温度センサとは、文字通り物や空気の温度を計測することが出来る機械である。主に、接触式と非接触式の二種類のものがある。

接触式温度センサには、熱電対、白金測温抵抗体、サーミスタ測温体、バイメタル式温度計、液体充満式温度計および水銀温度計等があり、広く使われている。対して非接触式温度センサは、物体から発せられる赤外線を計測することで温度を測定する。

② 超音波センサ

超音波の反射時間を利用して非接触で測距するモジュールです。外部からトリガパルスを入力すると超音波パルス（8波）が送信され、出力された反射時間信号をマイコン（Arduino等）で計算することによって距離を測ることができる。

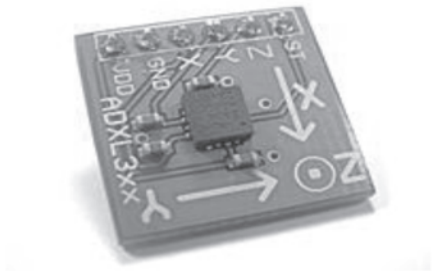


③ 加速度センサ

加速度センサは文字通り、1 秒における速度変化（加速度）を測定するセンサーのことである。

測定するものの中には重力加速度も含まれるため、人の動きや振動、衝撃まで検知できる。3 軸方向（X 軸・Y 軸・Z 軸）に適応するセンサであれば水平状態を検出でき、カメラの横方向による「手ぶれ補正」などにも加速度センサの機能が応用されている。加速度センサを用いる家電製品では、スマホやタブレット、ゲーム機のコントローラ、パソコンの HDD などがある。

加速度センサの一般的な測定方式は、バネと重りが一体化した部品に対して、加速度が発生した時の位置を測定するといったもの。それぞれの方向に取り付けた重りが、外部からの衝撃により移動することで、その位置変化をセンサ内部で測定。後はバネ係数などを考慮した力学の方程式（フックの法則）に数値を入力することで、加速度を導き出すことができるのである。



④ ジャイロセンサ

ジャイロセンサとは、基準軸に対して 1 秒間に角度が何度変化しているかを検知するセンサのことである。

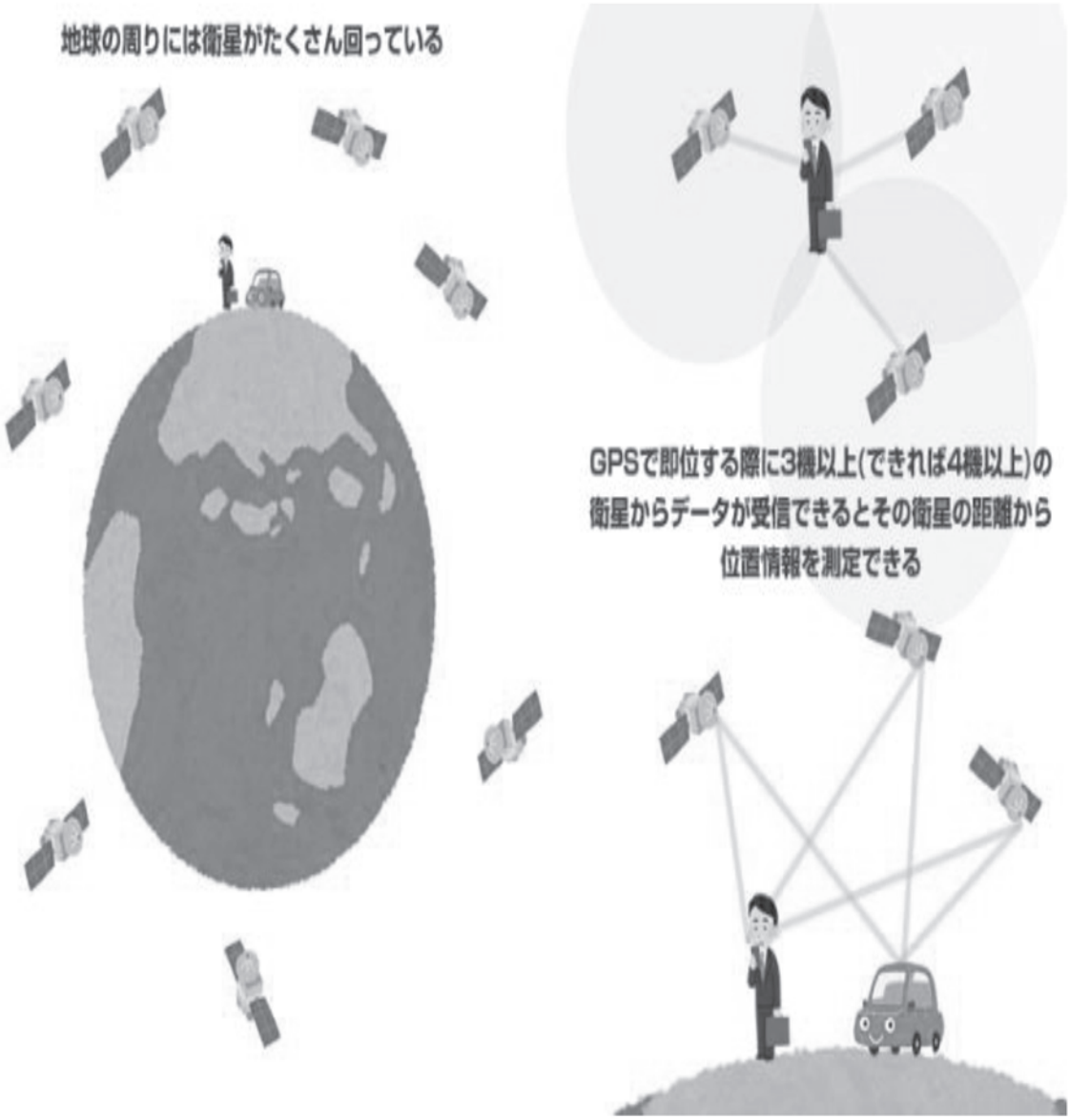
物体の回転運動を知ることができるため、加速度センサでは検知できない「回転の動き」を測定することが可能。

スマホやタブレッドなどで画面を傾けると、自動で見やすい方向に位置が切り替わるのも、ジャイロセンサの機能が応用されているからである。

ジャイロセンサはつい最近登場した技術というわけではなく、19 世紀頃から人工衛星や飛行機などに用いられていた。近年ではナノレベルで部品の小型化が進み、先にも述べたスマホやタブレット、カーナビ、腕時計などへ搭載されるようになっている。

⑤ GPS モジュール

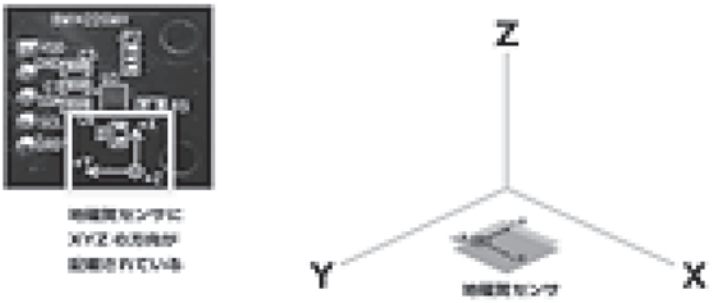
GPS のしくみは簡単に説明すると、上記図のように、地球上にたくさん回っている GPS で利用できる衛星からの電波を利用して、その衛星からの距離をもとに自分自身の位置情報を取得することができるしくみである。



⑥ 磁気センサ

磁気センサは、磁場（磁界）の大きさ・方向を計測することを目的としたセンサ。
測定対象磁場の強さ、交流・直流の別や測定環境等、目的に応じて多種多様な磁気センサが存在する。

用途は、純粋な磁場計測のみならず、電流センサ、磁気ヘッド、移動体探知器等、電気・電子系をはじめとして、ありとあらゆる工学分野にわたっている。



北の方角に向かってXYZ軸それぞれの向きを合わせたとき
センサの値（＝地磁気の強さ）が最大値になる

地磁気センサの種類は一般的に、地磁気センサは2軸のものと、3軸のものがある。2 軸の地磁気センサは、センサが地面に水平であることを仮定することで、方位を計算することができる。つまり、センサそのものが roll や pitch 方向に傾くと、上下方向の地磁気の影響で方位の精度が低下するという問題がある。

一方、3 軸の地磁気センサは、roll や pitch の情報が分かる場合は、その姿勢情報を利用して地磁気情報を補正することにより、センサの姿勢が変化した場合にも、方位を推定することができる。

⑦ 物体センサ

・フォトマイクロセンサ（透過形）

光を利用して物体の有無や位置を検出する小型のセンサである。コの字になって向かい合っているセンサ間の空間を物体が通過して光をさえぎることで感知するセンサ。

・フォトマイクロセンサ（反射形）

センサが発した光が検出物体にあたり反射した光の量を感知する。

・エリアセンサ

何本かの光の軸を出す光電センサで一本でも光がさえぎられると物体の有無や位置を感知するセンサで、広いエリアを検出するのに適している。

・光電センサ（透過型）

光を発する側と受ける側の間の空間を物体が通過して光をさえぎることで変わる光の量を感知するセンサで、検出距離が長く、不透明体であれば、材質・色・形状に関係なく検出が可能である。

・近接センサ

センサが発生させる磁界によって金属の接近を感知することができる。磁石につくものしか反応しないので動画では鉄板を使っている。

・ファイバセンサ

ファイバセンサをアンプに連結して物体の有無や位置を検出する。センサヘッドが小型化されているため狭い場所などへ自由に入り込んで検出できるようにしたもの。

・レーザセンサ

レーザ光によって物体の有無や位置を検出する小型のセンサ。レーザ光とはエネルギーを光としたきわめて純度の高い光である。動画では 1.5㎜ のアルミ板と 2㎜ の

アルミ板の厚みを感知して 2mm では反応しますが、1.5mm では反応しないようになっている。

・光電センサ

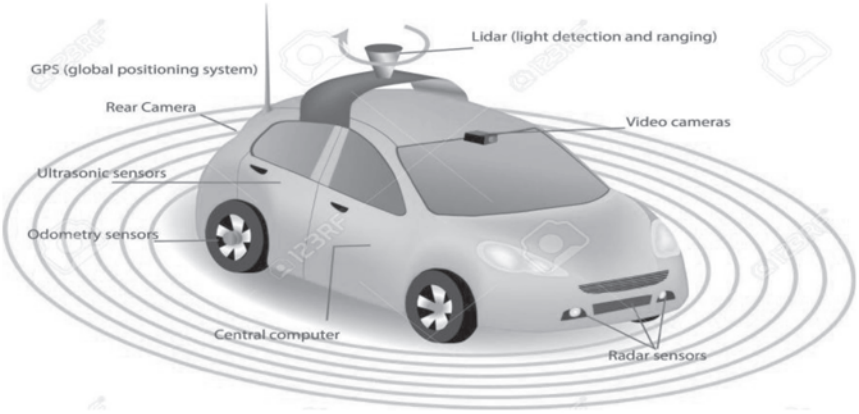
光電センサは、光線により物体が存在するかどうかを検出する。このテクノロジは、検知距離を長くする必要がある場合や、検知対象が金属でない場合に、誘導型近接センサにかわる最適な選択肢となる。

・誘導型近接センサ

誘導型近接センサは、金属製の物体を、触れることなく検出できる。このテクノロジは、検出対象の金属製の物体がセンサ面から 1～2 インチ以内にあるようなアプリケーションで使用される。、負荷の軽い梱包アプリケーションでも、自動車溶接機器での苛酷な環境や、食品加工工場における洗浄工程のような厳しい条件下においても使用することができる。

⑧ LIDAR

光を用いたリモートセンシング技術の一つで、パルス状に発光するレーザ照射に対する散乱光を測定し、遠距離にある対象までの距離やその対象の性質を分析するものである。

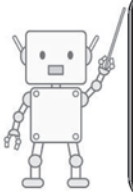


https://jp.123rf.com

この技法はレーダに類似しており、レーダの電波を光に置き換えたものである。対象までの距離は、発光後反射光を受光するまでの時間から求まる。

ライダーとレーダの最も基本的な相違は、ライダーはレーダよりも遥かに短い波長の電磁波を用いることである。典型的には紫外線、可視光線、近赤外線である。一般的に、検出できる物体や物体の特徴のサイズは、波長を下回ることができない。従って、ライダーはレーダよりもエアロゾルや雲の粒子の検出に向いており、大気の研究や気象学にとっても有用である。

2　ＡＩの概要



ポイント

◇AI の進化
◇AI のブーム
◇過去 2 回のブームが終わった理由

（1）AI の進化と利用の拡大

人工知能（AI）は、技術水準が向上しつつあるのみならず、既に様々な商品・サービスに組み込まれて利活用が はじまっている。身近なところでは、インターネットの検 索エンジンやスマートフォンの音声応答アプリケーションである米 Apple の「Siri」、Google の音声検索や音声入力 機能、各社の掃除ロボットなどが例として挙げられる。ま た、ソフトバンクロボティクスの人型ロボット「Pepper （ペッパー)」のように、人工知能（AI）を搭載した人型 ロボットも実用化されている。

（2）AI ブームの変遷

人工知能（AI）の研究は 1950 年代から続いているが、その過程ではブームと冬の時代が交互に訪れてきたとさ れ、現在は第三次のブームとして脚光を浴びている。

開発された年から実際に使いだすのは 10 年後である。

① 人工知能ブームの変遷

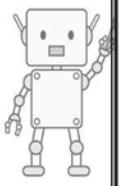
第一次人工知能（AI）ブーム	1950 年代後半～1960 年代である。コンピュータによる「推論」や「探索」が可能となり、特定の問題に対して解を提示できるようになったことがブームの要因である。冷戦下の米国では、自然言語処理による機械翻訳が特に注力された。 しかし、当時の人工知能 （AI）では、迷路の解き方や定理の証明のような単純な仮説の問題を扱うことはできても、様々な要因が絡み合っているような現実社会の課題を解くことはできないことが明らかになり、一転して冬の時代を迎えた。
第二次人工知能（AI）ブーム	第二次人工知能（AI）ブームは、1980 年代である。「知識」（コンピュータが推論するために必要な様々な情報を、コンピュータが認識できる形で記述したもの）を与えることで人工知能（AI）が実用可能な水準に達し、多数のエキスパートシステム（専門分野の知識を取り込んだ上で推論することで、その分野の専門家のように振る舞うプログラム）が生み出された。

	日本では、政府による「第五世代コンピュータ」と名付けられた大型プロジェクトが推進された。しかし、当時はコンピュータが必要な情報を自ら収集して蓄積することはできなかったため、必要となる全ての情報について、人がコンピュータにとって理解可能なように内容を記述する必要があった。 世にある膨大な情報全てを、コンピュータが理解できるように記述して用意することは困難なため、実際に活用可能な知識量は特定の領域の情報などに限定する必要があった。こうした限界から、1995 年頃から再び冬の時代を迎えた。
第三次人工知能（AI）ブーム	2000 年代から現在まで続いている。まず、現在「ビッグデータ」と呼ばれているような大量のデータを用いることで人工知能（AI）自身が知識を獲得する「機械学習」が実用化された。 次いで知識を定義する要素（特徴量＊11）を人工知能（AI）が自ら習得するディープラーニング（深層学習や特徴表現学習とも呼ばれる）が登場したことが、ブームの背景にある。

② これまでの人工知能ブームをふりかえって

過去 2 回のブームにおいては、人工知能（AI）が実現できる技術的な限界よりも、社会が人工知能（AI）に対して期待する水準が上回っており、その乖離が明らかになることでブームが終わったと評価されている。このた め、現在の第三次ブームに対しても、人工知能（AI）の技術開発や実用化が最も成功した場合に到達できる潜在 的な可能性と、実現することが確実に可能と見込まれる領域には隔たりがあることを認識する必要がある、との指 摘がある＊12。例えば、ディープラーニングによる技術革新はすでに起きているものの、実際の商品・サービスと して社会に浸透するためには実用化のための開発であったり社会環境の整備であったりという取組が必要である。 実用化のための地道な取組が盛んになるほど、人工知能（AI）が社会にもたらすインパクトも大きくなり、その 潜在的な可能性と実現性の隔たりも解消されると考えられる。

3　機械学習



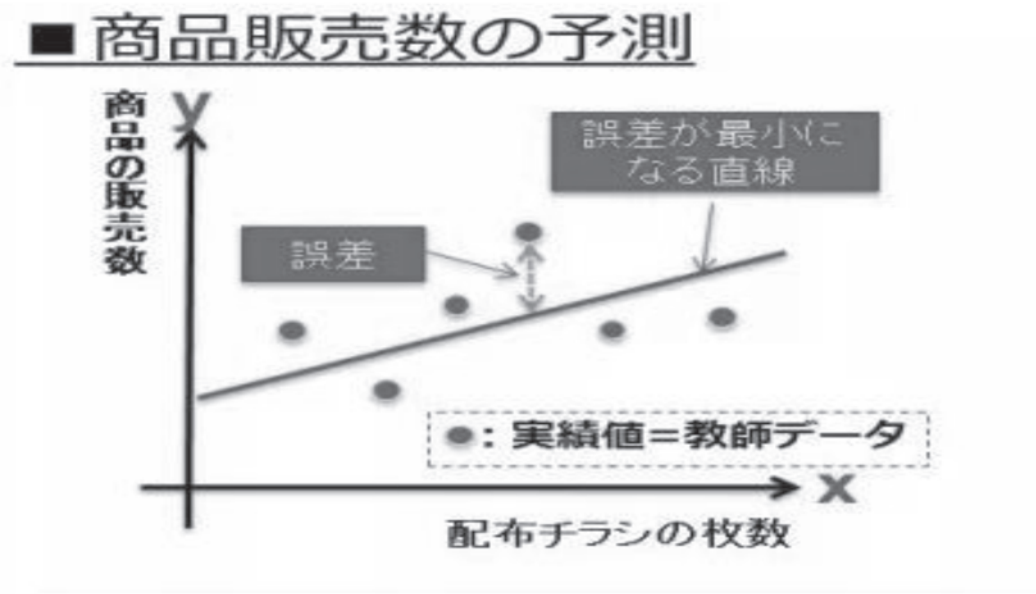
ポイント

◇機械学習の分類
　ア　教師あり学習
　　・学修データとして入力値とともに正解を与える
　イ　教師なし学習
　　・正解データを与えない
　　・入力データから、そのデータの構造、特性、新たな知見を学習する
　ウ　強化学習
　　・固定的で明確なデータを元にした学習ではない
　　・プログラム自体が与えられた環境（＝現在の状態）を観測し、各行動の、評価を自ら更新していき、連続した一連の行動の結果、価値が最大化する（＝報酬が最も多く得られる）行動・方法を自ら学習

機械学習とは予測や分類などのタスクを遂行するアルゴリズムやモデルを学習データから構築する手法である。機会学習は大きく分けて 3 種類がある。

（1）教師あり学習

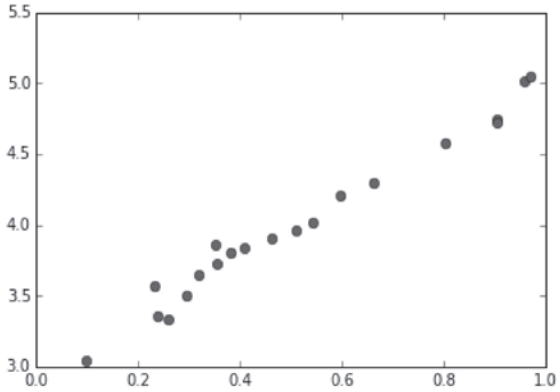
・「教師あり学習」は、学習データとして入力とその正解が与えられる。代表的なタスクは「識別」と「回帰」である。
・「識別」とは画像を入力し、正解としてあらかじめ定められたいくつかのクラスに部類するもの。
・「回帰」とは気温を入力しアイスクリームの販売額を予測するといったタスクである。
・「識別」とは画像を入力し、正解としてあらかじめ定められたいくつかのクラスに部類するもの
・「回帰」とは気温を入力しアイスクリームの販売額を予測するといったタスクである。

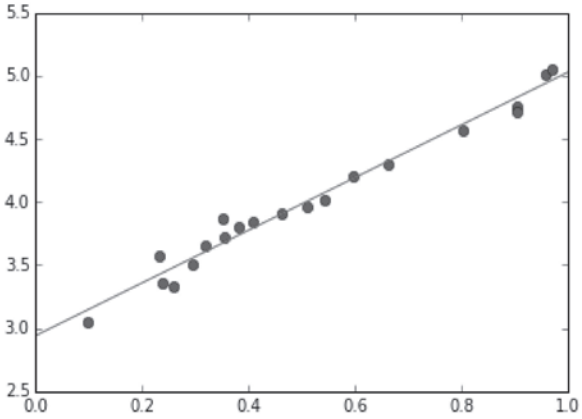


◇機械学習での教師あり学習は回帰問題と分類問題に分けられる。

【回帰問題】

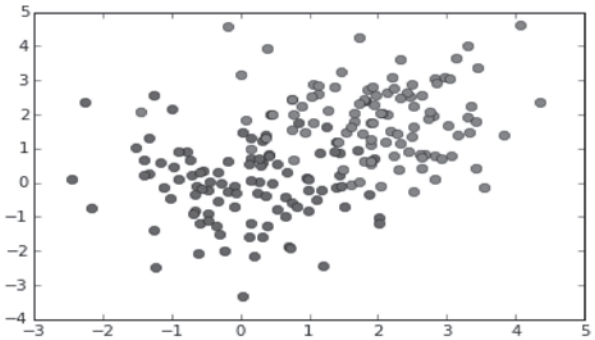
・データ群から線を求めるのが回帰問題というもので、株価の予測とか明日の気温の予測などに使われている。

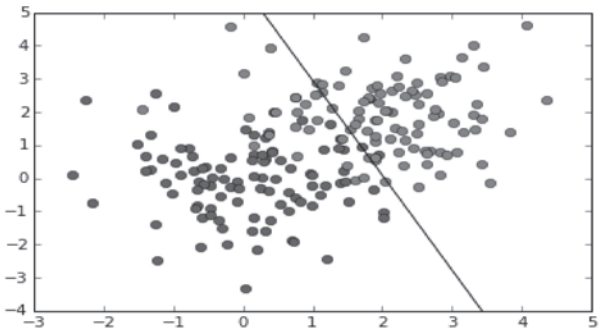




【分類問題】

・青と緑のおよその境目のところに線を引く。新しいデータが線の左側にきたら青に近しいデータ、線の右側にきたら緑に近しいデータと分類できる。スパムメールの分類とかに使われている。





（２）教師なし学習

・「教師なし」というのは、正解不正解のデータがないことである。そのままデータをつっこんでしまうのがこの学習方法である。

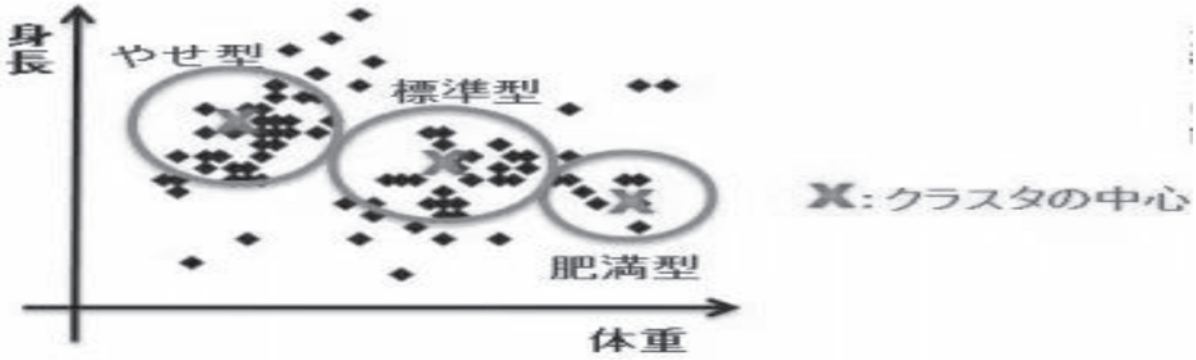
・「教師なし学習」の学習データは入力データのみで正解がない方法で、代表的なタスクとしては「クラスタリング」がある。

・クラスタリングは似た特徴を有するものを同じグループに分けるやり方である。たとえば身長や体重からデータをいくつかのグループに分類するような処理が該当する。

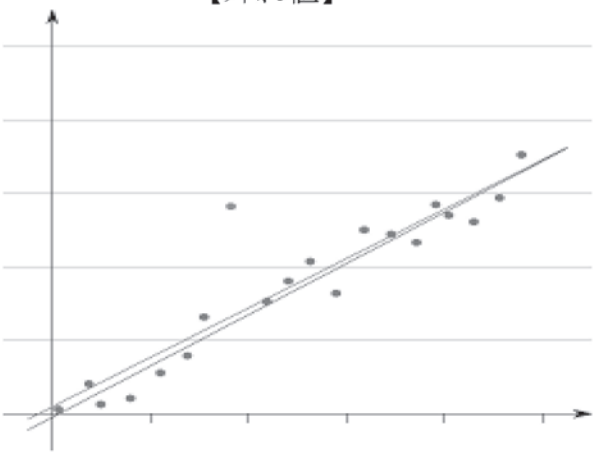
・外れ値が出る。その外れ値を処理しないと過学習となり判断を誤る。データを100%採用すると過学習となる。特に自然現象。

クラスタリング：対象データを値の類似性に着目して、複数のグループに分類する手法

データとデータの距離との直線距離を**ユークリッド距離**と呼ぶ。データの類似度は、この尺度を用い、距離の値が小さいほど類似度が高いとする。



【外れ値】



（３）強化学習

・「強化学習」も学習データとして用意するものは入力ｊだけで、出力の良し悪しに応じた報酬が与えられる。

・将来報酬が最大になるようにアルゴリズムやモデルを最適化する。ロボットの制御や「AlphaGo」のようなゲームに利用されている

・強化学習というのはつまり、いいことをしたら報酬を与えていくという学習方法で、どんどん効率的な方法を学んで賢くなっていく。

・最近ではロボット分野でも強化学習が行われている。実験ではロボットが穴の空いた部品に棒を通そうと試行錯誤をしているのだが、その様子がまるで何かを習得しようとしている子どものようである。

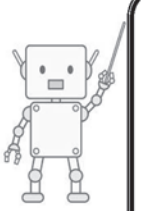
・つまり、学習対象のデータの正解がわからないが良さそうか悪そうかの報酬がわかる場合に、選択による報酬を手掛かりに学習する方法である。

◇統計学と機械学習の違い

数理的な背景はかなり似ているものがあるが、これらを実際にデータに使おうと思った時、「機械学習を使う」という場合と「統計学を使う」という場合では、次のような違いがある。

- 機械学習は予測に重きを置く
- 統計学はモデルの解釈に重きを置く

４　大人のＡＩと子供のＡＩ



ポイント

◇大人の AI
- **ビッグデータから一定のパターンを発見し、人間が判断基準を設定**
- **人間が裏で作り込みをし、特徴量の設計が必要となる。**

◇子供の AI
- **背景知識のなしで、AI 自身が試行錯誤を繰り返して性能を向上させる**
- **ディープラーニング**

AI 研究の松尾豊氏（東京大学准教授）により指摘されている、AI の２つの分け方である。

「大人の AI」は、すでに存在しているビッグデータから一定のパターンを発見するもので、発見にあたっての判断基準を人間が予め設定するタイプのもの。

特徴量の設計が必要で、ビッグデータや IoT に基づく判断、IBM のワトソン、Siri、Pepper などは、「大人の AI」に対応している。現状、販売やマーケティングなどに応用されており、今後は医療、金融、教育などへの応用が考えられている。

一方、「子供の AI」は、背景知識のほとんどない状態から、AI 自身が試行錯誤を繰り返して性能を向上させるタイプとのことである。

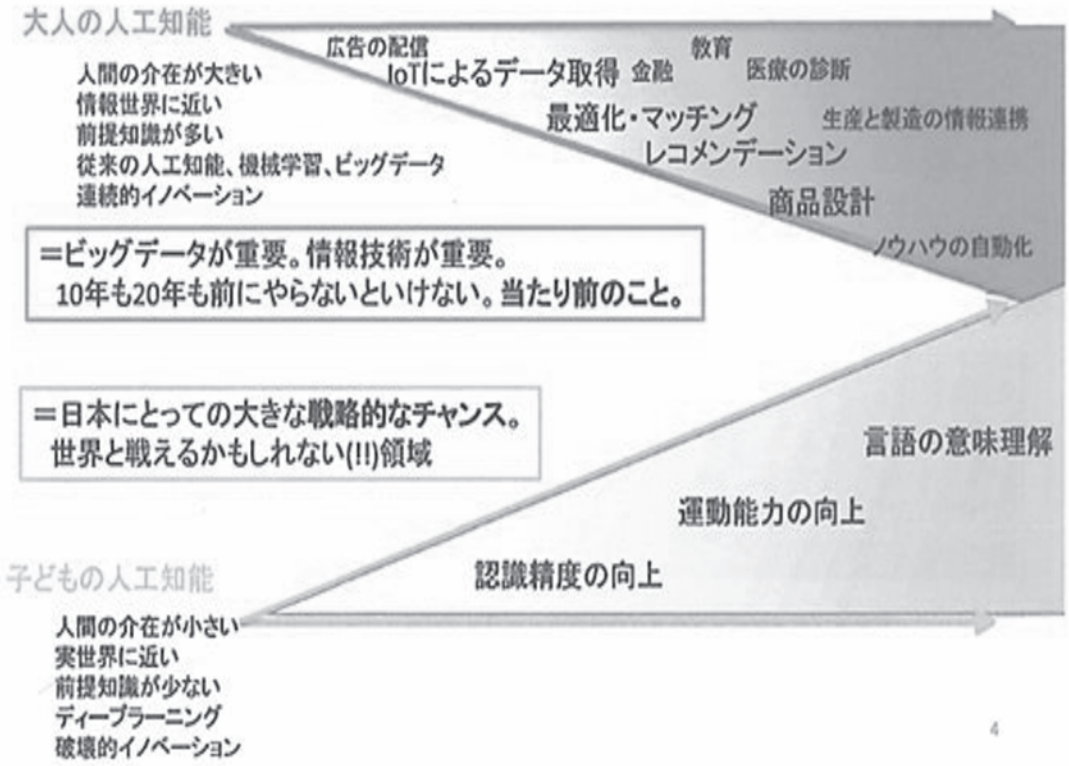
AI の研究の歴史において 50 年もの間、実現が難しいとされてきたが、ディープラーニング（深層学習）という、特徴量を自動で抽出できる技術の開発がブレークスルーとなって、今後への期待が高まってきている。

2015 年には画像の認識において人間の精度を超えるプログラムが米マイクロソフトや米グーグルで開発され、現実味を帯びてきた。

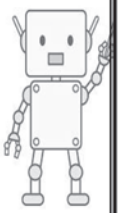
「子供の AI」は、認識技術の向上、運動能力の向上、言語の意味理解というように、人間と同じように技術進化していく。認識機能を持った「子供の AI」とロボットを組み合わせることにより、これまでにない新しいビジネスの創出ができると期待されている。

【松尾氏が提案する「子どもの人工知能に投資すべき」の概要】

	機械にとって容易	機械にとって困難
人にとって困難	データ分析・予測 意思決定支援 （大人の AI）	
人によって容易	従来までの 情報技術・人工知能の適応領域	パターン認識 身体制御 （子供の AI）


2つの人工知能:子どもの人工知能に投資すべき
大人の人工知能
人間の介在が大きい
情報世界に近い
前提知識が多い
従来の人工知能、機械学習、ビッグデータ
連続的イノベーション
広告の配信
IoTによるデータ取得
金融
教育
医療の診断
最適化・マッチング
生産と製造の情報連携
レコメンデーション
商品設計
ノウハウの自動化
=ビッグデータが重要。情報技術が重要。
10年も20年も前にやらないといけない。当たり前のこと。
=日本にとっての大きな戦略的なチャンス。
世界と戦えるかもしれない(!!)領域
言語の意味理解
運動能力の向上
認識精度の向上
子どもの人工知能
人間の介在が小さい
実世界に近い
前提知識が少ない
ディープラーニング
破壊的イノベーション
4

5　ディープラーニング（深層学習）



ポイント

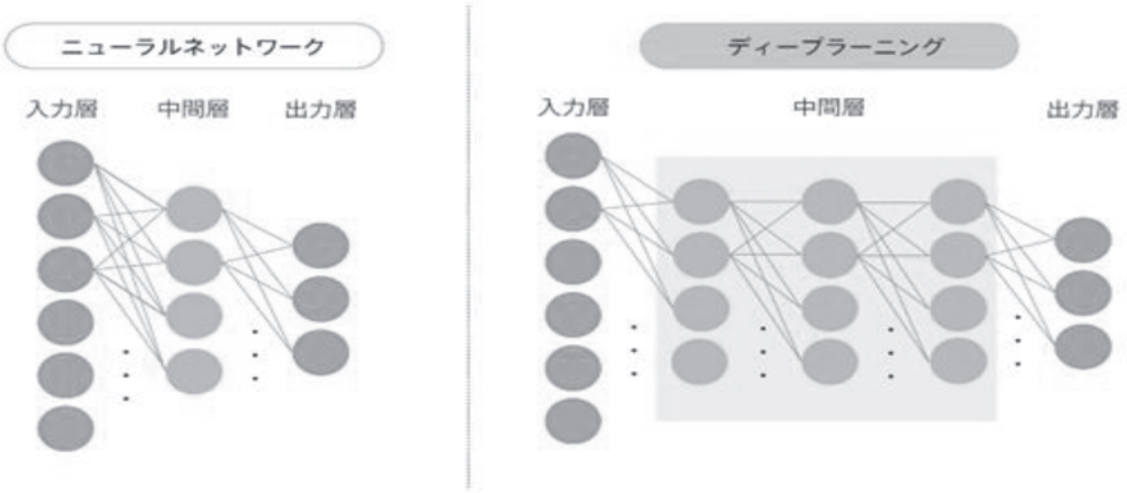
◇これまで人間が与えていたデータの特徴を AI 自身が見つけ出す
◇データから予測する過程で、多層ニューロンをつくる
◇多層ニューロンにより、データをベクトル化
◇それを何回も繰り返すことをディープラーニング（深層学習）という
◇人の脳細胞を模倣し、械学習を何度も繰り返し、精度を上げていく

（1）ディープラーニングの誕生

・近年の AI ブームは、2 つのテクノロジの研究が大幅に進んだことで起こった。それが「機械学習」と「ディープラーニング」である。
・機械学習に欠かせないものは、学習のために必要なデータである。近年、デジタル機器の急速な普及や通信技術の発達で大量のデータ・いわゆる「ビックデータ」が集まるようになった。
・これまでとは比べ物にならない量のデータ収集・解析することで、人工知能は活躍の場を広げるようになる。
・一方のディープラーニングとはこれまで人間が与えていたデータの特徴を AI 自身が見つけ出す仕組みを指す。これにより、AI は自ら新たな概念を理解したり、例外に対処できたりするようになった。
・またこれらのテクノロジを支える環境=高性能のコンピュータを気軽に利用出来るようになったことも、今の AI ブームを支える大きな要因である。

（2）ディープラーニングとは

・ディープラーニングとは、人間の脳が行っている知的な作業をコンピュータで模倣したソフトウェアやシステム。具体的には、人間の使う自然言語を理解したり、論理的な推論を行ったり、経験から学習したりするコンピュータプログラムなどのことをいう。
・人工知能を目指したシステムの開発は、コンピュータの高度な発達を背景に、学習、推論、認識、判断など、人間の脳の役割を機械に代替させようという研究分野、あるいはそのコンピュータシステムを云う。単純なデータの処理だけではなく、ビッグデータなど膨大な情報を繰り返し経験し、学習を続けていく仕組みのものである。
・ディープラーニングは中間層を多層にすることで情報伝達と処理を増やし、特徴量の精度や汎用性をあげたり、予測精度を向上させたりすることが可能になる。
・コンピュータがネコの画像をネコであると認識する場合、画像からなんらかの特徴を抽出し、あらかじめ記憶させたネコの基準となる特徴と照らし合わせる必要がある。
・ディープラーニングの登場以前には、研究者や技術者がネコの基準となる特徴をあらかじめ数量化した特徴量を設定したが、ディープラーニングにおいてはネコの属性をもつさまざまなものを大量に機械学習させることで、人が直接関与することなく、ネコの特徴をコンピュータが自動的に学んでいく。



（３）ディープラーニングの仕組

・ディープラーニング(深層学習とは、大量のデータを見てデータに含まれる「特徴」を段階的により深く（深層で）学習する機械学習である。

・ニューラルネットワークの構造を深く（狭義には４層以上の多層に）することで実現する。

・層の深いニューラルネットを特に深層ニューラルネットワークと云う。

・近年、多層ニューラルネットワークの学習の研究や、学習に必要な計算機の能力向上、および、Web の発達による訓練データ調達の容易化によって、十分学習させられるようになった。その結果、音声・画像・自然言語を対象とする問題に対し、他の手法を圧倒する高い性能を示し、2010 年代にさらに普及した。

・バックプロパゲーションにより推論結果が正しいか見つめ直して、アルゴリズムでどこが悪くどこを修正したらよかったかを学習する。

・複雑な問題を高い精度で解くことができ、画像認識、文字認識、音声認識などで成果が出ている。深層学習が人間以上の能力を発揮するようになってきたことから、深層学習あるいは機械学習を AI ということもある。

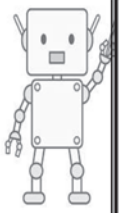
・深層学習は、教師あり学習、教師なし学習に加えて、強化学習、そして転移学習の進展が注目されている。

（４）ディープラーニングの分野

深層学習の本質は、特徴学習あるいは表現学習とされている。画像や音声、自然言語などの特徴量と呼ばれる数値を、抽出し学習する方法で、対象ごとに以下の分野に分かれる。

・画像認識、映像認識　・文字認識、音声認識　・自然言語処理および自然言語理解

6　ＡＩ要素技術



ポイント

◇人工知能とは
・人間が知能を用いて行う働きを機械に行わせようとする試み
◇人工知能はさまざまな要素技術の集合体である
◇AI 要素技術例
・音声認識システムとクラウド技術が協調することで携帯端末からの音声検索機能が実現
・GPS や地図情報システムなどと連係した高度なナビゲーションシステム

（１）AI 要素技術例：自然言語処理

自然言語処理とは、日常的に使っている「ことば」をコンピュータに処理させる一連の技術のことで、人間がお互いにコミュニケーションを行うために自然発生した。

そのため、プログラミング言語と違い、曖昧さを含む。また文化的な背景、時代や地域によって変化するため、ルール化が難しく扱いづらいという特性がある。

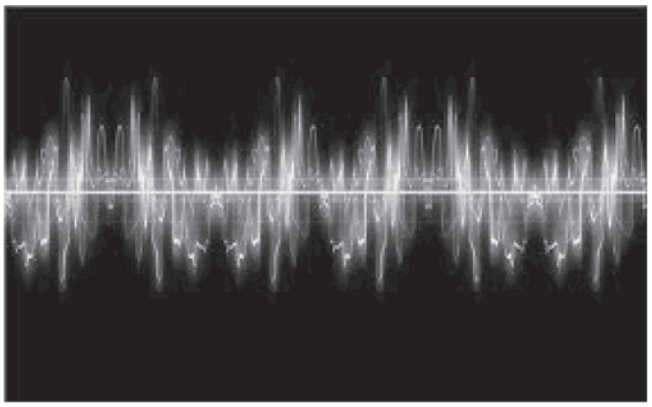
自然言語処理は、「形態素解析→構文解析→意味解析→文脈解析」の順で行われる。

形態素解析と構文解析は、コンピュータにとって比較的行いやすい作業だが、意味解析や文脈解析を行うには、大量のデータでコンピュータに学習させることが必要であり、「機械学習」が必要となる。

自然言語処理が使われている例は、日本語入力（かな漢字文字変換）、機械翻訳、対話システムなどがある。

（２）自然言語処理例：音声認識

・音声認識は、人間の会話の音声を認識して、それをテキストに変換する技術の事である。
・Siri に向かって何か話しかけると、それが認識されてテキストに変換されている様子が確認できる。それが、まさに音声認識である。
・一般にコンピュータにより音声からテキストを生成する技術であり、キーボードなどに代わる入力手段で、音声をコンピュータの扱いやすいデータに変換する。
・発生された音声は、見えないが音波として表現される。



・音声認識技術では、この音波から、音の最小構成単位である「音素」を特定し、それを手がかりにしてテキストに変換する技術である。
・「音素」は、「母音　アイウエオ」、「 撥音　ン」、「 子音　23 種類」の単位から構成される。
・声認識技術は、その音素を認識して、テキストに変換していく技術で、その後、音素の繋がりからパターンマッチング（辞書）の探索が行われ、テキストに変換される。

（３）画像認識

・画像認識とは、画像や動画から特徴をつかみ、対象物を識別するパターン認識技術の１つである。
・人間は、画像に写っているものが何であるか、これまでの経験から「理解」して判断することができる。しかし、コンピュータは画像に何が写っているかを「理解」することができない。その代わりに、膨大な画像データから、対象物の特徴を学習させることで、未知の画像を与えた時に、対象物が何であるかを「確率」として表現することができるようになる。
・画像認識の歴史は古く、1960 年頃から研究されてきた。昔はコンピュータの性能が低く、価格も高価であったため、大学の研究機関などの限られた分野での利用が主だったが、現在ではパーソナルコンピュータはもちろん、デジタルカメラやスマートフォンなど、多くの電子機器に画像認識機能が取り入れられている。
・人間は画像から対象物を視覚として判断できるが、コンピュータにとっては、画像はピクセルごとの情報（色、明るさ）の集合体でしかない。
・コンピュータに対象物を認識させるためには、おびただしい量の画像データとラベル（画像データが何を表すか）を与え、対象物を学習させる必要がある。

open-cv ホームページより

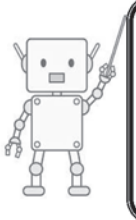
（4）ナチュラルインターフェイス

ナチュラルインターフェイスとは、タッチパネルなど人間が直感的に操作できる方法や、音声、表情、身振り手振りなど人間同士が意思疎通に用いる手段にコンピュータを介在させる技術である。

現在では、スマートフォンのタッチパネルがナチュラルインターフェイスに相当する。音声認識や画像認識、自然言語処理などの技術を活用したインターフェイスで、相手に合わせて柔軟に対応できる特性を持つ。「対話処理」や「マルチモーダル」などがモデル化されている。

Microsoft HoloLens より

7　ＡＩの情報処理方法

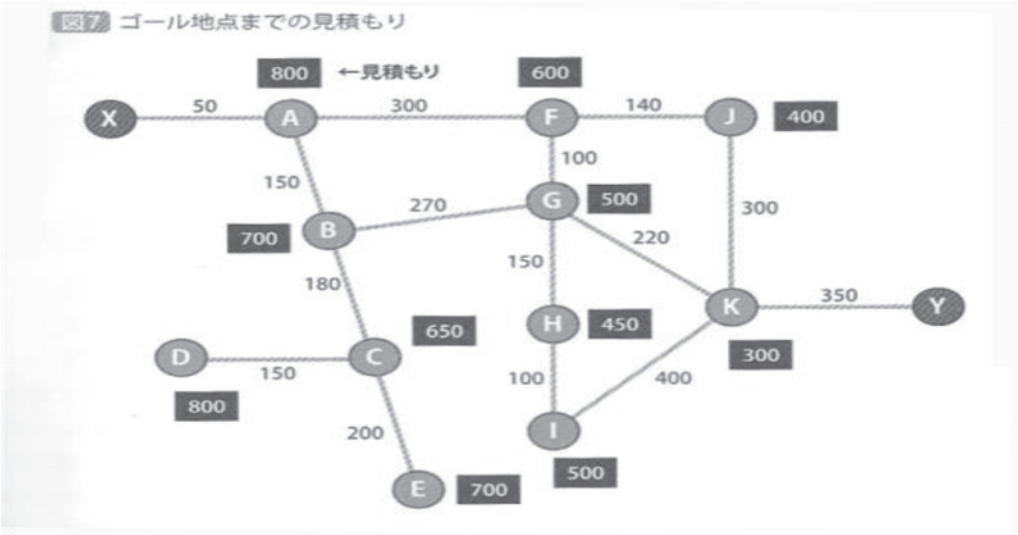


ポイント

◇人間は経験等に基づき、直観を使い、良いと思うものを選択する
◇コンピュータは、可能性のある全部の件数を探索し
　その中から確率の高いものを選択する

◇ある地点から別の地点を探索するときに用いられる経路探索の例である
　Ｘ（スタート）からＡに行く。

① Ａから行くことのできると地点はＢとＦ。先ずはＢに方向行く。
② Ｂから行くことのできる地点はＣとＧ。先ずはＣに行く。
③ Ｃから行くことのできる地点はＤとＥ。先ずはＤに行く。
④ Ｄから行くノードはないので、Ｃ－Ｄのルートを排除する。
⑤ Ｃから行くことのできる地点はＥなのでＥに行く。
⑥ Ｅから行くノードはないので、－Ｅのルートを排除する。
⑦ 最初から探索し直しで、Ｘ－Ａ－Ｂへ行く。
⑧ Ｂから行くことのできる地点はＧだけなので、Ｇに行く。
⑨ Ｇから行くことのできる地点はＨとＫ。先ずはＨに行く。
⑩ Ｈから行くことのできる地点はＩなのでＩに行く。
⑪ Ｉから行くことのできる地点はＫなのでＫに行く。
⑫ Ｋから行くことのできる地点はＹなのでＹに行く。
⑬Ｙに到達し１本目のルートが見つかる



次にあげる処理は、上記の経路探索と同じであるが、単純な逐次処理だけでなく、各地点の「見積もり距離」という知的情報を導入し、最適解を高速に見つけるという新たな情報の観点を具体化したものである。(Aスター法)

① X（スタート）からAに行く。予想コストは850となる。

② A から行くことのできると地点は B と F。B に行った際の予想コストは（900）で、Fは（　950　）。よって予想コストが低いBに行く。

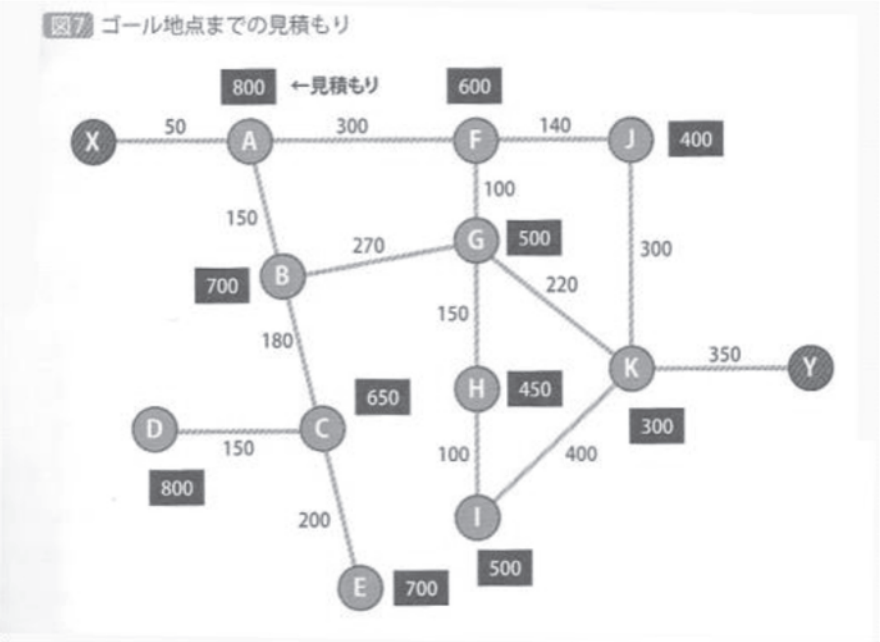
③ Bから行くことのできる地点はCとG。Cに行った際の予想コストは1030で、Gのコストは（　970　）。現時点で一番安い予想コストが一番低いのは②で検討したFなのでFに行く。

④ Fから行くことのできる地点はGとJ。Gに行った際の予想コストは(　950)で、Jのコストは890。現時点で一番安い予想コストが一番低いのはJなのでJに行く。

⑤ Jから行くことのできる地点はK。Kに行った際の予想コストは1090。現時点で一番安い予想コストが一番低いのは④で検討したGなのでGに行く。

⑥ Gから行くことのできる地点はHとK。Hに行った際の予想コストは(1050)で、Kの予想コストは（　970　）。現時点で予想コストが一番安いのはKなので、Kに行く。

⑦ Kから行くことのできる地点はIとY。Iに行った際の予想コストは1570で、Yは1020。よってYへ行く（ゴール）



◇プログラム例

```
def get_path(start, goal, prev):
    path = []
    now = goal
    path.append(now)
    while True:
        path.append(prev[now])
        if prev[now] == start: break
        now = prev[now]
    path.reverse()
    return path

def search(graph, start, goal):
    MAX_VAL = 0x10000000
    g_size = len(graph)
    visited = [False] * g_size
    cost = [MAX_VAL] * g_size
    es = [MAX_VAL] * g_size
    prev = [None] * g_size
    cost[start] = 0
    prev[start] = start
    now = start
    while True:
        min = MAX_VAL
        emin = MAX_VAL
        next = -1
        visited[now] = True
        for i in range(g_size):
            if visited[i]: continue
            if graph[now][i]:
                tmp_cost = graph[now][i] + cost[now]
                if cost[i] > tmp_cost:
                    cost[i] = tmp_cost
                    prev[i] = now
                    es[i]=cost[i]+estimate[i]
            if min > cost[i]:
```

```
                min = cost[i]
            if emin > es[i]:
                emin = es[i]
                next = i
        if next == goal: break
        now = next


    print(prev,cost[goal])
    return [get_path(start, goal, prev), cost[goal]]

g = [#0   1   2   3   4   5   6   7   8   9  10  11  12
    [ 0, 50,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],   # X (0)
    [ 0,  0,150,  0,  0,  0,300,  0,  0,  0,  0,  0,  0],   # A (1)
    [ 0,  0,  0,180,  0,  0,  0,270,  0,  0,  0,  0,  0],   # B (2)
    [ 0,  0,  0,  0,150,200,  0,  0,  0,  0,  0,  0,  0],   # C (3)
    [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],   # D (4)
    [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],   # E (5)
    [ 0,  0,  0,  0,  0,  0,  0,100,  0,  0,140,  0,  0],   # F (6)
    [ 0,  0,  0,  0,  0,  0,  0,  0,150,  0,  0,220,  0],   # G (7)
    [ 0,  0,  0,  0,  0,  0,  0,  0,  0,100,  0,  0,  0],   # H (8)
    [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,400,  0],   # I (9)
    [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,300,  0],   # J (10)
    [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,350],   # K (11)
    [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0]    # Y (12)
]
estimate=[0,800,700,650,800,700,600,500,450,500,400,300,0]
path, cost = search(g, 0, 12)
print('path:',path, 'cost: ', cost)
```

◇2 次のプログラムは「分類」の基本となるアヤメの判定プログラムである。

```
# -*- coding: utf-8 -*-
from sklearn import datasets, svm
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import pandas as pd
# アヤメのデータをロードし、変数 iris に格納
iris = datasets.load_iris()

# 特徴量のセットを変数 X に、ターゲットを変数 y に格納
X = iris.data
y = iris.target
# 特徴量を外花被片の長さ(sepal length)と幅(sepal width)の
# 2 つのみに制限(2 次元で考えるため)
df = pd.DataFrame(data=X, columns=['a', 'b', 'c','d'])
X = X[:,0:2]

# 分類用にサポートベクトルマシンを用意
#clf = svm.SVC(C=1.0, kernel='linear', decision_function_shape='ovr')
# 'auto'を指定すると 1/(次元)がセットされる。この場合、1/2=0.5
clf = svm.SVC(C=1.0, kernel='rbf', gamma='auto',
decision_function_shape='ovr')
# gamma を大きくすると、曲率が大きい（よく曲がる）境界となる
#clf = svm.SVC(C=1.0, kernel='rbf', gamma=1.0,
decision_function_shape='ovr')
# データに最適化
clf.fit(X, y)

##### 分類結果を背景の色分けにより表示

# 外花被片の長さ(sepal length)と幅(sepal width)の
# 最小値と最大値からそれぞれ 1 ずつ広げた領域を
# グラフ表示エリアとする
x_min = min(X[:,0]) - 1
```

```
x_max = max(X[:,0]) + 1
y_min = min(X[:,1]) - 1
y_max = max(X[:,1]) + 1

# グラフ表示エリアを縦横500ずつのグリッドに区切る
# (分類クラスに応じて背景に色を塗るため)
XX, YY = np.mgrid[x_min:x_max:500j, y_min:y_max:500j]

# グリッドの点をscikit-learn用の入力に並べなおす
Xg = np.c_[XX.ravel(), YY.ravel()]

# 各グリッドの点が属するクラス(0〜2)の予測をZに格納
Z = clf.predict(Xg)

# グリッド上に並べなおす
Z = Z.reshape(XX.shape)

# クラス0 (iris setosa) が薄オレンジ (1, 0.93, 0.5, 1)
# クラス1 (iris versicolor) が薄青 (0.5, 1, 1, 1)
# クラス2 (iris virginica) が薄緑 (0.5, 0.75, 0.5, 1)
cmap0 = ListedColormap([(0, 0, 0, 0), (1, 0.93, 0.5, 1)])
cmap1 = ListedColormap([(0, 0, 0, 0), (0.5, 1, 1, 1)])
cmap2 = ListedColormap([(0, 0, 0, 0), (0.5, 0.75, 0.5, 1)])

# 背景の色を表示
plt.pcolormesh(XX, YY, Z==0, cmap=cmap0)
plt.pcolormesh(XX, YY, Z==1, cmap=cmap1)
plt.pcolormesh(XX, YY, Z==2, cmap=cmap2)

# 軸ラベルを設定
plt.xlabel('sepal length')
plt.ylabel('sepal width')

##### ターゲットに応じた色付きでデータ点を表示

# iris setosa (y=0) のデータのみを取り出す
```

```
Xc0 = X[y==0]
# iris versicolor (y=1) のデータのみを取り出す
Xc1 = X[y==1]
# iris virginica (y=2) のデータのみを取り出す
Xc2 = X[y==2]

# iris setosa のデータ Xc0 をプロット
plt.scatter(Xc0[:,0],     Xc0[:,1],     c='#E69F00',     linewidths=0.5,
edgecolors='black')
# iris versicolor のデータ Xc1 をプロット
plt.scatter(Xc1[:,0],     Xc1[:,1],     c='#56B4E9',     linewidths=0.5,
edgecolors='black')
# iris virginica のデータ Xc2 をプロット
plt.scatter(Xc2[:,0],     Xc2[:,1],     c='#008000',     linewidths=0.5,
edgecolors='black')

# サポートベクトルを取得
SV = clf.support_vectors_
# サポートベクトルの点に対し、赤い枠線を表示
plt.scatter(SV[:,0],     SV[:,1],     c=(0,0,0,0),     linewidths=1.0,
edgecolors='red')

# 描画したグラフを表示
plt.show()
```

参考資料

1. AI プログラミング基礎(Python)

「AIプログラム」の開発に向けたコンピュータでの開発技術を想定していますが、本章では、開発言語である「Python」の基礎・入門から始めます。

1-1. 基本操作と変数・演算

1-1-1. プログラムが動く仕組み

コンピュータは人間の言葉を理解できず、0 と 1 で記述された機械語といわれるプログラム(命令)の通りに動きます。逆に人間は「0 と 1 で書かれた機械の命令」は理解することができません。

プログラミング言語はこの「人間の言葉」と「01 で書かれた機械の命令」の中間に存在しています。人間は「規則性を持ったプログラミング言語」で命令を書き、それをソフトウェアによって「0 と 1」に変換してコンピュータに実行させます。

- 機械の言語と人間の言語を結びつけるのがプログラミング言語
- 機械の言葉とは「0 と 1」だけでなる

私達人間は言語でものごとを考え、お互いの意思を会話や仕草で伝えます。程度は違っても動物もこれと同じです。これらの会話や頭の中での思考に使われる言語は「自然言語」と呼ばれており、日本語や英語などが含まれます。

一方、コンピュータは全く異なり、内部の電気信号で命令を実行し、データのやりとりも電気信号を使います。そのため人間の言葉といった曖昧なものではなく「電気の ON/OFF」という単純な組み合わせることで動いています。

電気の「ON と OFF」という 2 つの状態は、数字の「1 と 0」に対応付けることができます。そのため、機械に命令を伝えるときは「ON OFF ON OFF」と書くのではなく「1010」と表現します。機械ができる言葉である 01 の命令は「マシン語(機械語)」と呼ばれています。

1-1-2. プログラムの実行

Python に処理を実行させるにはプログラムをプログラムファイルに書いてそのファイルインタプリタに指定して実行するという方式をとります。

それで、プログラム開発をするには「プログラミング用のエディタ」でコードを書く必要があります。ファイルに書かれたプログラムはコンソールから python コマンドを使って実行します。

開発の流れ

Python のインタプリタでは、「ユーザーからの入力」と「Python の処理と出力」を繰り返します。

Python に決められた処理を実行させたい場合は「ファイルに書かれた Python のプログラムを Python に実行させる」という手法を使います。プログラマは実行したい処理をあらかじめファイルに書いておき、それを Python に実行させるという流れで行います。

Python でのプログラムの開発にはプログラミングに適した「IDE(開発環境)のエディタ」を使います。Windows のノートパッドで開発することもできますが、プログラミング用のエディタには「シンタックスハイライト」と呼ばれる文法を解釈して文字を色付けする機能や、「オートインデント」と呼ばれる整形機能があり、また高機能な IDE エディタでは「補完」と呼ばれる入力候補を示してくれる機能すらあります。

IDLE での開発

Python をインストールすると「IDLE」と呼ばれる開発環境も同時にインストールされています。プログラミングを全くしたことがない初心者はこれを使うのが簡単ですので、IDLE を使った開発方法をまずは紹介します。

次は「IDLE」と呼ばれる Python の開発環境を使ってコードを書いてみます。IDLE は先ほどのようなプロンプト画面も使えますし、Python を書くのに適したエディタの機能も持っています。では、さっそく IDLE を立ち上げてみましょう。

インタプリタ機能の利用

Windows は、Windows メニューにある「すべてのプログラム」から「Python」を選択し、その下にある「IDLE(Python GUI)」を選べば起動します。

プログラムの作成

IDLE はインタプリタだけでなく、Python の開発用のエディタとしても使うことができます。新規ファイルの画面を開くには、IDLE のメニューから[File]->[New File]とします。

これに以下の文を書き込んで「test.py」というファイル名で保存([File]->[Save])してください。ファイルの拡張子(.txt や.mp4 など)は、そのファイルがどの種類のものかを表します。Python のプログラムファイルは拡張子が「.py」となります。

```
print('hello')
```

このプログラムの中にある「print」はカッコのなかにあるデータを画面に表示しなさいという命令になります。

プログラムの実行

IDLE のエディタが選択されている状態で「実行」メニューを押すと、このファイルが Python で実行され、結果がインタプリタ側の画面に表示されます。
また、プログラムを実行するにはファイルが保存されている必要があります。編集中のファイルを実行しようとすると、その前に保存することを IDLE が要求してきます。

コンピュータの動作

AI プログラミングを行う上でコンピュータの動作はつまるところ

① コンピュータ内のメインメモリに膨大なデータ(センサーなどから環境からの認知情報やそれらを再活用するために保存していた外部記憶データをから再度取り込んだ情報)を、人間の能力が及ばないほど高速で処理し、判断基準となる質的・最適な再編を行うこと
② 前期の判断に基づいて外部の機器に動作・制御などの支持を高速行ったり外部のネットなどに通信したりすること

の 2 点に尽きます。

これらの目的を完遂させるため、コンピュータにとって処理しやすい(2 進数的に)ようにデータを加工、フォーマット化させる必要があります。

そのため、次節以降で説明するデータの型や変数といったプログラム上の規約の概念が存在します。

1-1-3. データと型

プログラミングには様々な種類の「データ」が登場します。たとえば計算をするための「数値」であったり、テキストを扱うための「文字列(テキスト)」などがあったりします。

プログラミング言語ではこの「データの種類」のことを「型」と呼んでいます。「型」と「型でできる処理」は密接に関係していて、たとえば数値であれば足し算や引き算ができます。

一方、文字列であれば文字列の操作など独特な処理ができます。

型とはなにか

プログラミングでは「数字」や「テキスト」といった様々な種類のデータが使われます。それらのデータの種類は「型」と呼ばれています。

たとえば、数字の 5 や 10 は「整数」という型です。一方、'Hello'や'Python'といったテキスト(文字列と呼ばれる)は「文字列」という型です。

型と密接に結びつく処理

型とそれらの処理は密接に結びついています。例えば数値は「10×5」といったようにかけ算が使えますが、文字列は「"Hello"×"Python"」などというようにはできません。

初心者がプログラミングを学ぶには「どのような型があり、それが何をできるのか」をまず覚える必要があります。

そ れぞれの型の細かい使い方は型ごとの解説ページで行います。ここでは初心者向けに実例を使って、型とそれができる処理が結びついていることを説明します。

Python のプロンプトを立ち上げて以下を実行してみてください。

インタプリタ:数値の足し算と引き算

```
>>> 3 + 5
8
>>> 3 - 1
2
```

上記のように数値は足し算、引き算することができます。

では、文字列はどうでしょうか。 文字列はシングルクオテーション「'」でアルファベットや記号を囲むことで作成できます。

インタプリタ: 文字列の足し算(結合)

```
>>> 'hello' + ' python'
'hello python'
```

文字列の後ろに別の文字列をくっつけるという「結合」処理がされています。

型の変換(キャスト)

ある型を別の型に変換することを「キャスト」といいます。たとえば数値型から文字列型への変換は以下のように行います。

```
>>> str(123)
'123'
```

先ほど type 関数で「文字列」に対応していた「str」を、関数として使っています。この関数に与えた数値型の「123」は文字列型の「'123'」となっています。

これとは逆に文字列を数値型(整数型)にキャストすることもできます。キャストには数値型を示す「int」を関数として使っています。

```
>>> int('123')
123
```

文字列型や整数型だけでなく、他の主要な型の多くはキャストして変換することができます。

文字列へのキャスト

文字列型へのキャストは他の型のキャストに比べてよく利用されます。

先ほどプログラムファイルを実行する際に print 関数を使ってコンソールへ出力をしました。print 関数は文字列以外に数値なども出力させることができますが、その出力するテキストを作る際にキャストを多用します。

プログラムの処理をコンソールに出力したりするために文字列を作ることはよくあります。 キャストを駆使して文字列を作る以外にも、文字列のフォーマットといったテクニックもあります。

1-1-4. 変数

プログラム開発で使う「変数」は様々なデータを保存したり取り出したりすることができる箱のようなもので、一時的にデータを保管します。変数を使うことで複数行に及ぶ複雑なプログラムで継続して使うといったことができます。

変数の値は参照されても変化しません。たとえば変数 abc に数値 123 が代入されているとすると、それを「abc + 456」としても「print(abc)」としても、変数 abc の中身は 123 です。ただ、その変数に値を再代入すると中身は上書きされます。

代入と演算を同時に行う「複合代入演算」と呼ばれる処理もあります。

変数

1. 変数とは箱のように「値の出し入れができるもの」
2. 変数で出し入れする「値」とは「数値」や「文字列
3. 変数は自由に名前を付けることができる
4. 変数を使うためには宣言が必要

変数の目的と利用方法

現実のプログラムは、1 行のプログラムで全ての処理を作ることはできません。そのため、複数の行で少しずつプログラムを組み立てていくという作業が必要です。

ですが、それよりも基本的な「前の行で処理した結果を、後の行で使う」ことが一番多いです。

「変数」は処理結果を格納したり、取り出したりするために使われる仕組みです。前の行でなんらかの処理をして結果を求め、それを変数に格納する。

そして後ろの行で変数から値を取り出して利用することで、複数行にまたがる処理を書くことができます。

変数には何回でも値を格納することができますが、すでに変数が値を持っている場合はそれが上書きされます。

これには関数の作成や条件分岐、ループといった以後のページで学ぶ制御用の文法も使います。

変数から値を取り出すことも何度もできます。値を取り出しても変数の中身は残ったままとなりますので、変数が上書きされていない限りは毎回同じ値を得ます。

変数の宣言と代入

実際に Python で変数を使いながら、その利用方法と特徴を確認します。

変数を使うには、最初に変数が作成される必要があります。変数を作成することを「変数を宣言する」といいます。

Python は変数を宣言する際に、同時に変数に値を格納する必要があります。変数に値を格納することを「代入」といい、宣言された変数に最初に代入することを特に「初期化」と呼びます。

変数の宣言と初期化(代入)は以下のように「=」記号(「代入演算子」と呼ばれる)の左に変数名を書き、右側に代入する値を書きます。

変数名 = 変数に入れたい値

注意をしてほしいのはプログラミングにおける代入演算子「=」は、算数や数学のイコール記号「=」とは異なることです。算数や数学のイコール記号は「=」の左辺(左側)と右辺(右側)が同じことを示していますが、プログラミングにおける代入演算子は右辺を左辺に代入することに使われます。

たとえば変数 abc を宣言し、数値の 1234 で初期化するとすれば以下のようになります。

```
>>> abc = 123
```

変数を今まで扱ってきたデータのように使えば、代入されている変数の中身の値は取り出されます。

インタプリタ:変数からの値の取り出し(直前のプログラムの続き)

```
>>> 10 + abc
133
```

```
>>> print(abc)
123
```

すでに宣言されている変数に値をもういちど代入する(「再代入」と呼ばれる)することもできます。プログラミング言語によりやりかたは異なりますが、Python では初期化と見た目は全く変わりません。

インタプリタ: 変数への再代入

```
>>> cd = 10
>>> print(cd)
10
>>> cd = 20
>>> print(cd)
20
```

「変数名」はアルファベットや数字、一部の記号を使って自分の好きなものを使うことができます。変数名の命名規則は次項で説明します

代入演算子の優先度

1 行の中に様々な演算子が混ざることがあります。たとえば以下のプログラムを見て下さい。

```
>>> ef = 10 + 5
```

このプログラムは「ef = 10」をした結果に対して「+ 5」をするのでしょうか。それとも「10 + 5」をした結果を「ef = 」とするのでしょうか。

答えは後者で、変数 ef に対して 15 が代入されています。

これには理由があり、結合演算子「+」のほうが、代入演算子「=」よりも優先度が高いためです。算数で「5 + 10 x 2」が、30 ではなく 25 になるのは足し算よりも掛け算の優先度が高いからです。

それと同じでプログラミングの演算子にも優先度があり、代入演算子の優先度はかなり低いです。

代入演算子の右側でなんらかの処理が書かれていれば、その処理がされた結果が変数に代入されます。変数に代入されてから代入演算子の右側の処理がされることはありません。

変数に同じ変数の値を加工して代入

変数に同じ変数の値を加工して代入することも可能です。たとえば、変数 a にすでに文字列が入っており、それに別の文字列を追加したいという場合は以下のように書きます。

インタプリタ: 変数内の文字列を変更

```
>>> a = 'hello'
>>> a = a + ' python'
>>> print(a)
hello python
```

上記の「a = a + ' python'」は以下の動きをします。

代入演算子の右にある a が'hello'という文字列を返す
それに' python'が結合されて'hello python'になる
代入演算子の右の結果である'hello python'が左の変数 a に再代入される

こういった処理は上から下に順番に実行していくような状況ではあまり利用されませんが、後ほど扱うループ処理(同じ処理を繰り返す)の中で利用されることがよくあります。

複合代入演算子

先ほどの「変数 a = 変数 a + 値 b」という処理をするための特別な代入演算子があります。「複合代入演算子」と呼ばれるもので「演算子=」という形式を取ります。

たとえば演算子「+」を使った複合代入演算子として「+=」があり、「変数 A += 値 B」とすると、変数 A の値に値 B をたした結果を変数 A に代入します。

他には「-=」や「*=」などもありますが、それぞれ「引いて代入」「かけて代入」となります。

```
>>> a = 'hello'
>>> a += ' python'
>>> print(a)
```

```
hello python
```

複合代入演算子は変数の中身を利用しますので、その変数は事前に初期化されている必要があります。初期化されていない変数で複合代入演算子を使うとエラーになります。

```
>>> b += 'python'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'b' is not defined
```

変数名の付け方

変数名はその変数にどのようなデータが入っているかを示しています。そのため、正しい変数名を付けるとプログラムの中で何を意味しているか分かりやすくなります。

変数名のつけかたには Python の文法上のルールを守った上で、分かりやすいものを付けるのが鉄則です。変数名は「アルファベット、数字、アンダーバー」のみで構成されており、一般的には英単語をアンダーバー「_」区切りで使います。

短いサンプルプログラムなどを除けば、変数名に「a」などといった用途が分からないものを使うことは避けて下さい。

Python の文法が利用する「if」といった予約語や、「str」「print」といった既に定義されているキーワードを変数として使うとプログラムがエラーになったり、関数が本来の動きと異なる動きをしたりするようになります。

変数名の重要性

変数名のルール

変数名の名付け方にはいくつかの文法上の規則があります。 これらの規則を破った変数名を使うとプログラムでエラーが発生します。

規則は以下のものとなります。

- 予約語でないこと
- アルファベット(大文字、小文字)か、アンダーバーから始まること
- アルファベット、数字、アンダーバーから構成されていること

予約語は変数名にできない

「予約語」は Python の文法において特別な意味を持つキーワードのことです。たとえば条件分岐に利用する「if」「else」といったものです。

これらを変数名として使おうとすると、予約語が間違った使い方をされていると Python が解釈してしまうため、プログラムの実行の前の文法チェックでエラーが発生してしまいます。

```
>>> name = 'taro'
>>> if = 'taro'
  File "<stdin>", line 1
    if = 'taro'
        ^SyntaxError: invalid syntax
```

また、予約語ではありませんが既に定義されている関数名などを変数名として使うこともしないでください。 たとえば関数「print」を変数名として使うこともできますが、それをやると関数の print が使えなくなります。

変数名の付け方

サンプルプログラムでは、変数名に「a」や「abc」といった意味のない名前をつけることがよくあります。これは Python の文法という観点では間違っていませんが、プログラムの見通しが悪くなる変数名です。

実際のプログラミングを行う際は、先の文法ルール以外にも、以下のルールを守って変数名をつけることが推奨されています。

- 分かりやすい変数名にする
- 名詞を使う
- 大文字は使わずに、アンダーバーで単語を区切る(python 流)
- 理由なくアンダーバーから始まる変数名は使わない(特別な意味を持つ)

主要な型と演算子

Python に限らずプログラミング初心者がプログラミングを書けるようになるには「使用頻度の高い重要な型を覚える」ことが大事です。重要な型はどのような処理を作る際にも、プログラムの「つなぎ」として使われます。

つまりそれらなしにはどのようなプログラムを組むこともできないということです。

Python で一般的に使われる型は 10 以上ありますが、そのなかでも特に大事なのは以下の 4 つです。

- 数値型(整数、長整数、浮動小数点、虚数など)
- 文字列型
- Bool 型
- リスト型

ここでは、それほど難しくない簡単だけれども重要な項目のみに絞って説明します。

数値型

Python の整数型と小数型

Python で数値を扱うのは簡単です。特に難しいことを考えなくてもおおよそ思い通りに動きます。

実は今まで数を扱う「数値型」と説明してきましたが、厳密にはそのようなものはありません。あるのは整数を扱う「整数型」と、浮動小数点を扱う「浮動小数点型」です。整数型と浮動小数点型は、英語ではそれぞれ「int」と「float」となります。

```
>>> type(123)
<class 'int'>
>>> type(0.456)
<class 'float'>
```

Python は整数と浮動小数点の区別なく使うことができ、変数に浮動小数点を代入してもそれが勝手に整数に丸められることはありません。

同様に整数の上限も特に決まっていないため、非常に大きな数の計算も桁を考えずに実行できます。これが今までの説明で整数型や浮動小数点型といわず、数値型としてきた理由です。

```
>>> 123 + 0.456
123.456
>>> 1234567891234567891234567 89 + 1234567891234567891234567 89
2469135782469135782469135 78
```

計算に使う演算子

数値型(整数型と浮動小数点型)でどのような処理ができるかという話に移りましょう。数と数の計算に使用する「+」や「-」といった記号は「演算子」と呼びます。

変数への代入に利用する「=」はこの演算子の一つで代入演算子と呼ばれています。ただ、計算ではなく代入処理に使うため、「+」や「-」とは少し異なります。

演算子の演算対象となる値は「オペランド」と呼ばれます。たとえば、計算式「1 + 2」の演算子は「+」であり、そのオペランドは「1」と「2」です。

```
>>> 1 + 2
3
```

Python の数値計算で使う初歩的な演算子は以下となります。

「+」や「-」は分かるかと思いますが、掛け算は算数や数学で使う記号ではなく「*」を使います。割り算は分数の「/」だと思えば分かりやすいでしょう。

演算子の優先度とその制御

足し算や掛け算には優先順位があります。たとえば「1 + 2 x 3」という計算をする場合、足し算よりも掛け算が優先されるため、「1 + 2」より先に「2 x 3」が計算されて、答えは 7 になります。

Python でも掛け算が足し算よりも優先されるため、「1 + 2 x 3」の計算結果は 7 となります。

```
>>> 1 + 2 * 3
```

7

算数で掛け算よりも足し算を優先する場合は「足し算を()で囲む」ことをします。 Python でも同じことができます。

```
>>> (1 + 2) * 3
9
```

演算子の優先度は掛け算と割り算の優先度が高く、足し算と引き算の優先度がそれより低くなっています。そして代入演算子の優先度は低いです。

演算子の優先度が分かりにくい場合は「()」で囲むことで明示的に示すようにしてください。「()」を使わなくても正しい場合であっても、結果が複雑な優先度によって求められる場合は「()」を使うほうが分かりやすいです。

複合代入演算子

変数のページで扱った「 複合代入演算子」は、演算子の左側の変数に「その変数自身と演算子の右側の演算結果」を代入するのでした。この複合代入演算子の動きは、数値計算の演算子と対応しています。

たとえば「+=」と「+」は対応しており、「*=」と「*」が対応しています。オペランドとなるデータの型にある演算子が使えれば、その複合代入演算子も使えます。

|な複合代入演算 |
説明 |

a+=b	a=a+b
a-=b	a=a-b
a*=b	a=a*b
a/=b	a=a/b

```
>>> a = 5
>>> a += 3
>>> print(a)
8
```

浮動小数点

Python の浮動小数点は整数の扱いとほぼ同じです。 整数が使える演算子や代入演算子を使うことができ、演算子の優先度も同じです。

整数の割り算などでも、必要があれば勝手に浮動小数点となります。

```
>>> 5 / 2
2.5
```

プログラミングでは、テキストデータを扱うことが多いです。「文字列型」はテキストデータを扱う型なので、プログラミングができるようになるには文字列型の習熟が欠かせません。

文字列作成記号

数字はそのまま書けば認識されていましたが、文字列は「特別な記号」でテキストを囲むことで Python が文字列と認識します。(囲まなければ変数名だと判断されます)

今まで文字列を作る記号として「シングルクオテーション」と呼ばれる「'」を使っていましたが、これは文字列を作る特別な記号のうちのひとつです。以下のように使うのでした。

```
>>> text = 'hello python'
>>> print(text)
hello python
```

実はシングルクオテーションだけでなく「ダブルクオテーション」と呼ばれる「"」でも文字列が作成できます。

```
>>> text = "hello python"
>>> print(text)
hello python
```

特に両者に大きな違いはありませんが、「シングルクオテーションが Python の標準的な書き方」であることと、「シングルクオテーションの文字列の中ではダブルクオテーションが使える」こと、またその逆の「ダブルクオテーションの文字列の中ではシングルクオテーションが使える」ことは覚えておいて下さい。
ダブルクオテーションを使うのは内部にシングルクオテーションを保つ場合のみにするのがよいかと思います。

文字列の演算
文字列も数値と同じように演算することができます。先に示したように「+」で結合もできますし、あまり知られていませんが掛け算である「*」記号で同じ文字列を繰り返すこともできます。

```
>>> text = 'hello' + ' python'
>>> print(text)
hello python
>>> text = 'hello' * 3
>>> print(text)
hellohellohello
```

文字列と他の型(例えば数値型など)を結合して新しい文字列を作る場合は、他の型を文字列型にキャストする必要があります。

Bool 型
Bool 型は別名「真偽値」と呼ばれます。真偽値という名前を聞くとなんだか難しそうに思えるかもしれませんが、要するに真偽に相当する「True/False」という「YES/NO」「正/非」の 2 値しかない単純な型です。
Python のプログラム中で「True」と「False」という文字は予約語として登録されています。それぞれその言葉のとおりに、YES/NO として Python に解釈されます。
Bool 型は今後学ぶ条件分岐やループ処理で利用されます。たとえば条件を満たす True の場合は処理 A を実施し、条件を満たさない False の場合は処理 B を実施するといったように、プログラムの制御に関わります。
Bool 型は自分で「True/False」を直接プログラムに書くだけでなく、「比較演算子」を使った演算結果や、関数の返り値として得られます。
Bool 型は様々な箇所で使われていますので、一箇所でまとめて扱うのではなく個別のテーマに付随する形で詳細を扱います。具体的には以下のようにしています。
Bool 型の解説ページ:全てに関わる一般的なことについて
　個別の型の解説ページ:その型の Bool 型に関わる関数や演算子、処理について
　その他のページ:解説する処理などに関わる Bool 型の使い方について

比較演算子
Bool 型の値は「比較演算子」と呼ばれる記号で 2 つの値を比較した際に返されます。例として数字の大小を比較してみます。
この比較演算子は算数で習った「大なり」記号である「>」や、「小なり」記号である「<」などです。
「10>5」は「10 は 5 より大きい」という意味で、それはあっていますので結果は「True」となります。

```
>>>10>5
True
```

同様に「10<5」は「10 は 5 より小さい」という意味で、これは間違っているため「False」が返ります。

```
>>>10 < 5
False
```

比較演算子の演算で得たブール型の値も数値や文字列と同じように変数に代入可能です。

```
>>> a = 10 > 5
>>> a
True
```

以下に代表的な比較演算子を記載します。
| 代表的な演算子 | 説明 | | A > B | A が B より大きければ True | | A >= B | A が B 以上なら True | | A < B | A が B より小さければ True | | A <= B | A が B 以下なら True | | A == B | A と B が同一なら True | | A != B | A と B が異なれば True |

どちらが大きいか小さいかの比較は算数の通りなので分かりやすいかと思います。プログラミング独特のものとしては「==」と「!=」があり、それぞれ「同じか」「同じでないか」を調べます。

```
>>> 'hello' == 'world'
False
>>> 'hello' != 'world'
True
```

リスト

「リスト」が本ページで紹介する最後の型です。その名前からわかるように「データをリスト状に並べた」値を持つ型です。数値や文字列はそれ自体がデータですが、リストはそれらを格納する入れ物のような型になります。

リストはループ処理に欠かせない型です。どういった用途で利用されるのかはループ制御のページで解説し、ここでは最も初歩的な操作方法のみ扱います。リストの詳細についてはリスト型のページをご参照ください。

リストの作成

リストは「[]」の記号のなかにコンマ(,)区切りでデータを羅列することで作成されます。たとえば「[1,2,3,4,5]」とすると、内部に「1,2,3,4,5」というデータが格納されたリスト型の値が作成されます。

```
>> a = [1,2,3,4,5]
>>> print(a)
[1, 2, 3, 4, 5]
>>> type(a)
<class 'list'>
```

リストの内部に格納する値はなんでもよいですが、一般的に複数の型の値を混ぜることはありません。リストの解説ページで扱いますが、リストのなかにリストを収める多重構造にする場合もあります。

内部に値を持たないリストを作ることもでき、その場合は単に「[]」とします。

リストの要素へのアクセス

リストが持つデータは「要素」と呼ばれており、前から何番目かという意味の「インデックス番号」を指定して中身の操作をします。注意をしてほしいのは「インデックス番号は 0 から数える」ということです。

たとえばリスト「[1,2,3,4,5]」のインデックス番号 0 の値は「1」で、インデックス番号 1 の値は「2」となります。

リストの要素へのアクセスは「リスト型の値[インデックス番号]」とします。要素の取得をしても中身はなくなりません。

```
>>> a = [1,2,3,4,5]
>>> a[0]
1
>>> a[3]
4
>>> print(a)
[1, 2, 3, 4, 5]
```

要素の変更は、インデックス番号でリストの要素を指定してそれに代入するという手法になります。

```
>>> a = [1,2,3,4,5]
>>> a[1] = 100
>>> print(a)
[1, 100, 3, 4, 5]
```

リストの注意点

リストには長さがあります。要素が 5 つ入っているリストの長さは 5 であり、インデックス番号は 0 から 4 まで使えます。

リストの長さを超えて要素にアクセスすると、エラーが発生します。

```
>>> a = [1,2,3,4,5]
>>> a[3]
4
>>> a[4]
5
>>> a[5]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

リストの長さを得るには「 len 関数 」を使います。

```
>>> a = [1,2,3,4,5]
>>> len(a)
5
```
つまり、アクセスしてよい最大のインデックス番号は「len(リスト値) − 1」までとなります。

1-2.　基本構文

1-2.-1. 繰り返し:ループ文

ループは同じ処理を何度も繰り返すためのプログラミングの制御方法です。

ループを使わずに似たような処理をなんども行う場合は、それを全てプログラムにベタ書きする必要があります。処理の回数に融通が効かず、無駄に長く、修正を加える場合も大変です。そのため、同じ処理を繰り返す場合はループを使います。

Python のループ処理は「for 文」か「while 文」を使います。for 文はリスト型の値(もしくはそれに似た型の値)の要素に対してなんらかの処理を繰り返すような使い方をし、while は条件式を満たすまでループ処理を繰り返すという使い方をします。

for によるループ

プログラミングの制御構造に「同じ処理を何度も繰り返す」というものがあり、それは「ループ」と呼ばれています。ループは少し複雑なプログラムでは必須となるテクニックです。

「for 文」は Python でループ処理をするための文法です。for 文に与えたリストの全ての要素に対して、同じ処理を実施します。

for 文は if 文と同じく制御のルールを書いた後で、実行する処理を書くコードブロックを続けます。for 文の構造は「for 変数 in リスト: 」とすると、リストのヘッド(インデックス 0)から要素を取り出して変数に格納し、for 文のコードブロックを実行します。

for 文のコードブロックの処理が終了すると、次の要素を取り出してコードブロックを再度実行します。これをリストの全てに対して行うまで繰り返します。

コード: for 文

```
t=0
for i in ..10
  t+=i
end
print(t)
```

for 文の構文のなかで必ずしも様子を格納する変数を使う必要はありません。また、要素の数が 0 であるリストを渡された場合は一度も処理を実行しないだけです。

ループ処理の必要性

たとえば「クラス全員のテストの平均点を求める」という処理を自分で実装するとしましょう。ループを使わなければ生徒全員の点数を「78 + 82 + 64 + 74 + …」として合計値を求め、それを人数で割るというという処理をします。

この時、生徒の人数が変わると足し合わせる処理の数が増減し、割り算の値も変わります。プログラムの処理が生徒の数に依存しているということです。

4 人の平均のプログラムは上記のものとなります。

新しい変数の作成、合計値の取得、割り算でコードが変更されています。

リストと for 文を使うとプログラムは以下のようになります。

```
students_score = [78, 82, 64, 74]
sum_score = 0
num_students = 0
for score in students_score:
  sum_score += score
  num_students += 1
ave_score = sum_score / num_students
```

このプログラムは先ほどのループを使わないプログラムと異なり、生徒の合計点を得る処理と、頭数での割り算の処理は生徒の人数に依存していません。10 人だろうと、100 人だろうと同じです。

```
students_score = [78, 82, 64, 74, 98]
sum_score = 0
num_students = 0
for score in students_score:
```

```
    sum_score += score
    num_students += 1
  ave_score = sum_score / num_students
```

for をシーケンス型に使う

for 文はリスト型以外でも使うことができます。たとえば文字列にたいして for 文を使うと、文字列を構成する文字ごとにループを回すことができます。

```
  text = 'hello'
  for character in text:
    print(character)
  h
  e
  l
  l
  o
```

while によるループ

「 while 文 」も for 文と同じくループ処理のための制御構造です。両者はループを回すためのルールが異なっており、while は「条件式が True なら処理を繰り返す」という動きをします。 while 文は「 while 条件式: 」を書き、それに続けて処理をするためのコードブロックが続きます。

```
  a = 5
  print('start')
  while a > 0:
    print(a)
    a -= 1
  print('end')
  start
  5
  4
  3
  2
  1
  end
```

for 文はループを回す回数が限られています。一方、while は条件式が True である限り何周でもするため、条件式が False にならないと永遠にループをし続けます。このような終わりのないループは「無限ループ」と呼ばれています。

break

for 文によるループは全ての要素を処理するまで回り続け、while によるループは条件式を満たさなくなるまで回り続けます。

ただ、「もうループを回す必要がない」という状況になることがあります。その場合には「 break 文 」を使ってループ処理を打ち切ります。

たとえばリストの中から特定の条件を満たす要素があるか否かということを調べるとします。今回は「5 である」という条件とします。

リストを先頭から順にループで調べていき、そこで 5 という要素を見つけた場合は、それ以後のループは回す必要がありません。

なぜなら 5 が存在するかどうかを調べているため、存在することが分かった時点でそれより後ろの要素を調べる必要がなくなるからです。そういった状況で break 文を使ってループを終了します。

break 文を使った要素 5 を探すプログラムを書くと以下のようになります。

```
  a = [3, 9, 10, 4, 5, 1, 8, 5, 3]
  has_5 = False
  for i in a:
    print(i)
```

```
  if i == 5:
      has_5 = True
      break
print("List has 5: " + str(has_5))
3
9
10
4
5
List has 5: True
```

ループで何かを探す際によく使われるテクニックですが、ループの前に「探している要素が存在するか」という変数を作って、そこに「存在しない」ということで False を代入しておきます。

ループを回した時に要素をチェックし、それが探している値(今回だと 5)であれば「存在する」として True を代入します。そして以後はループを回す必要がないので break 文でループを打ち切ります。ループが終わった場所で「探している要素が存在するか」という変数を調べれば、あったかなかったかが分かります。

このループ処理では要素を毎回 print 出力しています。3 から 5 までは各要素が出力されていますが、5 以降の 1 と 8 は出力されていません。ここから要素が 5 であった際に break 文を呼び出してループを抜け、残りの要素はループ処理されていないことが分かります。

continue

ループを回している際に「この回は条件を満たしていないので、処理をしなくていい」という状況になることがあります。

たとえばループで要素 5 に対してだけ処理をしたい場合であれば、要素が 5 以外であった場合は処理を継続する必要はありません。ただ、先ほどの break とは異なり、リストの残りの要素も処理するためループ自体は終了できないとします。

このような場合は「continue 文」を使うことで、いま実行されている回の処理を打ち切ります。打ち切った後はループを抜けるのではなく、for 文の場合は次の要素で処理を開始し、while 文の場合は条件式のチェックからやり直します。

リストの要素が 5 であった場合のみ print 処理をするプログラムを記載します。

```
a = [3, 9, 10, 4, 5, 1, 8, 5, 3]
for i in a:
  print(i)
  if i != 5:
      continue
  print('Found 5')
3
9
10
4
5
Found 5
1
8
5
Found 5
3
```

continue を呼び出した後は以後の処理がされなくなるため、「5 でない要素」は「'Found 5'」と出力がありません。

continue 文の代わりに if 文で同じことを実現することもできます。たとえば上記であれば「5 であれば'Found 5'と出力」としたほうが簡単です。

ただ、continue を使うと実行する条件が複雑な時にプログラムがシンプルに組める場合が多いです。また、処理を打ち切っているということが明確に分かるので、その回の処理を飛ばす際は continue の利用を検討してください。
-

1-2-2. 条件分岐文

条件分岐は「ある条件 A を満たす場合に B をする」といったかたちで、特定の条件に応じてプログラムの動きを変えるための Python の文法です。

条件分岐には if/elif/else の構文を使い、それぞれ「条件を満たすなら処理をする」「それより上の条件分岐を満たしておらず、この条件を満たすなら処理をする」「どの条件も満たさなければ処理をする」という役割を持ちます。if 文は必須ですが、elif と else は必ずしも必要ありません。そして elif は好きなだけ使えますが、if と else は 1 つまでです。

条件分岐

条件分岐は、条件分岐の式を満たすか満たさないかで実行される処理が変わるという制御構造です。Bool 型の値が条件判定に利用され、それが True なら処理 A をし、False なら処理 B をするといった形で、条件により実行するプログラムが変わります。

if 文

Python だけでなく多くのプログラミング言語は条件分岐を「if 文」で実現します。if 文のあとに「条件式」を書き、その条件式が満たされる場合に if 文の処理が実行されます。

条件式は「Bool 型の値」として判定されるコードで、一般的には「Bool 値を持つ変数」や「比較演算子を使った式」が使われます。文法としては以下のように使います。

if 条件式:

条件式が True の時に実行される処理

if のあとに条件式を書き、それに続けてコロン「:」を書きます。そのあとで改行及びインデント(字下げ)をし、条件式が True になる場合に実行される処理を書きます。

実際の Python のプログラムですと以下のようになります。

```
a = 5
if 10 > a:
  print('True: 10 > a')
True: 10 > a
```

上記の「10 > a」が条件式で、これが True となれば if 文の次の行の処理が実行されます。a には 5 が代入されているため、「10 > 5」は True となり、その次の print 文が実行されています。

a に代入する値を 100 に変えて、再度プログラムを実行してみます。

```
a = 100
if 10 > a:
  print('True: 10 > a')
```

今度は「10 > a」の a に 100 が代入されているため、条件式は False となります。そのため、if 文の次の行(条件式が True の場合だけ実行される)が実行されず、print 文の出力がありません。

コードブロック

先ほどの例では if 文が条件を満たす場合の処理を細かい説明なしに 1 行だけとしました。ただ、複雑なプログラムでは if 文の処理も複雑になるため複数行の処理を実行する必要がでてきます。

その際に if 文が「どこからどこまでをカバーしているか」をプログラムに教えてあげる必要があります。プログラミングでは条件分岐といったある特定の処理の「対象範囲」を「コードブロック」で示します。

Python ではコードブロックを「インデント(字下げ)」を揃えることで実現しています。具体的には以下のようなものです。

```
処理 1
if 条件式:
    # ここから
    処理 2
    処理 3
    # ここまでがコードブロック
```

処理 4

上記の例では if 文の後に if 文のコードブロックが続き、どこからどこまでが対象になるかをインデントで示しています。処理 2 と処理 3 は同じインデントレベル(深さ)で字下げされているので、同じコードブロックに属しています。

Python のインデントの仕方は「半角空白を 4 つ」が標準で、それに次いで半角空白 2 つも一般的です。タブでインデントをすることは推奨されておりませんが使えます。ただ、半角空白とタブを併用したインデントは避けて下さい。

具体例としては以下のようなものになります。

```
a = 5
print(1)
if 10 > a:
  print(2)
  print(3)
print(4)
```

```
1
2
3
4
```

変数 a に 5 が代入されているため、条件分岐で「10 > a」が満たされます。そのため if 文に属さない「print(1)」に続いて、if 文の処理である「print(2)」と「print(3)」が実行されます。

その後にある「print(4)」は if 文のコードブロック外なので、if 文の条件式に関わらず実行されます。

a に代入する値を 100 に変えて、条件式が False になる状態でプログラムを再度実行してみます。

```
a = 100
print(1)
if 10 > a:
  print(2)
  print(3)
print(4)
```

```
1
4
```

if 文のコードブロックに属する「print(2)」と「print(3)」が実行されなくなりましたが、コードブロックの範囲外である「print(1)」と「print(4)」は実行されています。

コードブロックのネスト

コードブロックの中にコードブロックを作ることも可能です。たとえば if 文による条件分岐の中に、さらに if 文の条件分岐を作るといった具合です。コードブロックに限らず、プログラミングで「入れ子」構造にすることを一般的に「ネスト」すると言います。

Python はコードブロックをインデントレベル(深さ)で示しますので、コードブロックの中のコードブロックはインデントレベルが 1 つ深くなります。具体的には以下のようになります。

```
if 条件式 A:
  処理 1
  if 条件式 B:
    処理 2
  処理 3
```

条件式 B を持つ if 文のコードブロックは、条件式 A の if 文のコードブロックの中にあります。コードブロックの中のコードブロックに属する処理 2 はインデントレベルが一つ下がっています。

実際のコードでは以下のようになります。

```
a = 5
b = 5
print(1)
if 10 > a:
  print(2)
  if 10 > b:
    print(3)
```

```
  print(4)
print(5)
```

1-2-3. 関数

関数は呼び出し元から「引数」として値を受けとり、内部で「処理」をして、結果を「返り値」として返します。引数と返り値は必須ではなく、どちらかもしくは両方がないこともあります。

関数は以下の役割を持っています

- Python が提供する機能を呼び出す
- プログラムを分かりやすくする
- プログラムの重複を減らす

Python が提供する関数を使うことでキーボードやファイルからの入力、ディスプレイへの出力といった処理を実現できます。

複雑な処理を Python が提供する関数で簡単に実現できたり、自分でその関数を作ったりすることで、プログラムの見通しがよくなります。

また、似たような処理をコピーペーストではなく関数を使って実現できるため、プログラムの重複が減ります。

Python が提供する組み込み関数

関数はプログラマが自分で作るものと、プログラミング言語の環境(Python)が提供するものがあり、後者を特に「組み込み関数」と呼びます。

たとえば出力をコンソールにする print 文も組み込み関数です。この関数に値を与えると、それがコンソールに表示されます。

```
print('Hello Python')
Hello Python
```

この関数の「()」にいれる値を「引数」と呼んでいます。関数の目線で「引き込む値」で、関数を使う側が関数に渡す値です。上記の例では「'Hello Python'」という文字列が引数であり、これが print 関数を使う側から print 関数に渡されています。

関数はそれぞれ役割を持っていて、print 関数は引数をコンソールに表示することが役割です。別の関数 len()はリストの長さを確認するための関数です。この関数は引数としてリストを受け取ると、リストの長さを求めるという処理をします。

ただ、先ほどの print 関数と異なり、この関数はリストの長さをプログラムの呼び出し側に「返す」という動きをします。この返す値のことを「返り値」と呼びます。

```
a = ['h', 'e', 'l', 'l', 'o']
b = len(a)
print(b)
5
```

先ほどの print 関数は引数を処理のなかで画面出力することが役割でしたので、特に値を呼び出し元に返す必要がありません。そのため返り値がありませんでした。

一方、関数 len は処理の役割がリストの長さを求めるということなので、求めた値を呼び出し元に返さないと呼び出した意味がありません。そのため、呼び出し元に返り値を返します。

関数は突き詰めると「呼び出し元が引数で値を関数に渡す」「関数の処理のなかで引数を使ってなにかをする」「関数が返り値として処理した結果を呼び出し側に返す」という 3 つの流れに集約されます。

関数の定義と利用

組み込み関数を使うだけでなく、関数を自分で作って使うこともできます。関数の定義も if 文や for 文と同じように、「def 文」を使って宣言し、コードブロックで処理を書きます。

リスト長を求める len()関数と同じことをする「mylen()」関数を定義します。

```
def mylen(list_):
  count = 0
  for(i in list_):
```

```
        count += 1

return count
```

def 文に続けて自分がつけたい関数名を書き、それに続けて()に囲まれた引数を書きます。 関数名や引数名は変数と同じように自分の好きな名前をつけられます。ただ、変数と同じく小文字のアルファベットとアンダーバーと数字を使うのが一般的です。

コードブロック内にその処理を書き、ここでは for 文を使って要素を全てループして数を数えています。求めた値を呼び出し側に返すには「return 文」を使います。

return 文は関数を打ち切るキーワードで、return 文以降のコードは実行されません。return 文の後ろに何も書かなければ返り値はなにもなく、後ろに値を書けばそれが返り値となります。return 文がない、もしくは if 文の関係で実行されなかったといった場合は、関数のコードブロックの最後まで実行されて返り値なしとなります。

自分が定義した関数を使う方法は組み込み関数と全く同じです。

```
def mylen(list_):
  count = 0
  for(i in list_):
     count += 1
  return count
a = ['h', 'e', 'l', 'l', 'o']
b = mylen(a)
print(b)
5
```

注意をして欲しいのは自分が作った関数を利用するのは定義をした後になるということです。上記では mylen の定義をしたあとで、それを使っています。もしこの順序が逆になれば「mylen は定義されていない」としてエラーが発生します。

なお、関数で定義している引数を「仮引数」と呼び、関数を呼び出す側で与える引数を「実引数」といいます。たんに引数といった場合はいずれかを意味しています。

引数

関数の引数は好きな数だけ定義できます。引数が存在しない場合は引数定義を「関数名()」と空にして、複数存在する場合は「関数名(引数名 1,引数名 2,引数名 3)」とコンマ区切りで並べます。

この関数の定義にある引数の数と、関数を呼び出す際に与える引数の数をあわせる必要があります。定義と呼び出しの引数の数があっていなければエラーが発生します。function は関数で、arg は引数、call は呼び出しの英語です。

```
def myfunction1():
  print('myfunction1 called')
def myfunction2(arg1, arg2, arg3):
  print('myfunction2 called')
myfunction1()
myfunction2(1, 2, 3)
myfunction1('test')
myfunction1 called
myfunction2 called
Traceback (most recent call last):
  File "sample.py", line 10, in <module>
     myfunction1('test')
TypeError: myfunction1() takes 0 positional arguments but 1 was given
```

引数の数を揃えた 1 番めと 2 番めの呼び出しは成功していますが、 引数がない関数を引数を与えて呼び出すと「関数 myfunction1 は引数を 0 受け取るが、1 つ与えられている」としてエラーになっています。

Python の関数の引数には中級者向けの様々なテクニックがあります。それらを使えば上記の引数の数が違う問題なども回避できます。これらについては関数の引数の詳細を扱うページにて解説します。

返り値なしは None を返す

返り値がない関数があると先ほど説明しましたが、厳密にいうとそれらの関数は「None」という値を返り値として返しています。None は日本語でいうと「何もない」ですので、何もないということを明示的に返しています。

先ほど print()関数は返り値がないと説明しましたが、あえて返り値を変数に代入して中身を確認すると、この None という値が返されていることが分かります。

```
a = print('hello')
print(a)
print(type(a))
hello
None
<class 'NoneType'>
```

なお、None が返されていることを「返り値は None」と明示的には言わず、「返り値はない」とされることが多いです。どちらの表現も間違いではありませんが「返り値は None」とわざわざ伝える必要はないです。

返り値がない自作関数

自作の関数で return 文を後ろの返り値なしで呼び出したり、関数内で return 文が呼び出されないと、None が返されます。

```
def myfunction1():
  1 + 1
def myfunction2():
  1 + 1
  return
print(myfunction1())
print(myfunction2())
None
None
```

複数の return 文

関数内に複数の return 文を書くことができます。これはループの中に書かれた break と似ていて、それが呼び出されたタイミングで全ての残りの処理が打ち切られて関数呼び出しが終了します。

プログラムの最後に break 文があれば関数の処理は最後まで実施されるが、途中にある場合はそれ以後は実施されないということです。

以下の関数では引数が 10 の場合のみ、if 文で return 文が実行されます。

```
def function(a):
 print(1)
  if(a == 10):
     return
  print(2)
function(3)
function(10)
1
2
1
```

関数のなかで複数の return 文を使うのであれば、全ての return 文が同じ形式の返り値を返すようにしてください。 状況に応じて引数 None が返されたり数値が返されたりすると、それを使う側は引数の型のチェックなどが必要になります。 そのような何が返されるか分からない関数は非常に使いにくいものとなります。

関数内の関数

Python の関数は実は関数の中に入れ子構造にすることができます。例えば以下のような使い方ができます。

```
def test1():
  a = 10
  def test2(x, y):
     return x + y
  return a + test2(5, 6)
print(test1())
# 21
```

上記の例では関数 test1()の中で関数 test2()を定義し、それを使っています。今回のサンプルのような短い関数であればこのような使い方は不要ですが、より複雑な処理を関数で行う場合は処理を綺麗に分類するために内部で関数を定義して使うことがあります。

2. プログラミングⅡ

2-1. データ操作(プログラミング実践Ⅰ)

2-1-.1. リスト(配列)

複数のデータの並びはリストで表します。0～9 の数を順番に並べたものはリストとして[0,1,2,3,4,5,6,7,8,9]と表すことができます。このようにリストは'['と']'括ったデータ構造である. リストの要素としては任意の型のデータを並べることができます。

例えば、['nakamura',51,'tanaka',22,'hashimoto',14]のように、型の異なるデータが要素として混在するリストも作成可能です。

また、リストを要素として持つリストも作成可能であり、例えば、['a',['b','c'],'d']といったリストも作成することができます。

リストの要素へのアクセス

リストの要素を取り出すには, 取り出す要素の位置を示すインデックスを'[...]'で括って指定します。これは次の例を見ると理解できる。

>>> lst = [2,4,6,8,10] ←リストの生成

>>> lst[1] ←1 番目の要素を指定

4 ←対象の要素が表示されています。

このようにリストの要素は 0 番目から始まる位置(インデックス)を指定してアクセスすることができます。

リストの中の、インデックスで指定した特定の範囲(部分リスト)を取り出すこともできます。

例. 部分リストの取り出し

>>> lst = [2,4,6,8,10]Enter ←リストの生成

>>> lst[1:4]Enter ←部分リスト(1～4-1 番目)の取り出し

[4, 6, 8] ←部分リストが得られています。

この例のように、コロン':'でインデックスの範囲を指定しますが、[n1:n2]と指定した場合は、n1～(n2−1)の範囲の部分を示していることに注意しなければなりません。

リストを用いることで、複数の要素を持つ複雑な構造を 1 つのオブジェクトとして扱うことができます。この意味で、リストは複雑な情報処理を実現するに当たって非常に有用なデータ構造です。

リストの編集

リストは要素の追加や削除を始めとする編集ができるデータ構造です。

リストを編集するための基本的な機能について解説します。

要素の書き換え

リストの指定したインデックスの要素を直接書き換えることができます。

例. リストの要素の書き換え

>>> lst = [0,1,2,3,4,5]Enter ←リストの作成

>>> lst[3] = 'x'Enter ←3 番目の要素を'x'に変更

>>> lstEnter ←内容確認[0, 1, 2, 'x', 4, 5] ←要素が変更されています

また以下のような操作が可能です。

- リストの連結
  演算子'+'によって複数のリストを連結し、結果を新しいリストとして作成することができます
  対象リスト.extend(追加リスト)
- 要素の削除
  del 削除対象のオブジェクト
  削除対象リスト.remove(値)

● 要素の存在検査
要素 in リスト

2-1-2. 辞書型(Hash)

辞書型オブジェクトはキーと値のペアを記録するもので、文字通り辞書のような働きをします。
辞書型オブジェクトの生成は{キー1:値 1,キー2:値 2, ...}とします。

例えば>>> dic ={'apple':'りんご','orange':'みかん','lemon':'レモン'}
とすると 3 つの英単語と和訳を保持する辞書 dic ができます。

この後、
>>> dic['apple']
として値を確認すると、'りんご'と表示されます。

辞書型のオブジェクトはキーの値でハッシュ化されており、探索が高速です。

キーと値のペアを新たに辞書に追加するには、dic['banana'] = 'バナナ'などとします。

● キーの存在検査
>>> 'grape' in dicEnter ←辞書 dic にキー'grape'が存在するか検査
False ←存在しない。

このように、キーin 辞書型オブジェクト という式を記述すると、キーが存在すれば True が、存在しなければ False が得られます。

空の辞書を生成するには dic = dict()とする. また、既存の辞書を空にするには clear メソッドを使用します。

● 辞書の要素の削除
del dic['banana']
● 辞書型オブジェクトから全てのキーを取り出すには、次のように keys メソッドと list 関数を使用します。
>>> dic={'apple':'りんご','orange':'みかん','lemon':'レモン'}r←辞書型オブジェクト生成
>>> k = dic.keys() ←全てのキーの取り出し
>>> k ←内容確認 dictkeys(['apple', 'orange', 'lemon']) ←結果表示

2-1-3. データ構造の変換

以下のような関数を使用してデータを変換することができます。

データ構造 d を別の型に変換する

list(d)

d をリストに変換します。

2-2.オブジェクトプログラミング

2-2-1. オブジェクト指向

オブジェクト指向プログラミング
　Python におけるオブジェクト指向の考え方も他の言語のそれと概ね同じです。ここでは、オブジェクト指向についての基本的な考え方の説明は割愛して、クラスとメソッドの定義の具体的な方法について説明します。

クラスの定義
　クラスの定義は class で開始します。
　《class の記述 》
書き方 1
　　:class クラス名:
(定義の記述)
「定義の記述」は class よりも右の位置に同一の深さのインデント(字下げ)を施して記述します。

コンストラクタ
　クラスのインスタンスを生成する際のコンストラクタは、クラスの定義内に次のように init を記述します。(init の前後にアンダースコアを 2 つ記述する)

《init の記述 》書き方:
　　definit(self,仮引数):
(定義の記述)
　仮引数は複数記述することができます。また仮引数は省略可能です。「定義の記述」は def よりも右の位置に同一の深さのインデント(字下げ)を施して記述します。
　self は生成するインスタンス自身を指しており、第 1 仮引数に記述します。ただし、コンストラクタを呼び出す際には引数に self は記述しません。

　コンストラクタはクラスのインスタンスを生成する際に実行される初期化処理です。
　インスタンスの生成は
　　インスタンス名=クラス名(引数)
　とします。
　「引数」にはコンストラクタの init に記述した self より右の仮引数に与えるものを記述します。

メソッドの定義
　クラスの定義内容としてメソッドの定義を記述します。これは関数の定義の記述とよく似ています。

《 メソッドの定義1 》
　インスタンスに対するメソッド書き方:
　　def メソッド名(self,仮引数):
(定義の記述)
　仮引数は複数記述することができます。また仮引数は省略可能です。
　「定義の記述」は def よりも右の位置に同一の深さのインデント(字下げ)を施して記述します。
　self は対象となるインスタンス自身を指しており、第 1 仮引数に記述します。当該メソッド呼び出し時には引数には self は記述しません。

3. オブジェクトプログラミング基礎

3-1. モジュール

Python が提供する機能は多く、print 関数といったよく利用される機能だけでなく、数学やネットワーク関連の機能もあります。

これらの補助的な機能については「モジュール」という形で Python はプログラマに提供しています。既に作成されているモジュールを取り込んで使うことで、自分で作成できない機能を利用し、開発の手間を省くことが可能になります。

また、Python はモジュールの集合をパッケージとしても提供しています。複雑なライブラリだと 1 つのモジュールで全ての機能を提供するのではなく、複数のモジュールをパッケージとしてグループ化して提供します。

モジュールノの全体像

「モジュール」はプログラムを分割する手法です。モジュールは Python 自体が提供する環境でも利用されており、自分でモジュールを作って利用することもできます。

Python 自体が提供するモジュールは「標準ライブラリ」と呼ばれています(ライブラリはモジュールの集合というニュアンスがあります)。これには今まで利用してきた print 関数といった特に宣言無く利用できる「組み込みモジュール」と呼ばれるものと、数学やネットワークの処理といった用途ごとのモジュールがあります。

Python が提供するこれらのモジュールの一部は C 言語で書かれていて、Python のプログラマが直接使うことができないシステム機能へのアクセス(たとえば画面出力など)を提供しています。

Python 環境自体が提供する標準ライブラリ以外にも、普及しているモジュールはあります。それは pip と呼ばれるパッケージ管理システムを使って追加インストールをしたり、ソースコードとして配布されているものを取り込んだりして使います。

pip で提供されるのはモジュールではなく複数のモジュールを含む「パッケージ」というものです。複雑な機能を実現するプログラムを 1 つのモジュールに詰め込むことはできないため、複数のモジュールに分けてそれをパッケージという入れ物にいれて提供します。

大規模なソフトウェアを開発する場合は自分が書くソースコードの量も数千行、数万行となります。そのときはプログラムの機能ごとにファイルを分けて、それらをモジュール化します。自分が開発したモジュールを自分のプログラムで読み込むことで、コードを整理できます。

以下に Python のモジュールの全体像を記載します。

モジュールの利用

モジュールを利用するためにはその宣言が必要です。その宣言方法は複数ありますが、最もよく使われるのが「import 宣言」です。

import 宣言は「import 文」に続けて、モジュール名を書きます。たとえば、数学処理がまとめられた math モジュールを利用するには以下のようにします。

```
import math
```

import 宣言されたモジュールは「モジュール名.モジュールの関数」などとして、頭にモジュール名を付けて利用します。たとえば、math モジュールの切り捨て関数である floor を使うには以下のようにします。

```
>>> import math
```

```
>>> math.floor(5.5)
5
```

仮に import 文なしに math モジュールの floor 関数を使おうとすると、math という名前は宣言されていないとエラーが発生します。

```
>>> math.floor(5.5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'math' is not defined
```

モジュールが持つのは関数だけでなく、クラスや定数、変数などもあります。それらも共通して「モジュール名.定数」などとして使います。

import A as B

import で読み込むモジュール名が長かったりわかりにくかったりする場合は、モジュールにに別名を付けることが可能です。

別名をつけて import するには「import モジュール名 as 別名」とします。たとえば GUI のライブラリである「math」を「mathematics」として読み込むには、「import math as mathematics」とします。

```
>>> import math as mathematics
>>> mathematics.floor(5.5)
5
```

from A import B

モジュール名を書かずにモジュールの関数などを使うこともできます。それには「from モジュール名 import 使う関数など」とします。

たとえば「from math import floor」とすると、floor 関数を math モジュールの後ろに付けずに使うことができるようになります。

```
>>> from math import floor
>>> floor(5.5)
5
```

頻繁に使う関数であればこのように使用しても構いませんが、一般的には from を使わずに import のみを使うことがよいとされています。なぜならその関数がどのモジュールに属しているかひとめで分かるからです。

あまり見かけませんが、先ほどの「import A as B」と from を組み合わせることで、あるモジュールに属する関数を別名で使うこともできます。

```
>>> from math import floor as f
>>> f(5.5)
5
```

from A import *

モジュール内の全ての関数やクラスなどを、モジュール名なしで呼び出すこともできます。それには「from モジュール名 import * 」という宣言をします。

3-2. オブジェクトプログラミング例

以下のような実例をやってみよう

3-2-1. サンプル 1

```
class Item:
    item_list = [
        {'name': 'None', 'price': 0},
        {'name': 'pencil', 'price': 100},
        {'name': 'eraser', 'price': 80},
        {'name': 'compass', 'price': 300},
    ]

    def __init__(self, item_id):
        item = self.search_item_by_id(item_id)
        self.id = item_id
        self.name = item['name']
        self.price = item['price']

    def search_item_by_id(self, item_id):
        return self.item_list[item_id]


class Cart:
    def __init__(self):
        self.items = []

    def add_item(self, item, number):
        for i in range(number):
            print(item.name + ': ' + str(item.price) + '円')
            self.items.append(item)

    def calc_subtotal(self):
        p = 0
        for item in self.items:
            p += item.price
        return p

def add_tax(price):
    return int(round(price + price * TAX_RATE * 0.01))

# 税率を設定
TAX_RATE = 8
# 商品をカートに追加して小計と合計を計算
my_cart = Cart()
my_cart.add_item(Item(1), 2)  # 鉛筆 2 個
my_cart.add_item(Item(2), 1)  # 消ゴム 1 個
my_cart.add_item(Item(3), 1)  # コンパス 1 個
subtotal = my_cart.calc_subtotal()  # 小計
# staticmethod はインスタンス化せず クラス名.メソッド名() で実行可能。
total = add_tax(subtotal)  # 合計

print('-----------------')
print('小計: ' + str(subtotal) + '円')
print('合計: ' + str(total) + '円')
```

```
print('------------------')
```

3-3-2. サンプル 2

```
import group as gp
import sys

class Person:
    def __init__(self,name,kokugo,sugaku):
        self.name=name
        self.kokugo=kokugo
        self.sugaku=sugaku
class Group:
    def __init__(self):
        self.persons=[]
        self.num=0
    def plus(self,p):
        self.persons.append(p)
        self.num+=1
    def avg(self):
        k=0;s=0
        for i in range(self.num):
            k+=self.persons[i].kokugo
            s+=self.persons[i].sugaku
        return k/self.num,s/self.num

aGroup=Group()
aGroup.plus(Person("Ichiro",80,80))
aGroup.plus(Person("Jiro",70,75))
aGroup.plus(Person("Saburo",90,90))

for i in range(aGroup.num):
    print(aGroup.persons[i].name)
    print(aGroup.persons[i].kokugo)
    print(aGroup.persons[i].sugaku)
print(aGroup.avg())
```

group.py のモジュール

```
class Person:
    def
__init__(self,name,kokugo,sugaku):
        self.name=name
        self.kokugo=kokugo
        self.sugaku=sugaku
class Group:
    def __init__(self):
        self.persons=[]
        self.num=0
    def plus(self,p):
        self.persons.append(p)
        self.num+=1
```

本報告書は、文部科学省の生涯学習振興事業委託費による委託事業として、≪学校法人誠和学院　専門学校日本工科大学校≫が実施した　2019年度　「専修学校による地域産業中核的人材養成事業」の成果をとりまとめたものです。

2019年度　文部科学省委託事業「専修学校による地域産業中核的人材養成事業」
～Society5.0等対応カリキュラムの開発・実証～

Society5.0社会を支えるエンジニア育成事業 ＡＩ基礎テキスト

２０２０年　２月発行

発行所・連絡先
学校法人誠和学院　専門学校日本工科大学校
〒672-8001　兵庫県姫路市兼田383-22
TEL 079-246-5888　　FAX 079-246-5889
http://www.seigaku.ac.jp/

本書の内容を無断で転記、転載することを禁じます。