第 2 回　プログラミング言語について

プログラミング言語の役割について学ぶ。

【ハードウェアとソフトウェア】

パソコン、スマホ、テレビ、冷蔵庫、洗濯機、インターフォン、ウォシュレット、その他あらゆる電子機器には、ハードウェアとソフトウェアがある。

ハードウェア（hardware）とは、物理的な構成要素である。たとえば、モニタ、キーボード、マウス、CPU（中央処理装置）、メインメモリ（主記憶装置）、外部記憶装置（ハードディスク、SSD、USB メモリ）、ケース、マザーボード、などである。

ソフトウェア(software)とは、ハードウェアとしてのコンピュータを動作させるための、プログラム、データなどの総称[1]であり、ハードウェアのような物理的実体はない。コンピュータはソフトウェアの指示に従って、その目的とする処理を行う。ソフトウェアを取り替えることで、コンピュータはさまざまな処理を行うことができる。「コンピュータ、ソフトがなければ ただの箱」と言われるように、コンピュータにとってソフトは必要不可欠なものである。

【プログラム】

プログラミング言語とは、コンピュータに理解できるよう、動作手順を表現してコンピュータに伝える人工的な言語であり、動作手順をプログラミング言語の言葉として表現したものがプログラムである[2]。

【基本ソフトウェア】

基本ソフトウェア（OS: Operating System)とは、コンピュータを利用するための基本的な制御やプロセス管理などを行うソフトウェアであり、基本ソフトウェアの上で応用ソフトウェアが動く。基本ソフトウェアは、ハードウェアを制御する基本的なソフトウェアを用意して、応用ソフトウェアとハードウェアとの仲立ちをする[3]。

【パソコンの主な OS】

- MS-DOS（Microsoft Disk Operating System）
  - Microsoft 社が 1981 年に開発した IBM PC 用の OS。
  - その後、バージョンアップを重ね、当時の IBM PC/AT 互換機パソコン用 OS の業界標準となった。
- macOS
  - Apple 社のパソコン Macintosh 用の OS。OS の名称は Mac OS, Mac OS X, OS X などと変更され、2016 年に開発されたバージョン 10.12 から macOS となった。
  - 早くから優れた GUI を備え、Windows などに大きな影響を与えた。
- Windows
  - Microsoft 社が MS-DOS の後継として開発した OS。
  - 1985 年に最初のバージョン 1.0 が発表された後、3.1, 95, 98,XP,Vista,7,8,10,

---

1 JIS の定義では、ソフトウェアは「データ処理システムを機能させるための、プログラム、手順、規則、関連文書などを含む知的な創作」とされていて、この場合、規則や関連文書も「ソフトウェア」に含まれる。

2 ソフトウェアは、コンピュータに関わるすべての知的情報で、プログラムは、コンピュータの動作手順を表したものに限定されるため、ソフトウェアの方がプログラムよりも広い概念で、プログラムは、ソフトウェアに含まれる。

3 JIS の定義では、「プログラムの実行を制御するソフトウェアであって、資源割振り、スケジューリング、入出力制御、データ管理などのサービスを提供するもの」とされている。

11 などバージョンアップを重ねている。

- UNIX
  - AT&T のベル研究所で 1969 年に開発されたワークステーション等で使われた OS。
  - 大学や研究機関を中心に広く普及。
- Linux （リナックス）
  - フィンランドの大学生リーナス・トーバルズが、UNIX 互換の OS カーネルを開発。
  - オープンソースソフトウェアとして公開され、改変が自由に行われている。

【スマホの主な OS】

- Android
  - Google が開発している OS で、スマホのメーカーがカスタマイズできる。
- iOS, iPadOS
  - Apple が iPhone や iPad などの自社製品のために開発している OS。

【応用ソフトウェア】

応用ソフトウェア（アプリケーションソフトウェア application software; アプリ app）とは、一般のユーザーが利用する目的で作られたプログラムである。プログラムの開発元が作成したアプリをユーザーがパソコンあるいはスマホの OS にインストールし（あるいは、製品購入時にすでにインストールされている）、実行する。たとえば次のようなものがある。

1. ワープロソフトウェア
2. 表計算ソフトウェア
3. データベースソフトウェア
4. プレゼンテーションソフトウェア
5. ウェブブラウザ
6. グラフィックスソフトウェア
7. コミュニケーションアプリ
8. 業務ソフトウェア
9. ゲームソフトウェア

誰かが作ったプログラムをインストールして実行するだけでなく、自ら応用ソフトウェアのプログラミングをして実行することもできる。この授業では自らが作成したプログラムを実行するという経験をする。

【プログラミング言語の種類】

1. 機械語（マシン語）：CPU が解釈し実行できる唯一の言語。機械語のプログラムは、バイナリ形式で記述され、CPU が直接実行できる命令の集合体。
2. アセンブリ言語：機械語の命令を人間に分かりやすいような表現で記述したもの。アセンブラにより、機械語に変換してから実行される。
3. 高水準言語：自然言語に近く、人間の考えを表現しやすいプログラミング言語。人間が理解しやすい構文（自然言語に近い抽象度)」を持つ言語。

高水準言語によって書かれたプログラムの実行方法には、コンパイラによって機械語のプログラム（実行ファイル）に変換してから実行する方法と、インタプリタを起動して、インタプリタにプログラム（ソースファイル）を読み込ませて実行する方法がある[1]。

[1] 最初はインタプリタとして動作して、コンパイルが完了すると高速に実行する動的コンパイルなどの手法もある。

【主な高水準言語】
特に有名なものを選んで簡単に紹介する。

1. Fortran：1954 年に IBM が開発。コンピュータにおいて広く使われた最古の高水準言語であり、特に科学技術計算の分野では今でも利用されている言語。
2. LISP：1958 年に計算機科学者のジョン・マッカーシーが考案。現在広く使われているプログラミング言語の中では、Fortran に次いで古い。S 式というリスト構造の表記法を使うことが特徴であり、そのために大量の括弧を使うところが印象的である。
3. COBOL：1960 年に CODASYL が開発。一部の大規模な基幹システムでは、現在も保守・運用のために利用され、金融・政府・保険業界でいまだに現役である。
4. BASIC：1970 年代以降にパソコン用の言語として広く使われた言語。Visual Basic, Visual Basic .NET (VB.NET)などへ発展し、Microsoft Office には VBA (Visual Basic for Applications)として搭載されている。
5. Pascal: 1970 年に構造化プログラミングとして設計され、特に 1980 年代に人気を博した言語。Turbo Pascal や Delphi などの統合開発環境が発売された。
6. C：1972 年に開発されて UNIX の記述に用いられるようになり、現在も幅広く利用されている言語。ハードウェアに近い操作（ポインタ操作など）が可能。その後、様々な C 系言語が派生した。
7. Smalltalk：最初の純粋なオブジェクト指向言語の一つであり、オブジェクト指向パラダイムの先駆者。1972 年に開発が開始され、1980 年に公開された。
8. C++：C 言語をベースに、オブジェクト指向などの機能を取り込んで 1983 年に公開された。現代では、大規模な開発に適したマルチパラダイム言語として扱われている。
9. Objective-C: 1983 年に Brad Cox らが開発。その後スティーブ・ジョブズの NeXT コンピュータの主力言語となり、権利を買い取る。さらにアップルが NeXT 社を買収し、Mac OS X の Cocoa フレームワークのコア言語として採用した。
10. Perl: ラリー・ウォールが 1987 年に開発したスクリプト言語。テキスト処理やシステム管理、ウェブアプリケーションなどに使われ、Perl 5 系では高い後方互換性が維持されている。
11. Haskell: 純粋関数型プログラミング言語。習得の難易度は高いがバグが発生しにくいとされている。1987 年にアメリカのポートランドで開かれた関数型プログラミング言語に関する学術会議で委員会が結成され、1990 年に作成された。
12. Python：1991 年に公開されたスクリプト言語。読みやすく簡潔にコードが書けるように設計されている。この授業で採用するプログラミング言語であるため、この後にさらに詳しく紹介する。
13. R: 統計解析やデータ可視化に特化したプログラミング言語。1993 年に Ross Ihaka と Robert Gentleman によって開発され、学術分野やデータ分析で広く用いられている。豊富なパッケージが利用可能で、データ処理や機械学習にも対応。
14. Lua: 1993 年にブラジルで開発された軽量スクリプト言語。組み込み用途向けに設計されており、ゲーム開発やアプリケーションの拡張機能で広く利用される。シンプルで高速、柔軟な文法が特徴。
15. Ruby: 1995 年にまつもとゆきひろが開発したスクリプト言語。日本で開発されたプログラミング言語としてははじめて国際電気標準会議で国際規格に認証された。ウェブアプリケーションフレームワークの Ruby on Rails が広く使われている。
16. Java：1995 年に Sun Microsystems が開発し、Oracle が開発を引き継ぎ、オープンソースとしても開発されている。C++風の構文と Smalltalk のオブジェクト指向を組み合わせた、機種非依存のオブジェクト指向言語。企業の業務システム、PC やスマホのアプリ、家電製品の組み込み、など幅広い用途に利用される。
17. JavaScript: 1995 年に Netscape Communications が開発し、Netscape Navigator というウェブブラウザで実装された言語。主要なウェブブラウザ上で動作するため、動的

なウェブサイト構築やウェブサイト上でのアプリケーション開発に用いられる。現代ではウェブブラウザ外でも使われる。Java と混同されがちであるがまったく異なる。

18. PHP: ラスマス・ラードフが 1995 年に公開したスクリプト言語。ウェブアプリケーション開発で使われている。
19. C#: Microsoft が 2000 年に開発したプログラミング言語。C、C++、Java、Delphi などの影響を受けている。現在はクロスプラットフォームの .NET 上で動作し、企業システムやゲーム開発で広く使われている。
20. Scratch: 子供に楽しくプログラミングを学習させることを目的に 2006 年に MIT メディアラボが開発した。パレットにブロックを並べていくというインターフェイスを採用しているため、敷居が低くなっている。
21. Rust: C や C++に代わるシステムプログラミング言語として設計され、性能とメモリ安全性、並行処理の安全性を両立する。2006 年に Graydon Hoare が開発を始め、2009 年から Mozilla Research の公式プロジェクトとなった。
22. Go: 2009 年に Google が発表したプログラミング言語。golang とも呼ばれ、簡単に書ける並行処理機能が特徴で、クラウドアプリ開発などで広く使われている。
23. 2010 年代前半に開発された Dart, TypeScript, Swift, Kotlin, Julia は、いずれもこれまでの多くのプログラミング言語の良い特徴を取り入れて使いやすくされていることから人気が高く、今後の発展が期待されている。いずれもオープンソースである。
24. クロスプラットフォームアプリ開発フレームワークとして、Dart ベースの Flutter, JavaScript, TypeScript ベースの React Native, Cordova, Ionic, C#ベースの .NET MAUI（旧 Xamarin）などが使われている。

【Python について】

Python（パイソン）は、グイド・ヴァン・ロッサム[1]が 1991 年に公開したプログラミング言語であり、世界中のプログラマーのコミュニティによって開発が続けられ、主要な開発プロセスが PEP（Python Enhancement Proposal）に基づいて透明に行われている。2001 年には非営利団体 Python ソフトウェア財団（PSF）が立ち上がり、フリーソフトウェアの PSF ライセンスで配布されている。

Python は多くの環境で動作し、簡潔にコードを書けることが特徴として挙げられる。科学計算でよく使われ、数値計算ライブラリの NumPy や科学計算ライブラリの SciPy、データ解析ライブラリの pandas、グラフ描画ライブラリの Matplotlib などを使うことで、高度なデータ処理や統計解析をすることができる。

このように学術的な分野で人気が高い Python であるが、実務でも Django のような Web アプリケーションフレームワークによって Web サイトの構築ができることから、よく使われている。YouTube、Instagram、Dropbox は Python で開発されたとされている。

この授業では、Scratch のような教育用のプログラミング言語を使わずに、Python を学習する。この授業をきっかけとしてプログラミングを本格的に学び、これからの大学での学習や就職後に使うスキルとして役立てるためには、よく使われている言語で習熟する方が良いと考えるためである[2]。まずは Python で簡単なプログラミングができるようになることで、必要に応じて他の言語も自習により習得できるようになるであろう。

【★課題】

ToyoNet-ACE の小テストを提出期限までに受験すること。提出期限をすぎた場合は理由の如何を問わずに提出を受け付けない。

---

[1] Guido van Rossum – Personal Home Page https://gvanrossum.github.io/

[2] プログラミング言語の人気については https://redmonk.com/ / https://www.tiobe.com/tiobe-index/ / https://spectrum.ieee.org/ / https://youtu.be/Og847HVwRSI などを参照