2. 파이썬 언어 특징 및 변수의 생성 과정

Made By. 김규일

프로그래밍 언어의 이해와 프로그래밍 언어의 종류

- 프로그래밍 언어를 나누는 기준에는 많은 것이 있다.
  - 저수준 언어 vs 고수준 언어
  - 객체 지향, 절차 지향, 함수 지향 등의 기준
  - 명령형 언어 vs 선언형 언어
  - 컴파일 언어 vs 인터프리터 언어
  - 정적 언어 vs 동적 언어

- 프로그래밍 언어란 컴퓨터와 의사소통을 하기 위해 만든 인공적인 언어이다.
- 컴퓨터는 2진수로 이루어져 있기 때문에 인간이 2진수를 이용해 코딩하는 것은 어렵다.
- 따라서 인간의 언어를 컴퓨터가 알아 들을 수 있게 바꿔주는 프로그래밍 언어가 만들어지고 진화했다.

- 저수준 언어 vs 고수준 언어
  - 저수준 언어는 컴퓨터에 가까운 언어이다. 즉 2진수 체계에 더욱 가깝다면 저수준(low-level) 언어라고 한다.
  - 반대로 사람의 언어와 비슷할수록 고수준(high-level) 언어라고 한다.
  - 보통 저수준 언어에는 기계어(machine language)와 어셈블리어(assembly language)가 있다.
  - 현재 우리가 사용하는 대부분의 언어가 고수준 언어이다.

- 절차 지향 , 객체 지향, 함수 지향
  - 절차적 프로그래밍(Procedure Programming)는 프로그램의 작업이 프로그래머가 정한 순서에 따라서만 진행하는 언어이다.
  - 절차적 프로그래밍은 순서가 엄격히 지켜진다. 따라서 한 부분이 잘못되면 전체 시스템이 작동하지 않는다.
  - 순서를 엄격히 지켜야하기 때문에 순서를 정확히 짜려는 노력이 필요하다. 따라서 프로그램의 규모가 커지면 개발 난이도가 상승하고 유지보수가 힘들다는 단점이 있다.
  - 절차적 언어에는 C, 알골(ALGOL), 포트란(Fortran) 등이 있다.

  - 객체지향 프로그래밍(Obejct-Oriented Progamming)은 순서보다 기능을 중요시한다. 특정 기능을 수행하는 대상 사이의 관계가 중요하다.
  - 객체, 클래스, 인터페이스 등이 존재하고 캡슐화, 다형성, 추상화 등의 특징이 있다.
    - 더욱 자세한 내용은 다루지 않는다.
  - 규모가 큰 소프트웨어를 개발하고 관리하기 편하다. 하지만 구조가 복잡하다는 단점이 있고 객체들 사이의 의존성이 높아지는 점, 객체들끼리 어느 시점에 어느 객체를 호출하는지 알기 어렵다는 점 등과 같은 단점이 존재한다.
  - JAVA, C++, C#, Python 등이 있다.

  - 함수형 프로그래밍이란 순수 함수로 코드를 작성해서 프로그래밍의 부작용을 줄이는 프로그래밍 기법이다.
    - 순수 함수란 부작용이 없는 함수. 즉, 동일한 입력값에 대해 동일한 결과값을 보이는 함수이다.
  - 함수형 프로그래밍은 객체 내에서 상태 변화가 없는 것을 목표로 한다.
  - 멀티스레드 환경에서 강점을 보이기 때문에 대규모 개발에 적합하다.
  - 스칼라(Scala), 하스켈(Haskell), 클로저(Clojure) 등이 있다.

- 명령형 언어, 선언형 언어

    - 명령형 프로그래밍은 무엇을 어떻게 할 것인가에 대한 언어이다. 문제를 어떤 방법을 이용하여 해결하는지에 대해 초점을 맞춘다.
    - 선언형 프로그래밍은 무엇에 초점을 맞춘다. 어떻게는 이미 알고 있다.
    - 명령형은 방법을 명시하고 목표를 명시하지 않는다. 반대로 선언형은 목표륾 명시하고 방법을 명시하지 않는다.
    - 명령형 언어에는 C, ALGOL, Fortran 등이 있고 선언형 언어에는 HTML, XML 등이 있다.

- 컴파일 언어, 인터프리터 언어
    - 컴파일러와 인터프리터는 고수준의 언어를 저수준의 언어로 번역해주는 역할을 한다.
    - 두 언어의 차이점은 번역 과정에 있다. 대표적으로 컴파일 언어에는 빌드 과정이 존재하고 인터프리터 언어에는 빌드 과정이 존재하지 않는다.
    - 빌드 과정이란 프로그램을 구성하는 여러 소스 코드 파일을 한곳에 모아 목적 파일(Object File)을 만드는 과정이다. 목적 파일은 CPU가 이해할 수 있는 바이너리 코드 또는 바이트 코드로 이루어진 파일이다.
    - 컴파일러는 빌드 과정을 통해 미리 프로그램을 1회 수행하기 때문에 코드 실행 속도가 빠르다.
    - 인터프리터 언어는 빌드 과정이 없이 코드가 한줄씩 진행하면서 번역된다. 따라서 코드 실행 속도가 컴파일러에 비해 느리다. 하지만 빌드 과정이 없기 때문에 개발 속도가 빠르다.
    - 컴파일 언어에는 C, C++ 등이 있고 인터프리터 언어에는 Python, Javascript 등이 있다.

- 정적 언어, 동적 언어
    - 정적 언어와 동적 언어의 차이점은 변수의 자료형을 컴파일 과정에서 정하는 것과 실행 과정에서 정하는 것이다.
    - 정적 언어는 컴파일 과정에서 자료형이 결정된다. 따라서 변수를 선언할 때 반드시 변수의 자료형을 지정해야 한다.

    - 반면 동적 언어는 실행 과정에서 자료형이 결정된다. 따라서 변수의 자료형을 선언할 필요가 없다.
    - 정적 언어는 C, C++, JAVA 등이 있다. 동적 언어는 Javascript, Python, Ruby, PHP 등이 있다.

파이썬 언어 특징

- 파이썬은 컴파일 없이 인터프리터를 사용하는 스크립트(Script) 언어아다, → 결과를 빠르게 확인하고 빠르게 수정 가능하지만 느린 수행 속도 존재.
- 파이썬은 동적 타입이다. 즉 자료형을 명시하지 않아도 된다. 자료형 변환 시 번거로운 과정을 거치지 않아도 된다는 장점이 있지만 코드 실행 도중 에상하지 못한 타입으로 인한 에러가 발생할 수 있다.
- 파이썬은 플랫폼 독립적(Platform-Independent이다. 파이썬은 리눅스, 유닉스, 윈도우, 맥 등 다양한 운영체제에서 모두 동작한다. 운영체제별로 컴파일 할 필요가 없이 한번 소스코드를 작성하면 바로 실행 가능하다.
- 파이썬은 간결하고 쉬운 문법이다. 문법이 쉽고 간결하여 코딩이 쉽다.
    - 코딩 테스트에서 파이썬을 가장 많이 사용한다.
- 파이썬은 높은 확장성과 이식성을 가진다. 다른 언어나 라이브러리에 쉽게 접근해서 연동 가능하다.
- 파이썬은 활발한 생태계이다. 수많은 라이브러리가 존재하고 많은 프로그래머들이 개발중이다.

변수의 이해와 생성 과정

- 프로그래밍에서 변수는 변하는 값이다. 조금 더 정확하게 말하면 하나의 값을 저장하기 위해 확보한 메모리 공간 자체 또는 그 메모리 공간을 식별하기 위해 붙인 이름이다.
- 실제로 값을 저장하기 위해서는 물리적인 저장 공간이 필요하다. 보통 메모리(RAM)에 저장한다.
- 메모리르 직접 건드리면 시스템의 치명적 오류를 일으킬 수 있다. 또한 메모리 공간의 주소는 계속 바뀌기 때문에 재사용이 힘들다.

- 따라서 변수를 이용하여 값을 저장할 수 있는 메모리 공간을 확보하고 식별할 수 있다.

- 변수는 크게 선언, 초기화, 할당 단계가 있다.
- 변수의 선언 단계에서는 값을 저장하기 위한 메모리 공간을 확보한다. 또한 저장한 메모리를 식별하기 위한 변수 이름을 설정하는 것까지 변수 선언 단계이다.
- 변수의 초기화 단계에서는 변수 선언 단계를 거쳐 메모리 공간을 확보한 후에는 해당하는 메모리 공간에 undefined를 할당하여 초기화한다.
- 변수의 할당 단계에서는 초기화 단계를 거치고 undefined된 공간에 원하는 값을 재할당한다.