파이썬 이란

- 1990년 암스테르담의 귀도 반 로섬(Guido Van Rossum)이 개발한 인터프리터 언어
- 컴퓨터 프로그래밍 교육을 위해 많이 사용하지만, 기업의 실무를 위해서도 많이 사용하는 언어. 구글에서 만든 소프트웨어의 50%이상이 파이썬으로 작성

파이썬의 특징

- 파이썬은 인간다운 언어이다
- 파이썬은 문법이 쉬워 빠르게 배울 수 있다
- 파이썬은 무료이지만 강력하다
- 파이썬은 간결하다
- 파이썬은 프로그래밍을 즐기게 해준다
- 파이썬은 개발 속도가 빠르다

파이썬으로 할 수 있는 일

- 웹 개발: Django, Flask 등의 프레임워크를 사용하여 웹 애플리케이션을 개발할 수 있습니다.
- 데이터 분석: Pandas, NumPy, SciPy와 같은 라이브러리를 사용하여 데이터를 분석하고 처리할 수 있습니다.
- 머신 러닝과 인공지능: TensorFlow, PyTorch, Scikit-learn 등의 라이브러리를 활용하여 머신 러닝 모델을 구축하고 훈련시킬 수 있습니다.
- 자동화: 파이썬 스크립트를 작성하여 일상적인 작업을 자동화하고, 시스템 관리 작업을 수행할 수 있습니다.
- 게임 개발: Pygame과 같은 라이브러리를 사용하여 간단한 게임을 개발할 수 있습니다.
- 모바일 애플리케이션 개발: Kivy 또는 BeeWare와 같은 라이브러리를 사용하여 모바일 애플리케이션을 개발할 수 있습니다.
- 데스크탑 애플리케이션 개발: PyQt, Tkinter 등의 라이브러리를 활용하여 데스크탑 애플리케이션을 개발할 수 있습니다.
- 시스템 스크립팅과 네트워킹: 시스템 유틸리티를 개발하거나 네트워크 프로토콜을 구현할 수 있습니다.
- 임베디드 시스템과 하드웨어 제어: 라즈베리 파이와 같은 임베디드 시스템을 제어하고 하드웨어를 프로그래밍할 수 있습니다.
- 사이언티픽 컴퓨팅: 과학적 연산과 시뮬레이션을 위해 파이썬을 활용할 수 있습니다.
- 교육: 파이썬은 초보자에게 프로그래밍을 가르치는 데 이상적인 언어로 평가받고 있습니다.
- 파이썬의 다양한 라이브러리와 프레임워크 덕분에, 이러한 분야에서의 작업이 더욱 쉽고 효율적으로 수행될 수 있습니다.

```
# ! 기호는 Colab 셀에서 Unix/Liunx 셀 명령어를 실행
!python --version
```

```
Python 3.10.12
```

```
# 라인 매직은 단일 라인에 대해 실행되며 % 하나를 사용하고 셀 매직은 셀 전체에 적용되며 %%를 사용
# 셀 전체에 적용시 %%를 사용함.
# 현재 작업폴더 확인
%pwd
```

```
'/content'
```

```
%%time

sum = 0
for i in range(1000):
  sum+= i

print(sum)
```

```
499500
CPU times: user 1.34 ms, sys: 0 ns, total: 1.34 ms
Wall time: 1.35 ms
```

용어

- 식별자 : 프로그램밍 언어에서 이름을 붙일 때 사용하는 단어. 주로 변수 또는 함수 이름 등으로 사용
- 주석 : 프로그램을 설명하기 위해 사용. # 기호로 주석 처리
- 연산자 : 스스로 값이 되는 것이 아니고 값과 값 사이에 무언가 기능을 적용할 때 사용
- 자료 : 리터럴이라고 하는데 숫자이든 문자이든 어떠한 값 자체를 의미. 1, 10, "Hello"
- 키워드 : 파이썬이 만들어질 때 이미 사용하겠다고 예약해 놓는 것. False, None, True, ...
- 프로그램밍 언어에서 사용자가 이름을 정할 때 키워드는 사용할 수 없음

식별자

count, user_name, _is_valid, calculate_area, Car, model, year, math 및 m 모두 유효한 식별자. 각각의 식별자는 특정한 데이터 또는 기능에 이름을 부여하여 코드 내에서 해당 데이터나 기능을 참조할 수 있게하며 코드의 가독성과 유지 보수성을 높이는 데 중요한 역할

```
# 변수 식별자
count = 10
user_name = 'Hyun'
_is_vaild = True

# 함수 식별자
def calculate_area(radius):
  return 3.14159 * radius * radius

# 클래스 식별자
class Car:
    def __init__(self, model, year):
        self.model = model
        self.year = year

# 모듈 식별자
import math as m
```

식별자 기본 규칙

- 키워드를 사용하면 안된다
- 특수문자는 언더 바만 사용
- 숫자로 시작하면 안된다.
- 공백을 포함할 수 없다.

```
import keyword
print(keyword.kwlist, 'Wn')
len(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally

35
```

```
# Q. 주어진 문자열 리스트에서 유효한 Python 변수 이름만을 반환하는 함수를 작성하시오.
# result1 = text1.isalnum()
identifiers = ["var1", "2things", "variable_name", "time"]

def func1(x):
    return x.isalnum()

results = list(filter(func1, identifiers))
print(results)
```

```
['var1', '2things', 'time']
```

```
identifiers = ["var1", "2things", "variable_name", "time"]

import re

def check_integer_before_string(s):

    match = re.match(r'^(\d+)', s)
    # 문자열에 대해 정규식 패턴을 일치하려고함.
    # s. 패턴은 문자열( )의 시작 부분에 있는 r'^(\d+)'하나 이상의 숫자( )와 일치함.
    # 괄호는 일치하는 숫자를 캡처하는 캡처 그룹을 만드는 데 사용되며, 이 작업의 결과는 변수에 저장됨.

    if match:
      # 정규식 패턴과 일치하였는지 확인함.

        num_length = len(match.group(1))
        if len(s) > num_length and s[num_length].isalpha():
          # 1. 입력한 문자열의 길이가 s(num_length) 숫자의 길이보다 긴지 확인
          # 2. 숫자 바로 뒤의 문자 s가 알파벳 문자인지 확인
            return False
    return True
# 두 조건이 모두 True 이면 숫자 뒤에 문자가 있고 숫자 뒤에 첫 번째 문자가 문자라는 의미
# 19번 열에 있는 return False의 조건이 충족되면 함수는 False 입력한 문자열을 만족하지 않아 반환됨.
# 20번 열에 있는 return True 이전에 있는 조건 중 하나라도 충족되지 않거나 일치 항목이 하나도 없는 경우
# 이것들을 반환하여 True 입력하며, 문자열이 기준을 충족함을 나타냄
result = list(filter(check_integer_before_string, identifiers))
print(result)

result = check_integer_before_string("123abc")
print(result)  # False
```