(주)오파스넷

Python Programming Fundamentals

FOR BEGGINERS

비즈니스전략본부 > S/W 개발팀

January 28, 2022

Contents

01
Chapter

Python Language & Development Tools

파이썬과 개발 툴

파이썬

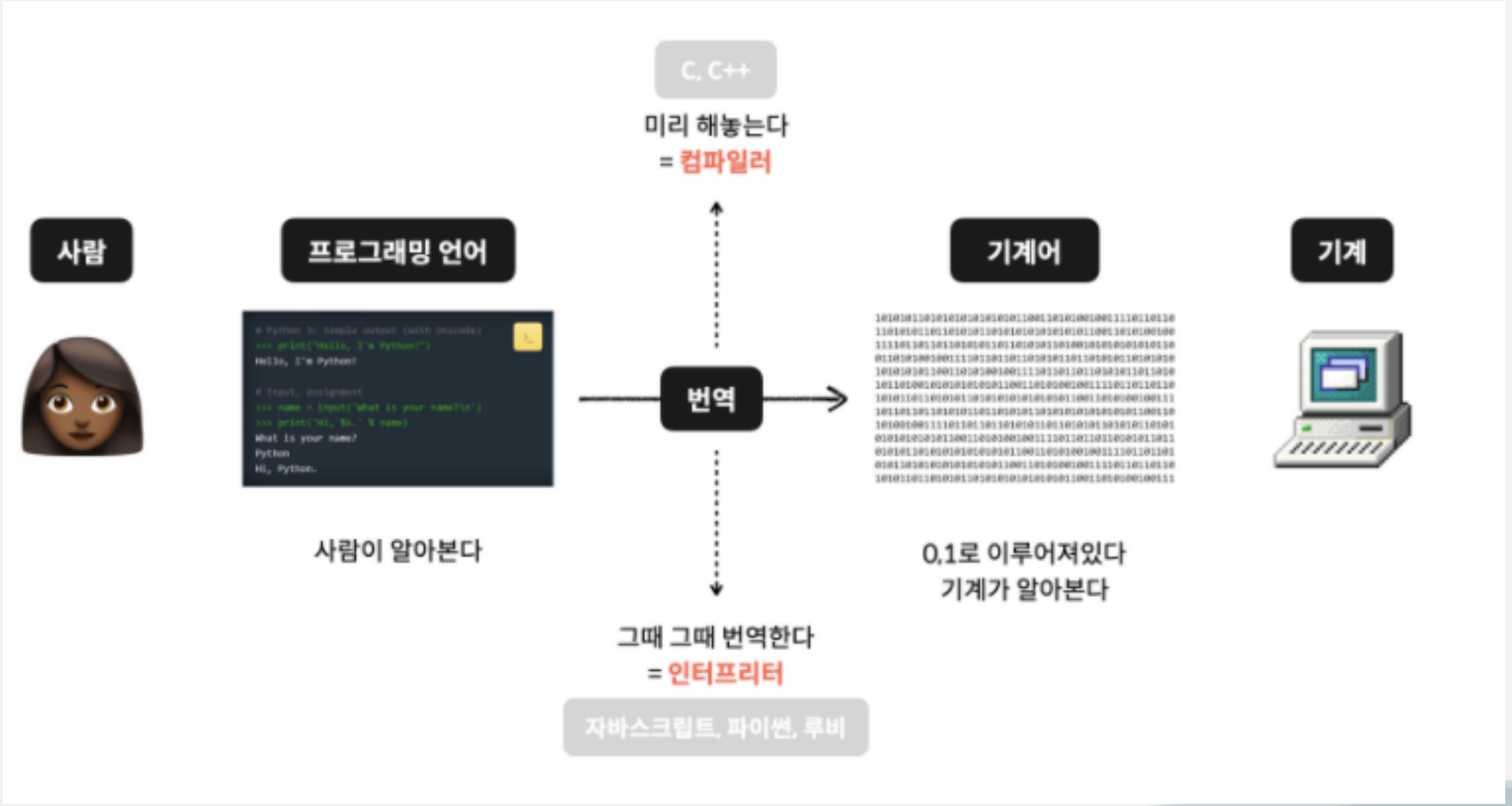
❖ 1991년 귀도 반 로썸(Guido van Rossum)이 발표한 인터프리터 언어

- ✓ 프로그래밍 언어의 소스 코드를 바로 실행하는 컴퓨터 프로그램 또는 환경
- ✓ 2000년에 python2, 2008년에 python3 발표

❖ 현재 전 세계에서 많이 사용되는 프로그래밍 언어 중 하나

- ✓ 구글에서 만들어진 소프트웨어의 50% 이상 파이썬 사용
- ✓ 드롭박스, 인스타그램
- ✓ 이해하기가 쉬워 공동작업과 유지보수가 편함

인터프리터 언어 vs 컴파일 언어



인터프리터 언어 vs 컴파일 언어

❖ 인터프리터 언어 (Interperter Language)

✓ 인터프리터로 소스 코드를 한줄 한줄 읽어 바로 실행하는 방식으로 동작하는 언어, 스크립트 언어라고도 한다.
✓ 컴파일을 하지 않고 바로 실행한다는 특징이 있지만, 소스 코드를 읽으며 실행하기 때문에 프로그램의 실행 시간은 느리다.
✓ 소프트웨어의 동작 내용을 대본(스크립트)로 보고 제어하는 언어

e.g.) Python, R, Ruby, php, Javascript

❖ 컴파일 언어 (Compiled Language)

✓ 소스 코드를 컴파일한 후 기계어를 CPU/메모리를 통해 읽어 실행하는 방식으로 동작하는 언어
✓ 컴파일을 하기 때문에 규모가 큰 프로그램이라면 컴파일 시간이 오래 걸릴 수 있지만, 컴파일 후의 기계어를 통해 프로그램을 실행하기 때문에 실행 시간은 빠르다.

e.g.) C, C++, Java

인터프리터 언어 vs 컴파일 언어

Python

helloworld.py

```
#!/usr/bin/env python

print('Hello World!')
```

실행
python helloworld.py
Hello World!

php

helloworld.php

```
<?php

echo "Hello World!"

?>
```

실행
php helloworld.php
Hello World!

java

helloworld.java

```
public class Helloworld {
    public static void main(String args[]) {
        System.out.println("Hello World!");
    }
}
```

컴파일(linux)
javac java_helloworld.java

실행
java helloworld
Hello World!

C++

helloworld.cpp

```
#include<iostream>
using namespace std;

int main()
    {
        cout << "Hello World!" << endl;
        return 0;
    }
```

컴파일(linux)
g++ -o helloworld helloworld.cpp

실행
./helloworld
Hello World!

파이썬의 특징

1. 직관적이고 쉽다
아주 간단한 영어 문장을 읽듯이 보고 쉽게 이해할 수 있도록 구성

2. 널리쓰인다
구글, 아마존, 핀터레스트, 인스타그램, IBM, 디즈니, 야후, 유튜브, 노키아, NASA 등과 네이버, 카카오톡의 주력 언어 중 하나

3. 개발환경이 좋다
빅 데이터 분석, 인공지능, 수치해석, 웹, 게임 등 개발에 필요한 다양한 라이브러리 제공, 온라인 커뮤니티 활성화

4. 강력하다
이미지 처리, 웹 서버, 알고리즘 처리, 빅 데이터 처리 등의 난이도 높은 프로그램을 쉽고 빠르게 개발 가능

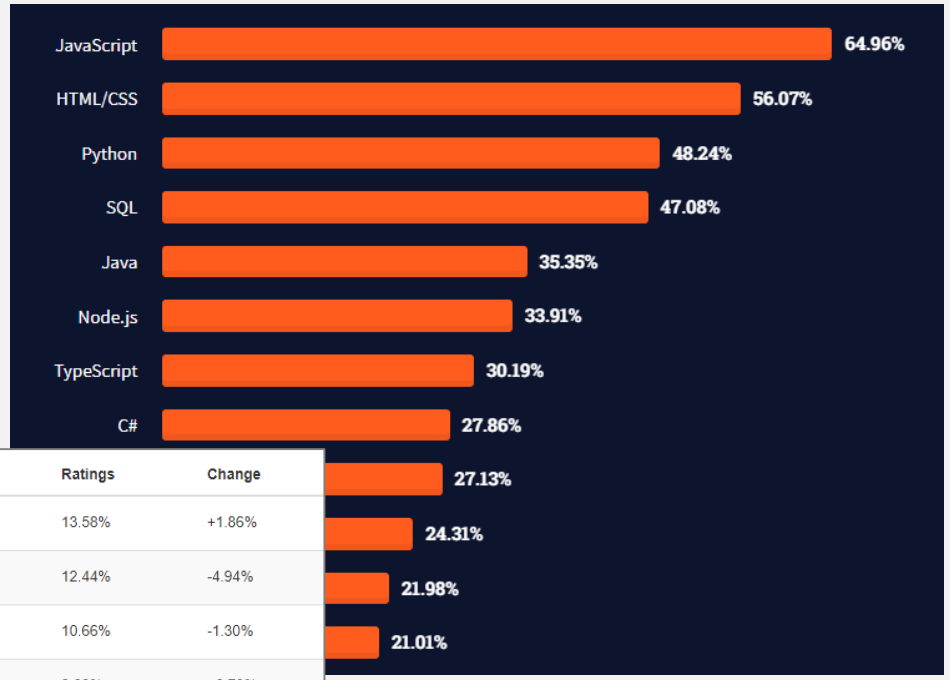
세계 시장에서 컴퓨터 언어 점유율

https://pypl.github.io/PYPL.html

Worldwide, Jan 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	28.74 %	-1.8 %
2		Java	18.01 %	+1.2 %
3		JavaScript	9.07 %	+0.6 %
4	↑	C/C++	7.4 %	+1.1 %
5	↓	C#	7.27 %	+0.7 %
6		PHP	6.06 %	+0.0 %
7		R		
8		Objective-C		
9		Swift		
10		TypeScript		

https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language



Jan 2022	Jan 2021	Change	Programming Language	Ratings	Change
1	3	↑	Python	13.58%	+1.86%
2	1	↓	C	12.44%	-4.94%
3	2	↓	Java	10.66%	-1.30%
4	4		C++	8.29%	+0.73%
5	5		C#	5.68%	+1.73%
6	6		Visual Basic	4.74%	+0.90%
7	7		JavaScript	2.09%	-0.11%
8	11	↑	Assembly language	1.85%	+0.21%
9	12	↑	SQL	1.80%	+0.19%
10	13	↑	Swift	1.41%	-0.02%

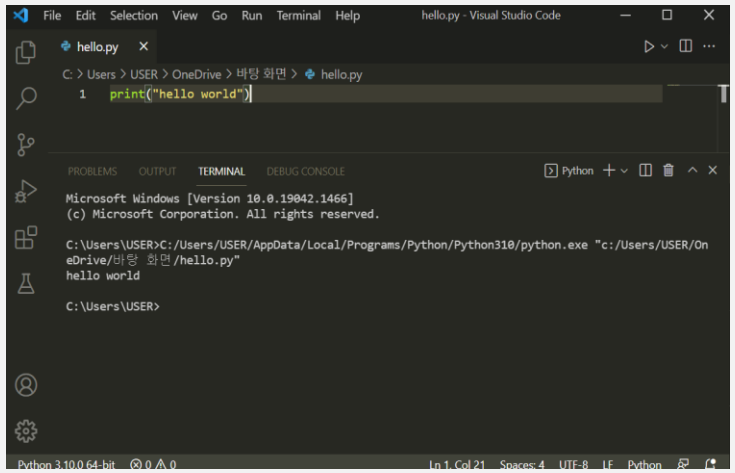
https://www.tiobe.com/tiobe-index/

프로그램 개발 툴

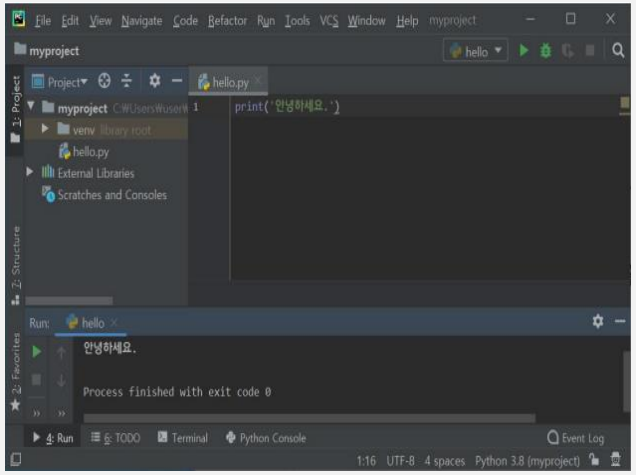
❖ IDE (Integrated Development Environment)

- ✓ 개발을 하면서 사용되는 도구들의 집합
- ✓ 코딩, 디버그, 컴파일, 배포 등 프로그램 개발에 관련된 모든 작업을 하나의 프로그램안에서 처리하는 환경을 제공하는 소프트웨어

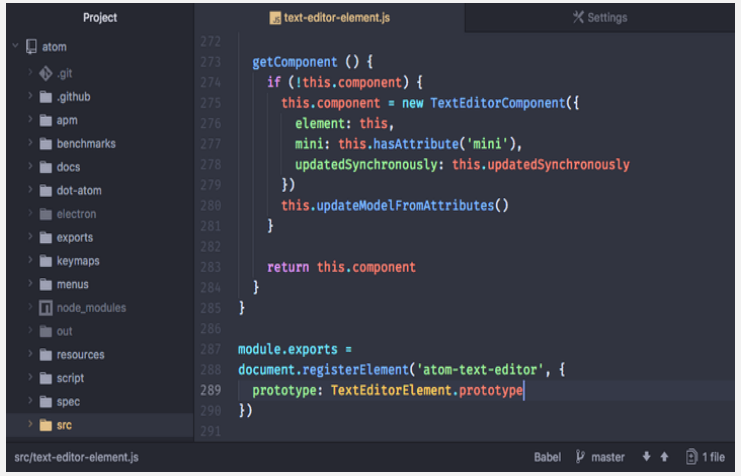
Visual Studio Code



PyCharm

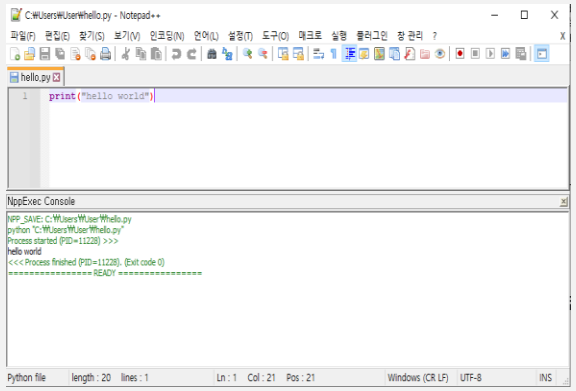


Atom

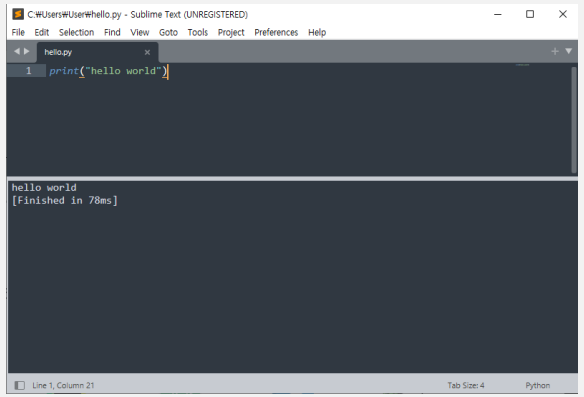


프로그램 개발 툴

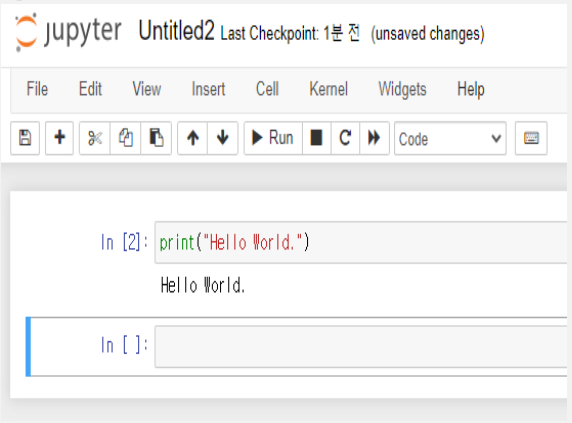
Notepad++



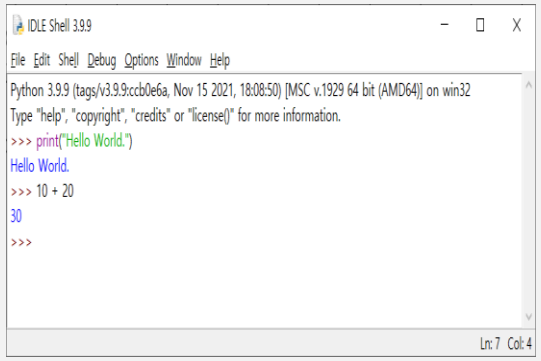
Sublime text



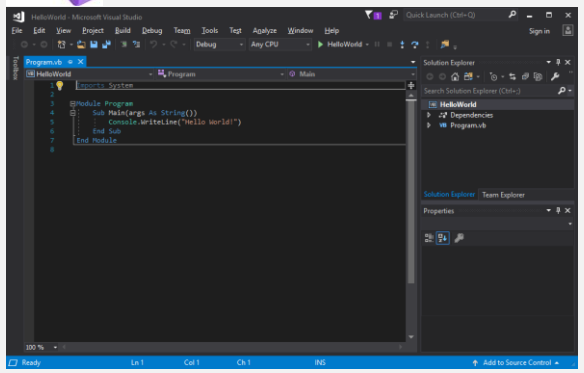
Jupyter Notebook



Python IDE



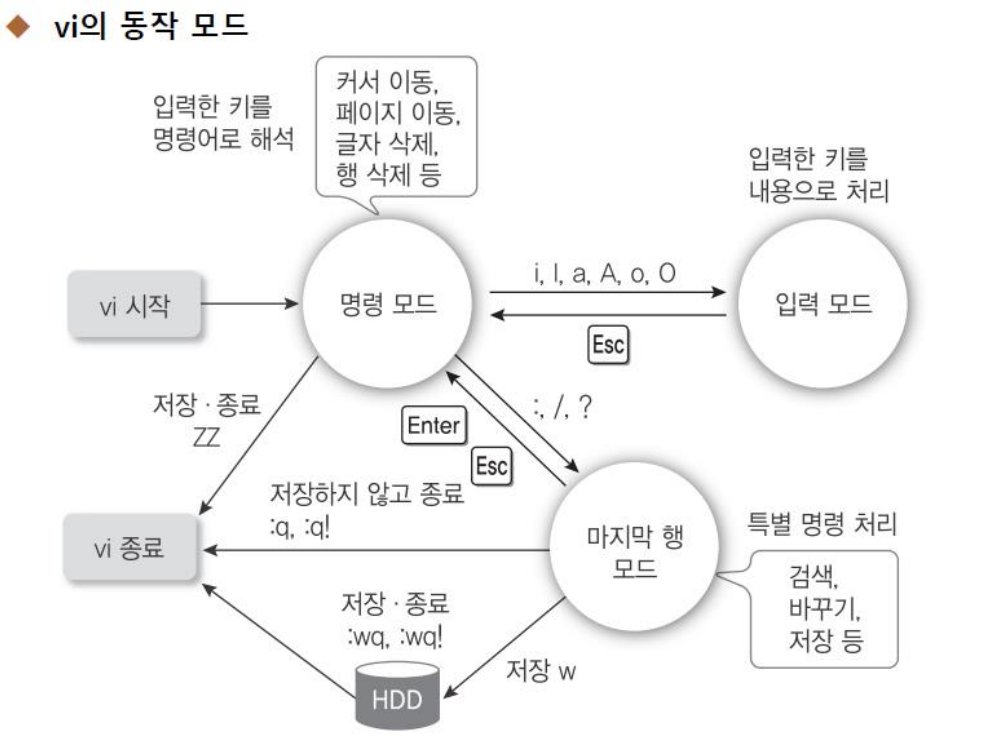
Visual Studio



Vi 편집기

❖ Vi 편집기

- ✓ 유닉스 계열 운영체제에서 주로 쓰이는 오픈소스 문서 편집기
- ✓ CLI 환경으로 단축키, 명령어 등을 이용해 텍스트를 편집한다.



Vi 편집기

❖ Vi 편집기 시작하기

- ✓ vi 파일명
- ✓ 해당 파일이 있으면 파일의 내용이 보이고, 없는 파일이면 빈 파일이 열린다.

❖ Vi 편집기 종료하기

- ✓ 명령모드나 마지막행 모드에서 저장하고 종료 가능

구분	명령키	기능
마지막행 모드	:q	vi에서 작업한 것이 없을 때 그냥 종료한다.
	q!	작업한 내용을 저장하지 않고 종료한다.
	:w [파일명]	작업한 내용을 저장만 한다. 파일명을 지정하면 새 파일로 저장한다.
	:wq , :wq!	작업한 내용을 저장하고 vi를 종료한다.
명령모드	shift+ZZ	작업한 내용을 저장하고 vi를 종료한다.

02
Chapter

Python Syntax

파이썬의 기본 문법

변수(variable)

- ✓ 메모리에 데이터를 저장하는 데 사용되는 공간의 이름
- ✓ 변수에는 정수, 실수, 문자열 등 다양한 값 할당 가능
- ✓ 파이썬에서 =는 할당을 의미

```
>>> a = 2020
>>> b = 3030
>>> c = a + b
>>> a
5050

>>> var1 = "Foo"
>>> VAR1 = "Bar"
>>> new_string = var1 + VAR1
>>> new_string
'FooBar'
```

변수명 규칙

- 변수명은 영문 대소문자, 밑줄(_), 숫자를 조합해서 사용
- 특수문자(!@#$%^&*)나 공백 ' '은 변수명에 사용할 수 없음
- 변수명은 숫자로 시작하면 안됨
  - ✓ a, b, friends, grunt, var3, _attr, is4later (O)
  - ✓ 2_Variable, hello~, hi! (X)
- 영문에서 대문자와 소문자는 서로 다름
- 예약어는 변수명으로 사용할 수 없음
  - ✓ and, as, assert, break, class, continue, def, del, elif, else, except, is, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield

파이썬의 자료형

❖ 파이썬에서는 다양한 자료형과 그에 따른 연산자를 기본으로 제공

- ✓ Numeric
- ✓ String
- ✓ Boolean
- ✓ List
- ✓ Tuple
- ✓ Dictionary

❖ type()

- ✓ 인자로 전달한 자료형이 무엇인지 반환하는 함수
- ✓ 각 자료형만의 함수가 존재하기도 함

숫자형(Numeric)

- ✓ 숫자로 이루어진 자료형
- ✓ 정수나 실수를 다룰 수 있음
- ✓ 정수형(Integer): 음수, 0, 양수로 구성된 숫자
- ✓ 실수형(Floating Point, 부동 소수점): 소수점이 있는 숫자

```
>>> type(3)
<class 'int'>

>>> type(-128)
<class 'int'>

>>> type(1.4)
<class 'float'>
```

문자열(String)

- ✓ 하나 또는 여러 개의 문자로 구성
- ✓ 작은따옴표(‘’) 또는 큰따옴표(“”)로 구분
- ✓ 부호는 반드시 쌍을 이뤄야함

```
>>> type('Hello')
<class 'str'>

>>> type("World")
<class 'str'>

>>> a = "1"
>>> type(a)
<class 'str'>

>>> b = int(a)
>>>type(b)
<class 'int'>
```

문자열 슬라이싱

✓ 인덱스(Index)는 문자열 요소의 위치를 가리킴

✓ 0부터 시작

```
>>> a = "Life is too short, You need Python"
>>> print(a[:])
Life is too short, You need Python
>>> print(a[0])
L
>>> print(a[0:4])
Life
>>>print(a[:17])
Life is too short
>>>print(a[19:])
You need Python
```

Boolean

- ✓ 참, 거짓을 나타내는 자료형
- ✓ True, False로 나타낼 수 있음
- ✓ 논리연산이나, 수치 간의 비교연산의 결과로 사용

```
>>> a = 1
>>> b = 2
>>> a > b
False

>>> c = a > b
>>> type(c)
<class 'bool'>
```

리스트(List)

- ✓ 여러 값을 함께 모아서 저장한 것
- ✓ 순서가 존재하고, 여러 종류의 값을 담을 수 있음
- ✓ 값을 변경 할 수 있음.
- ✓ [요소1, 요소2, …]

```
>>> mylist = [1, 2, 'a', 'b']
>>> mylist
[1, 2, 'a', 'b']

>>> mylist[0]
1

>>>mylist[3] = 3
>>> mylist
[1, 2, 'a', 3]
```

튜플(Tuple)

- ✓ 여러 값을 함께 모아서 저장한 것
- ✓ 순서가 존재하고, 여러 종류의 값을 담을 수 있음
- ✓ 값을 변경 할 수 없음.
- ✓ (요소1, 요소2, ...)

```
>>> mytuple1 = (1, 2, 'a', 'b')
>>> mytuple
(1, 2, 'a', 'b')

>>> mytuple2 = 1, 2, 3
>>> type(mytuple)
<class 'tuple'>

>>>mytuple[3] = 3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

딕셔너리(Dictionary)

- ✓ 관련된 정보를 서로 연관시켜 놓은 것
- ✓ 키(key)와 값(value)을 쌍으로 가짐
- ✓ 키를 이용해 값을 가져올 수 있고, 인덱스는 지원하지 않음
- ✓ {key1: value1, key2: value2, ...}

```
>>> mydict = {'name': 'Tom', 'phone':' 010-1234-5678', 'birth': 0126}
>>> mydict
{'name': 'Tom', 'phone':' 010-1234-5678', 'birth': 0126}

>>> mydict['name']
'Tom'

>>>mydict['email'] = 'tom@opasnet.co.kr'
>>> mydict
{'name': 'Tom', 'phone':' 010-1234-5678', 'birth': 0126, 'email': 'tom@opasnet.co.kr'}
```

숫자 연산과 문자열 연산

```
>>> a = 2020
>>> b = 3030
>>> c = a + b
>>> a
5050

>>> name = 'Tom'
>>> greet = 'Welcome ' + name + '!'
>>> print(greet)
Welcome Tom!
```

산술연산자	설명
+	더하기
-	빼기
*	곱하기
/	나누기
%	나머지
//	몫
**	거듭 제곱

사용자 입력 함수 - input

- ✓ 사용자가 입력한 값을 변수에 넣고 싶을 때 사용
- ✓ input 괄호 안에 질문을 입력하여 프롬프트를 띄울 수 있다.
- ✓ input은 입력되는 모든 것을 문자열로 취급한다.

```
>>> a = input()
Hello world
>>> a
'Hello world'

>>> num1 = input('Enter a Number: ')
Enter a Number: 1
>>>type(num1)
<class 'str'>
```

출력 함수 - print

- ✓ 자료형을 출력할 때 사용
- ✓ 여러 값을 전달할 수 있다.
- ✓ 콤마를 사용하면 문자열 사이에 띄어쓰기를 할 수 있다.
- ✓ sep 파라미터는 구분자(기본값 공백), end 파라미터는 끝문자(기본값 개행)를 지정할 수 있다.

```
>>> year = 2021
>>> month = 12
>>> day = 5
>>> print(year, month, day)
2021 12 5

>>> print(year, month, daysep='/')
2021/12/5
```

03
Chapter

Control Statements in Pyhton

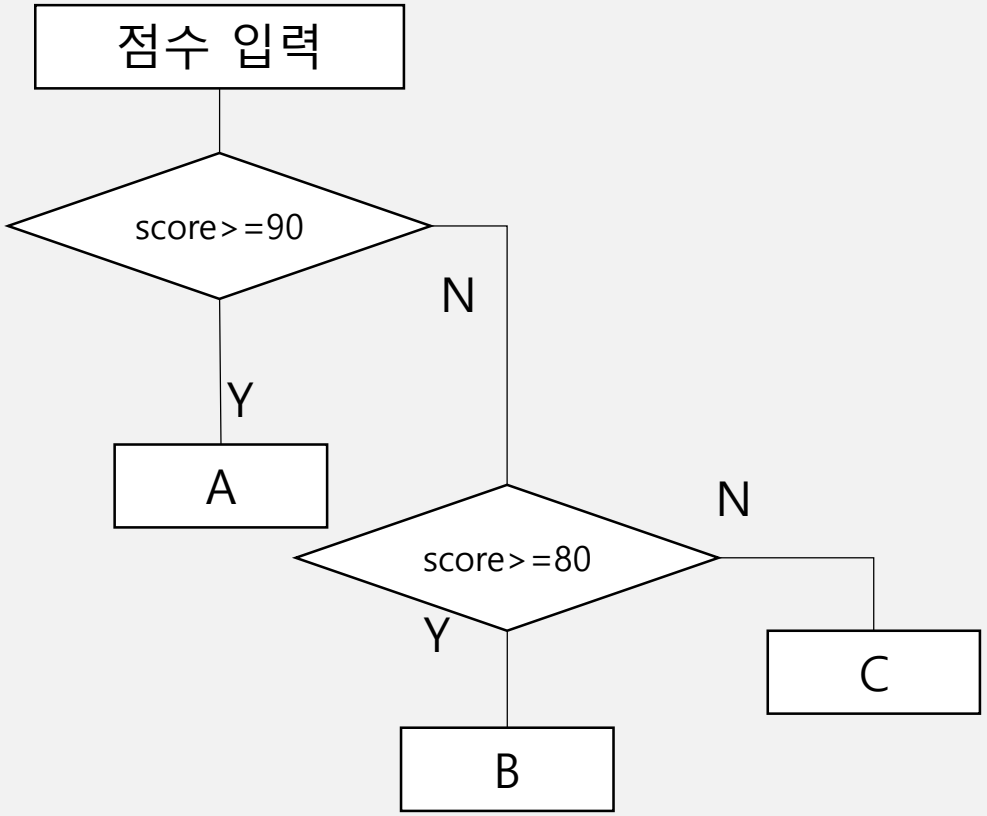
제어문

조건문 if

- ✓ '만약 ~하면 ~ 하다'와 같은 상황에서 사용
- ✓ 조건이 참인지 거짓인지 판단해서 명령을 실행
- ✓ 들여쓰기에 주의한다. (띄어쓰기 4칸)

```
>>> score = 85

>>> if score >= 90:
...     print("A")
... elif score >= 80:
...     print("B")
... else:
...     print("C")
...
B
```



조건문 if

❖ 짝수/ 홀수 판별

```
>>> num1 = 100

>>> if num1 % 2 == 0:
...     print(“짝수”)
... elif num1 % 2 == 1:
...     print(“홀수")
...
홀수
```

비교 연산자	설명
a>b	a는 b보다 크다
a<b	a는 b보다 작다
a==b	a와 b는 같다
a!=b	a와 b는 같지 않다
a>=b	a는 b보다 크거나 같다
a<=b	a는 b보다 작거나 같다

조건문 if

❖ 합격/ 불합격 판별

```
>>> score1 = int(input('필기성적을 입력하세요 : '))
>>> score2 = int(input('실기성적을 입력하세요 : '))

>>>if score1 >= 80 and score2 >= 80:
...     print('합격!')
... else:
...     print('불합격!')
...
필기성적을 입력하세요 :90
실기성적을 입력하세요 :85
합격!
```

논리 연산자	설명
조건1 and 조건2	조건1과 조건2가 둘 다 참이어야 전체 결과가 참이 된다
조건1 or 조건2	조건1과 조건2 중 하나만 참이어도 전체 결과가 참이 된다
not 조건	조건이 참이면 그 결과는 거짓, 조건이 거짓이면 그 결과는 참이된다

반복문

- ✓ 특정 코드를 반복해서 실행하는 구문
- ✓ for문과 while문이 있음
- ✓ 반복하는 부분을 구분하기 위해 콜론(:)과 들여쓰기를 사용한다.
- ✓ 들여쓰기에 주의한다. (띄어쓰기 4칸)

반복문 for문

- ✓ 컨테이너 안의 값을 전부 순회할 때 까지 반복
- ✓ 들여쓰기에 주의한다. (띄어쓰기 4칸)

```
>>> num_list = [1, 2, 3]
>>> for num in num_list:
...     print(num)
...
1
2
3

>> for i in range(0, 3):
...     print(i)
...
0
1
2
```

반복문 for문

❖ 5개의 숫자가 홀수인지 짝수인지 여부 구분하여 출력

```
>>> num_list = [1, 2, 3, 4, 5]

>>> for num in num_list:
...         If num % 2 == 0:
...             print('짝수')
...         else:
...             print('홀수')
...
홀수
짝수
홀수
짝수
홀수
```

반복문 중첩 for문

✓ 반복문을 중첩해서 사용이 가능

✓ 들여쓰기에 주의.

```
for i in range(횟수):          # 바깥쪽 루프
    for j in range(횟수):      # 안쪽 루프
        가로 처리 코드
    세로 처리 코드
```

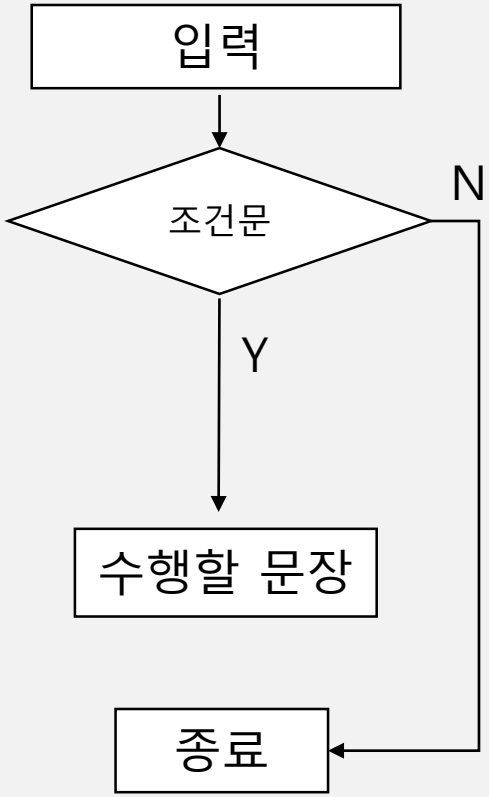
```
>>> for i in range(1, 4):
        for j in range(1, 4):
            print('i={}, j={}'.format(i, j))
...
i=1, j=1
i=1, j=2
i=1, j=3
i=2, j=1
i=2, j=2
i=2, j=3
i=3, j=1
i=3, j=2
i=3, j=3
>>>
```

반복문 while문

✓ 조건문이 참인 동안에 수행할 문장을 반복해서 수행

✓ 들여쓰기에 주의한다. (띄어쓰기 4칸)

```
>>> i = 0
>>> while i < 5:
        print("Hello")
        i = i+1
...
Hello
Hello
Hello
Hello
Hello
```



반복문 while문

❖ 1~10까지의 숫자에서 짝수만 출력

```
>>> num1 = 0
>>> while num1 <=10:
...         num1 = num1 + 1
...         if num1 % 2 == 0:
...             print(num1)
...
2
4
6
8
10
```

방법1

※ continue: 다시 조건으로 돌아간다.
break: 반복문을 종료한다.

```
>>> num1 = 0
>>> while True:
...         num1 = num1 + 1
...         if num1 % 2 == 0:
...             print(num1)
...         else:
...             continue
...         if num1 ==10:
...             break
...
2
4
6
8
10
```

방법2

04
Chapter

Python Functions

함수

함수

- ✓ 입력값을 받아서 특정 과업을 수행하여 출력값을 반환하도록 짜여진 코드블록
- ✓ 반복되는 코드에 이름을 붙여 다시 사용할 수 있게 한다
- ✓ 사용자 정의 함수와 파이썬 내장함수가 있다

```
def 함수이름(매개변수):
    실행할 명령1
    실행할 명령2
    ...
    return 결과
```

함수 정의

```
함수이름(입력값)
```

→ 함수 호출

```
>>> def add(num1, num2):
...         result = num1 + num2
...         return result
...
>>>add(3,4)
7

>>> def say_hello():
...         print("Hello")
...
>>>say_hello()
Hello
```

함수 예제

❖ 여러 개의 입력값을 받아 합을 구하는 함수 만들기

```
>>> def add_many(*args):
...         result = 0
...         for i in args:
...             result = result + i
...         return result
...
>>>
>>> add_many(1,2,3)
6
>>> add_many(1,2,3,4,5,6,7,8,9,10)
55
```

함수 예제

❖ 사용자에게 두 개의 숫자를 입력 받아 더해주는 함수

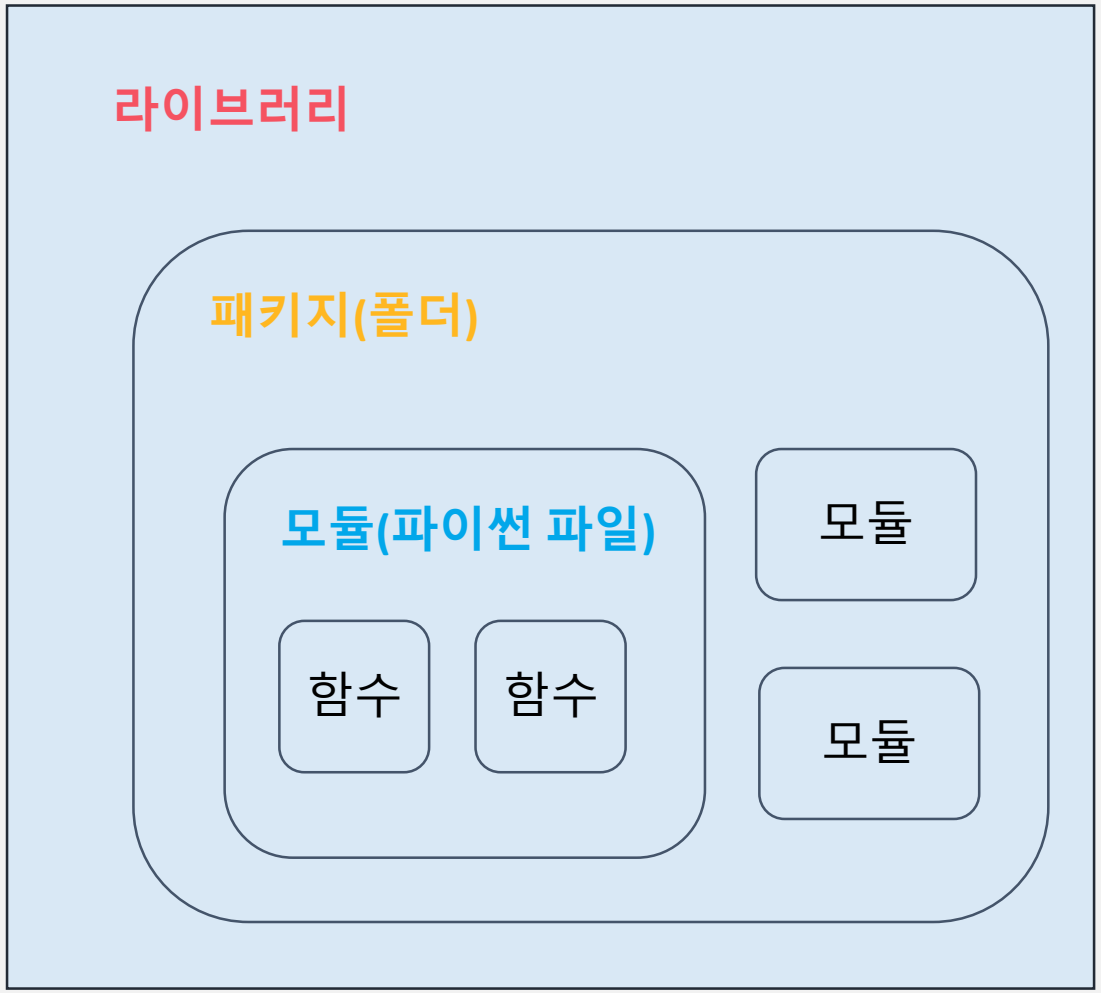
```
>>> def add():
...         num1 = input("첫번째 숫자: ")
...         num2 = input("두번째 숫자: ")
...         result = int(num1) + int(num2)
...         print(num1, '+', num2, '=', result)
...
>>>
>>> add()
첫번째 숫자: 100
두번째 숫자: 200
100 + 200 = 300
```

함수 예제

❖ 주어진 숫자가 홀수인지 짝수인지 판별해주는 함수 만들기

```
>>> def is_odd(number):
...         if number % 2 == 1:
...             return True
...         else:
...             return False
...
>>>
>>> is_odd(3)
True
>>> is_odd(4)
False
```

모듈(Module)



❖ 모듈

✓ 함수나 변수를 불러와서 사용할 수 있게 모아놓은 파이썬 파일

❖ 패키지

✓ 모듈을 폴더에 넣어둔 것

✓ Pandas, Numpy, TensorFlow 등

❖ 라이브러리

✓ 패키지를 저장해두는 곳

라이브러리(Library)

- ✓ 파이썬은 제공하는 라이브러리들이 많아서 구현할 필요 없이 바로 가져다 쓸 수 있다.
- ✓ 라이브러리에는 파이썬 설치 시 기본으로 설치되는 파이썬 표준 라이브러리와 외부 라이브러리로 나뉜다.
- ✓ 표준 라이브러리는 다른 설치 없이 import 라이브러리명 으로 사용이 가능하다.

  ex) os, sys, math, datetime, random ….
- ✓ 외부 라이브러리는 pip을 이용해 설치 후 사용이 가능하다.
- ✓ 명령 프롬프트 창에 pip install 라이브러리명으로 설치가 가능하다.

  ex) pip install pandas
- ✓ PYPI(Python Packagy Index) : 파이썬 관련 패키지들이 모여있는 저장소, 자신이 개발한 모듈을 업로드할 수 있고, 누구에게나 공개되어 있다.

  https://pypi.org/

random 라이브러리

- ✓ 난수를 만들거나 무작위와 관련된 함수를 포함한 모듈
- ✓ random.choice() : 리스트의 값 중 하나를 랜덤하게 선택
- ✓ random.sample() : 리스트의 값 중에서 지정한 개수만큼 랜덤하게 선택
- ✓ random.randint() :특정 범위의 정수 중 하나를 랜덤하게 선택

```
>>> import random
>>> fruits = ['orange', 'strawberry', 'banana']
>>> print(random.choice(fruits))
'orange'
>>> print(random.sample(fruits, 2))
['orange', 'banana']

>>> num1 = random.randint(0, 10)
>>> print(num1)
8
```

opas net
(주)오파스넷

수고하셨습니다.