

파이썬 설치 및 기초

연산자, 문자열, 리스트 I



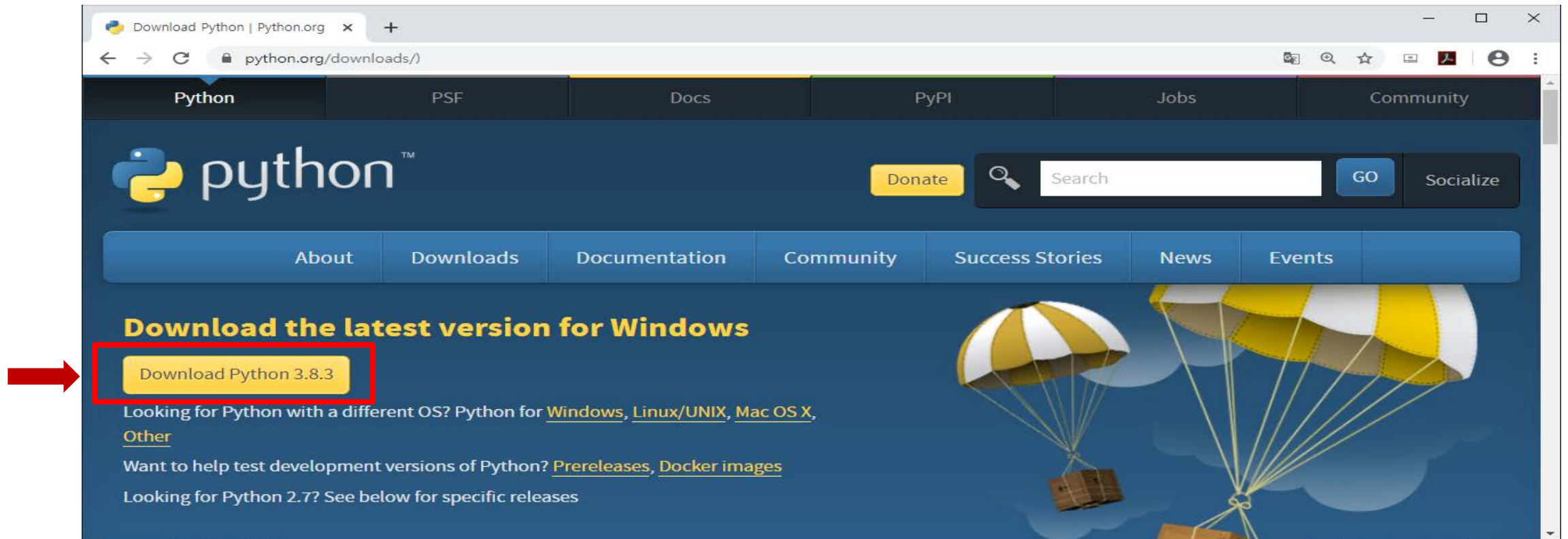
파이썬 설치(PYTHON 3.8)



윈도우에서 파이썬 설치

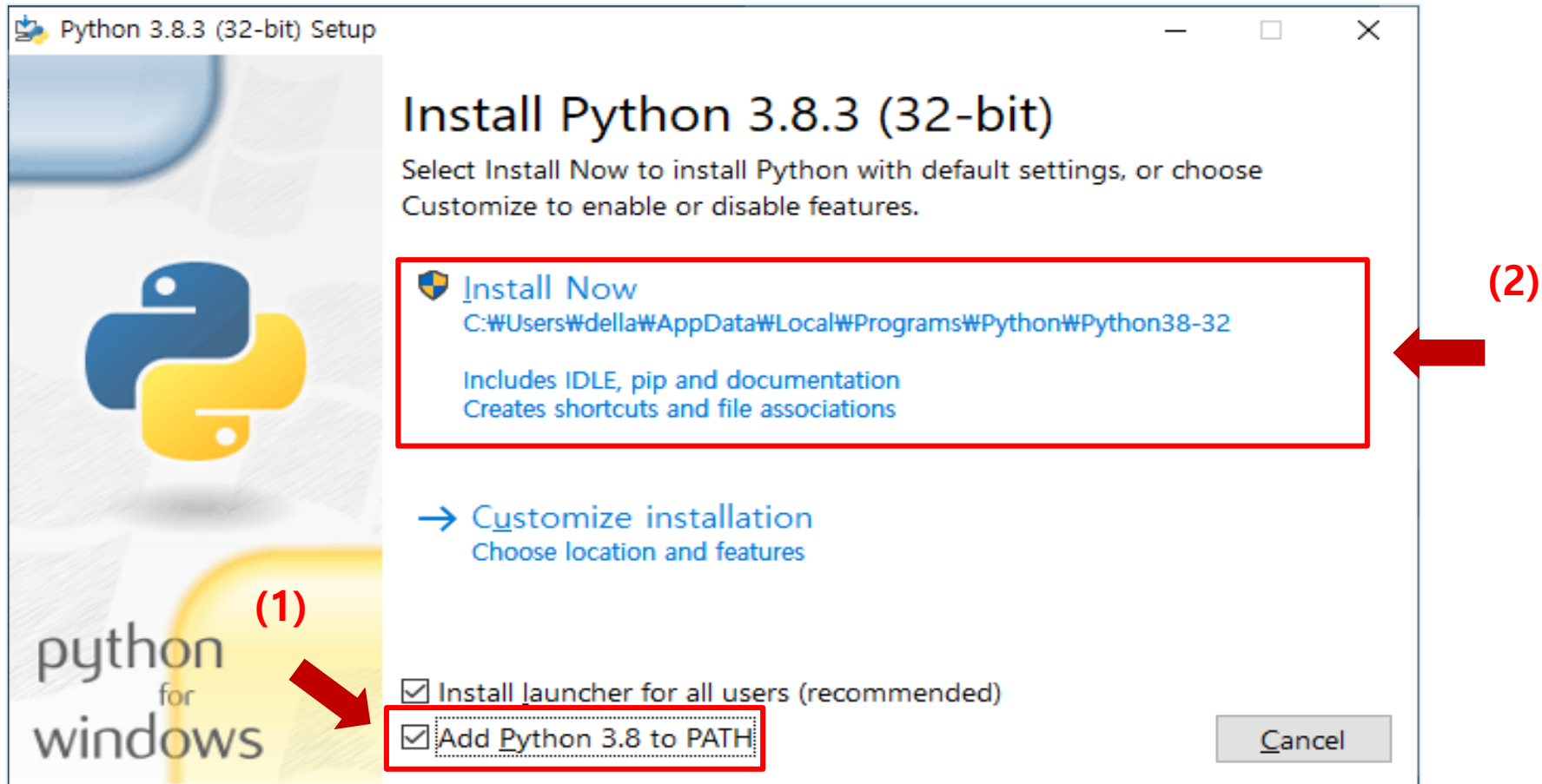
- 파이썬 공식 홈페이지의 다운로드 페이지

(<https://www.python.org/downloads/>)에서 윈도우용 언어 패키지를 다운로드 한다.



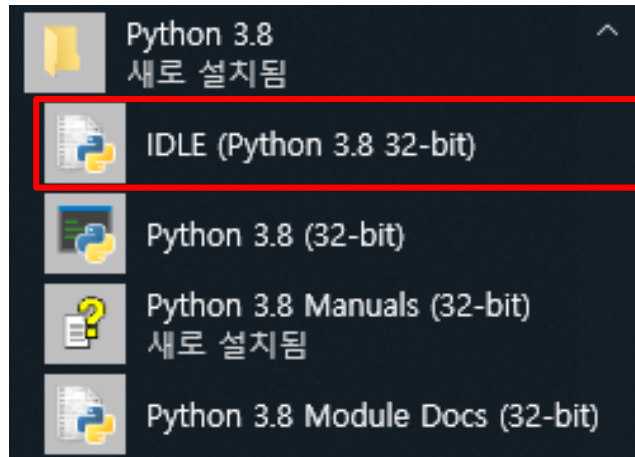
윈도우에서 파이썬 설치

- python 3.8 용 인스톨러 파일을 받아서 설치한다.



윈도우에서 파이썬 설치

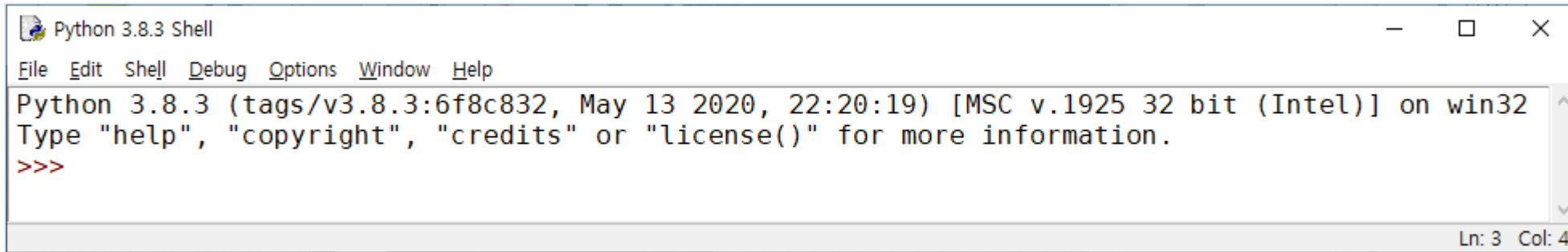
- 파이썬이 정상 설치되었다면, [시작]-[모든 프로그램]-[Python 3.8]을 확인할 수 있다.



← 선택하여 실행

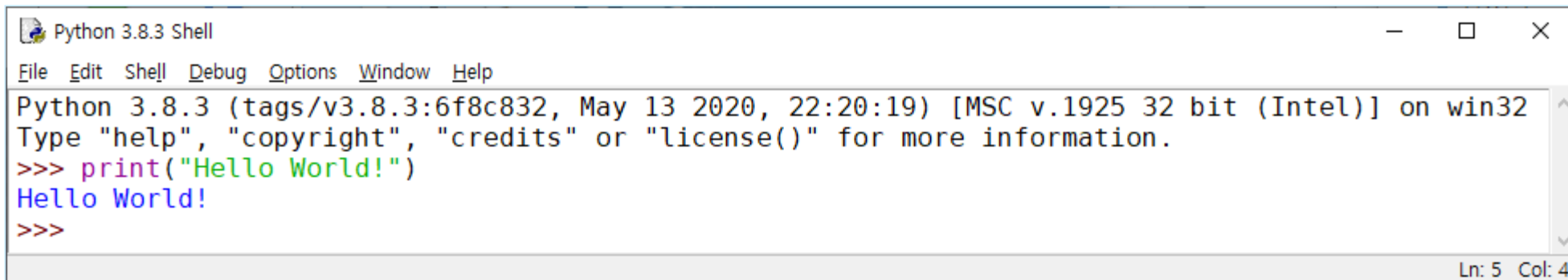
윈도우에서 파이썬 시작

■ "IDLE" 프로그램 실행



```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

- 파이썬 쉘에서는 >>> 뒤에 사용자가 명령어를 입력하고 엔터키를 누르면 명령어가 실행되고 화면에 출력된다.



```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World!")
Hello World!
>>>
```

Linux(Ubuntu)에서 파이썬 실행

- 터미널 실행
 - 프로그램 > 보조프로그램 > 터미널 (Ctrl + Alt + t)
- 리눅스의 경우 기본적으로 파이썬이 설치되어 있음
 - Python 실행 파일 위치 및 버전 알기
which python3 --> /usr/bin/python
python3 -V --> 3.6.x

```
$ python3
Python 3.6.9 (default, Apr 18 2020, 01:56:04)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

리눅스 소프트웨어 설치

■ apt-get

- 리눅스 소프트웨어를 설치하고 제거하는 일을 함
 - 소스코드 컴파일을 통해 소프트웨어 패키지 확인, 구성, 설치 자동화
 - `sudo apt-get install <패키지 이름>`

■ vim을 설치 실습

```
sudo apt-get install vim
```

- `sudo`를 붙이는 이유는 시스템에 소프트웨어를 설치하기 위해 root 권한이 필요하기 때문

pip (python 패키지 관리자)

- Python에서 사용되는 모듈을 쉽게 설치할 수 있도록 함.

- pip 설치

sudo apt-get install python-pip

예) Numpy 설치

pip install numpy

파이썬 실행

- 짧고 간단한 예제 코드는 파이썬 셸을 이용
- 길고 복잡한 예제 코드는 편집기를 이용

(1) 파이썬 셸로 코딩하기

셸 종료: **exit()**라고 입력 또는 **Ctrl+D**

```
>>> print("Hello, World.")
```

(2) 코드 편집기로 코딩 후, 스크립트 실행

```
$ vim run_test.py
$ python run_test.py
Hello, World!!
```

run_test.py

```
# run_test.py
print("Hello, World!!")
```



SAMPLE PROGRAM



산술 연산자

■ 산술 연산자

$+, -, *, /$	더하기, 빼기, 곱하기, 나누기
$\%$	나머지 (modulo)
$//$	몫
$**$	제곱

※ Python 3.x 버전
/ 연산자 : float 으로 처리 함
`>>> 10 / 10`
1.0

※ 지수(**)는 다른 연산자들 보다 높은 우선 순위를 가짐

```
pirl@pirl-Precision-Tower-7910: ~  
>>> 2 + 3  
5  
>>> 210 - 91  
119  
>>> 3 * 12  
36  
>>> 6 / 2  
3  
>>>
```

※ Python 2.x 버전


■ 실습하기

- (1) `7 % 3`
- (2) `7 // 3`
- (3) `2 ** 3`

- (4) `2 ** 4 + 5`
- (5) `3 * 5 ** 2`

변수

- 변수(variable)
 - 어떤 값을 저장하기 위한 메모리 공간
- 변수 이름 규칙
 - 변수 이름은 알파벳, 숫자, underline(_)로 구성되는데, 첫 글자는 반드시 알파벳 또는 underline(_)로 시작함
 - 이름 중간에 공백이 올 수 없음
 - 대소문자 구분함
 - 이미 사용하고 있는 몇몇 예약어(if, for, return 등)는 사용할 수 없음
 - 내장 함수 이름이나 모듈 이름은 피할 것



```
>>>import keyword  
>>>keyword.kwlist      # 예약어 목록을 보여줌
```

변수와 기본 자료형

■ 변수 생성

- 파이썬은 변수에 값이 대입되는 시점에 변수의 자료형이 자동으로 정해짐

```
pirl@pirl-Precision-Tower-7910: ~  
>>> a = 100  
>>> print(a)  
100  
>>>
```

※ = 는 대입연산자로 오른쪽의 것을 왼쪽으로 넣는다라는 의미

■ 기본 자료형

```
pirl@pirl-Precision-Tower-7910: ~  
>>> boolVar = True  
>>> intVar = 100  
>>> floatVar = 123.45  
>>> strVar = "Python"  
>>> type(boolVar), type(intVar), type(floatVar), type(strVar)  
(<class 'bool'>, <class 'int'>, <class 'float'>, <class 'str'>)  
>>>
```

※ type() 함수는 변수의 종류를 확인하는 함수

대입연산자(=, +=, -=, *=, /=)

```
>>> a = 1
>>> a = a + 1
>>> print(a)

>>> a += 2    # ???

>>> b = 10
>>> b -= 1    # ???
```

input()를 사용한 입력

■ input() 사용법

변수

사용자가 입력한 문자열을 정수로 변환한다

```
number = int(input("첫 번째 숫자를 입력하세요 : "))
```


안내 메시지를 출력하고 사용자가 입력한 값을 문자열 형태로 받는다

```
pir1@pir1-Precision-Tower-7910: ~  
>>> name = input("Enter your name: ")  
Enter your name: Hong Gildong  
>>> print(name)  
Hong Gildong  
>>> type(name)  
<class 'str'>  
>>>
```


input()를 사용한 입력

■ 문자열을 정수로 변경

```
pirl@pirl-Precision-Tower-7910: ~  
>>> number = input("Enter a number: ")  
Enter a number: 123  
>>> print(number)  
123  
>>> type(number)  
<class 'str'  
>>>
```



```
pirl@pirl-Precision-Tower-7910: ~  
>>> number = int(number)  
>>> print(number)  
123  
>>> type(number)  
<class 'int'  
>>>
```

※ 형 변환

- ※ 문자열 -> 정수: int() 함수 이용
- ※ 문자열 -> 실수: float() 함수 이용
- ※ 숫자를 문자열로 변환: str() 함수 사용

print를 사용한 출력

■ print

- 파이썬 3 버전 (출력할 문자열에 괄호를 필요로 한다.)

```
print("Hello Python")
```

- 파이썬 2.7 버전

```
print "Hello Python"
```

※ 파이썬 2.7 버전에서는 괄호를 사용해도 동일하게 동작한다.

■ 문자열 합치기 +

- 큰 따옴표(")로 둘러싸인 문자열은 + 연산과 동일

```
In [1]: 1 print ("life" "is" "too short")
```

```
lifeistoo short
```

```
In [2]: 1 print ("life" + "is" + "too short")
```

```
lifeistoo short
```

print를 사용한 출력

- 문자열 띄어쓰기는 **콤마**로 한다

```
In [3]: 1 print("life", "is", "too short")
```

```
life is too short
```

- 변수와 같이 출력

```
In [4]: 1 a = 100
        2 b = 200
        3 result = a + b
        4 print(a, "+", b, "=", result)
```

```
100 + 200 = 300
```

- 문자열 곱하기 *

```
In [5]: 1 print("python" * 3)
```

```
pythonpythonpython
```

print를 사용한 출력

- 문자열 내에 변수의 값을 출력하고 싶으면 %

```
print ("I eat %d apples." % 3)
```

```
number = 10
```

```
day = "three"
```

```
print ("I ate %d apples, so I was sick for %s days. " % (number, day))
```

- %s 는 어떤 형태든 대입이 가능
(%s는 자동으로 % 뒤의 값을 문자열로 치환)

```
print ("I eat %s apples." % 3)
```

실습 문제 1

- 화씨온도(°F)를 입력 받아서 섭씨온도(°C)로 바꾸는 프로그램을 작성하시오. (밑줄은 사용자 입력)

$$C = (F - 32) * \frac{5}{9}$$

In [37]:

```
화씨온도: 100  
섭씨온도: 37.7777777778
```

실습 문제 2

- 사용자로부터 투입한 돈과 물건 값을 입력 받아, 잔돈을 계산하여 출력한다. 단, 동전의 개수는 최소화 할 것
- (가정)
 - 물건 값은 100원 단위
 - 자판기의 동전은 500원, 100원만 있음

투입한 돈: **5000**

물건값: **2700**

거스름돈: 2300

500원짜리: 4개

100원짜리: 3개



시퀀스 자료형



시퀀스(Sequence) 자료형

- 내부 원소들이 순서를 가지고 구성된 자료형
- 문자열, 리스트, 튜플 등
- 공통 특성
 - (1) 연결('+ 연산자)
 - (2) 반복('* 연산자)
 - (3) 인덱싱(Indexing)
 - (4) 슬라이싱(Slicing)
 - (5) 멤버확인('in')
 - (6) 크기정보(len())



문자열



문자열

- 문자열(String)

- 문자들의 집합으로 작은따옴표(' ') 또는 큰따옴표(" ")로 묶임

```
>>> "Hello World"
'Hello World'

>>> a = 'Python is fun!'
>>> a
'Python is fun!'
>>> type(a)
<class 'str'>

>>> "0904"    # 문자열??
```

문자열 연결 및 반복

■ '+' 연산자

- 자료형이 동일한 두 시퀀스 자료를 연결하여 새로운 시퀀스 자료로 생성
- 문자열 사이에 사용하여 합치는 기능을 함

```
>>> 'Python' + 'is' + 'fun!'  
>>> 'Pythonisfun!'
```

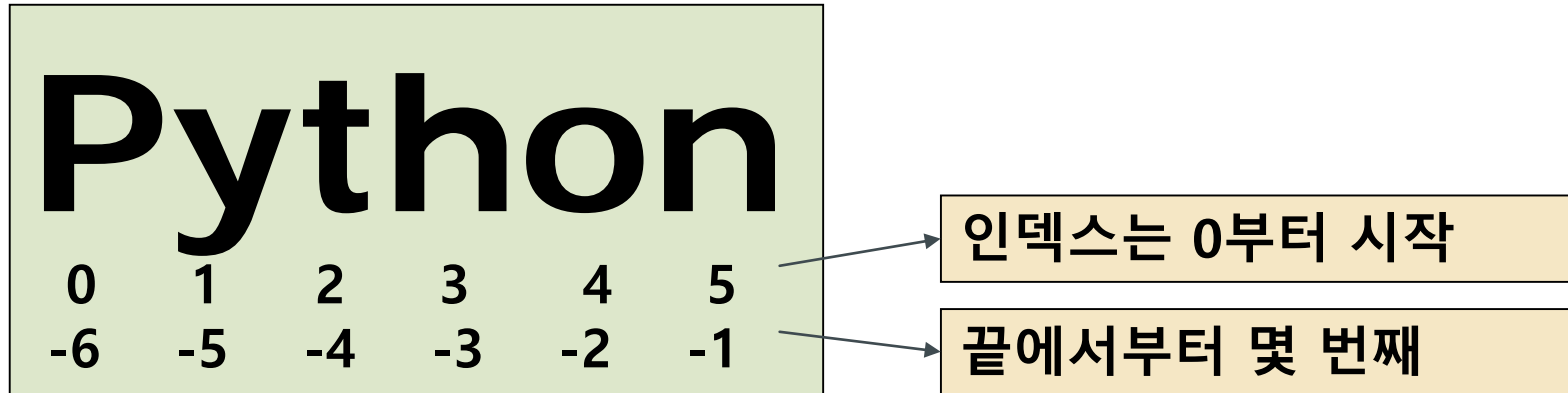
■ '*' 연산자

- 시퀀스 자료를 여러 번 반복하여 새로운 시퀀스 자료로 생성

```
>>> 'Python' * 3  
'PythonPythonPython'
```

문자열 인덱싱

- 인덱스를 통해 해당 값에 접근



```
>>> word = 'Python'
>>> word[0]
'P'
>>> word[-1]
'n'
```

문자열 슬라이싱

- 특정 구간의 값을 취할 수 있음(한 번에 여러 개 추출)

[시작 인덱스:끝 인덱스+1:스텝]

- 시작 인덱스부터 끝 인덱스까지의 자료를 취하라는 의미
- 스텝은 자료를 취하는 간격으로 생략해도 되며 기본값은 1

```
>>> word = 'Python'
>>> word[0:2]
'Py'
>>> word[:2]      # 콜론 앞 생략: 문자열 시작부터 ~
'Py'
>>> word[3:]      # 콜론 뒤 생략: ~ 문자열 끝까지
'hon'
```

```
>>> word = "12345"
>>> word[:]
'12345'
>>> word[::-1]
'54321'
>>> word[:2]
'12'
```

멤버확인 - 문자열

- 'in' 키워드를 사용하여 특정 값이 시퀀스 자료의 요소로 속해 있는지 확인

<값> in <자료>

```
>>> word = 'Python'
>>> 'o' in word      # 문자열에 특정 문자 있는지 확인
True
>>> 'x' in word
False

>>> 'on' in word    # 문자열에 특정 문자열 있는지 확인
True
```

문자열 관련 함수

■ len()

- 시퀀스 자료의 크기를 알 수 있음
- 문자열의 문자 개수 반환하는 내장 함수

```
>>> word = 'Python'
>>> len(word)
6
>>> len('Python is fun!')
??
```

■ count()

```
>>> s = 'Python is fun!'
# 문자열 중 문자 'n'의 개수 반환
>>> s.count('n')
2
```

■ index()

```
>>> s = 'Python is fun!'
# 문자열 중 문자 'n'이 처음으로 나온 위치 반환
>>> s.index('n')
5
```

■ join()

```
>>> a = ','
>>> a.join('1234') # 문자열 합해줌
'1,2,3,4'
```

실습 문제 3

- 문자열의 마지막 문자를 구해서 출력해 보자.

```
>>> word = 'Hello'
>>> word[-1]    # 음의 인덱스 사용
# len() 함수 이용
```

He11o

0 1 2 3 4
-5 -4 -3 -2 -1

문자열 관련 함수

■ split()

- 문자열을 공백이나 다른 문자로 분리해서 *리스트*로 반환

```
>>> s1 = "python is fun"
>>> s1.split()           # 공백으로 분리
['python', 'is', 'fun']

>>> s2 = "one:two:three"
>>> s2.split(":")        # :(콜론)으로 분리
['one', 'two', 'three']

>>> s2
'one:two:three'
```



리스트



리스트

- 여러 정보를 하나로 묶어서 저장하고 관리할 수 있게 함

- 선언 `리스트이름 = [값1, 값2, 값3]`

: []안에 쉼표로 구분하여 정보를 적어 주면 됨

- 문자열을 원소로 가지는 예제

```
>>> fruit = ["banana", "apple", "cherry"]
```

- 숫자를 원소로 가지는 예제

```
>>> score = [70, 99, 25, 100]
```

- Empty list

```
>>> empty_list = []
```

리스트 인덱싱

■ 인덱스

: 원소가 배열된 순서를 나타냄. (0번부터 시작)

```
>>> season = ['spring', 'summer', 'fall', 'winter']
```

spring	summer	fall	winter
season[0]	season[1]	season[2]	season[3]

■ 인덱싱: 인덱스를 이용하여 원소에 접근할 수 있음

```
>>> season[1]
```

결과??

리스트 슬라이싱

- 특정 구간의 값을 취할 수 있음(한 번에 여러 개 추출)

[시작 인덱스:끝 인덱스+1:스텝]

```
>>> a = [45, 87, 99, 21, 55]
>>> a[0:2]
[45, 87]
>>> a[:2] # 콜론 앞 생략: 리스트 시작부터 ~
[45, 87]
>>> a[3:] # 콜론 뒤 생략: ~ 리스트 끝까지
[21, 55]
```

리스트 더하기(+) 및 반복(*)

■ '+' 연산자, '*' 연산자

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

```
>>> b = [4, 5, 6]
```

```
>>> a + b
```

```
[1, 2, 3, 4, 5, 6]
```

```
>>> a * 3
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

멤버확인 - 리스트

■ 'in' & 'not in'

in	list의 element 인가를 결정하는 연산자
not in	list의 element가 아닌 element를 결정하는 연산자

```
>>> fruit = ['banana', 'apple', 'orange']  
>>> 'apple' in fruit  
True  
>>> 'cherry' in fruit  
False
```

리스트 수정 및 삭제

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

```
>>> a[2] = 8
```

a[2]의 요소 값 수정

```
>>> a
```

```
[1, 2, 8]
```

```
>>> del a[0]
```

del 키워드 사용해 리스트 삭제 (del a[x:y] 형태로 사용가능)

```
>>> a
```

```
[2, 8]
```


리스트 패킹, 언패킹

- 패킹: 한 변수에 여러 개의 데이터를 넣는 것
- 언패킹: 한 변수의 데이터를 각각의 변수로 반환

```
>>> t = [1, 2, 3]      # 1,2,3을 변수 t에 패킹
>>> a, b, c = t        # t 값 1,2,3을 a,b,c에 언패킹
>>> print (a, b, c)
1 2 3

>>> x, y = t           # 개수가 맞지 않을 때 ???
```

실습 문제 4

- 아래의 실행예제를 참고하여 프로그램을 작성하라.
- 실행예제(밑줄은 사용자 입력)

날짜(연/월/일)입력: 2020/07/28
입력한 날짜의 10년 후는 2030년 7월 28일