

김범수



Python 시작하기

Basic of python

By POSCAT



Contents

Python 설치(windows) 및 에디터 설정	03
--------------------------------	----

변수와 리스트	11
---------	----

조건문(if)과 반복문(for, while)	20
--------------------------	----

함수	27
----	----

Python 설치(windows) 및 에디터 설정

- Python은 문법이 간단해서 C++, Java 등의 언어보다 학습하기 간단하다는 장점을 가지고 있습니다. 그 외에도 다양한 외부 모듈을 쉽게 설치할 수 있어서 다양한 분야에서 사용될 수 있습니다.
- 최신 버전인 3.9.x 버전은 다른 모듈과 호환이 안되니, 3.8.x, 3.7.x과 같은 구 버전 설치를 권장합니다. (강의에서는 3.8.6 환경을 사용할 예정입니다.)
- <https://www.python.org/downloads/release/python-386/> (파이썬 3.8.6 설치 주소)
- 위 사이트에 들어가서 맨 밑에 설치 파일들이 존재하는 데, 자신의 환경에 맞는 것을 설치하면 됩니다
- | | |
|---|---------|
| Windows x86-64 executable installer | Windows |
|---|---------|

 (Windows 64-bit 기준)
- | | |
|--|---------|
| Windows x86 executable installer | Windows |
|--|---------|

 (Windows 32-bit 기준)
- cmd 창에서 `wmic os get osarchitecture` 를 입력하면 32-bit인지 64-bit인지 확인할 수 있습니다.

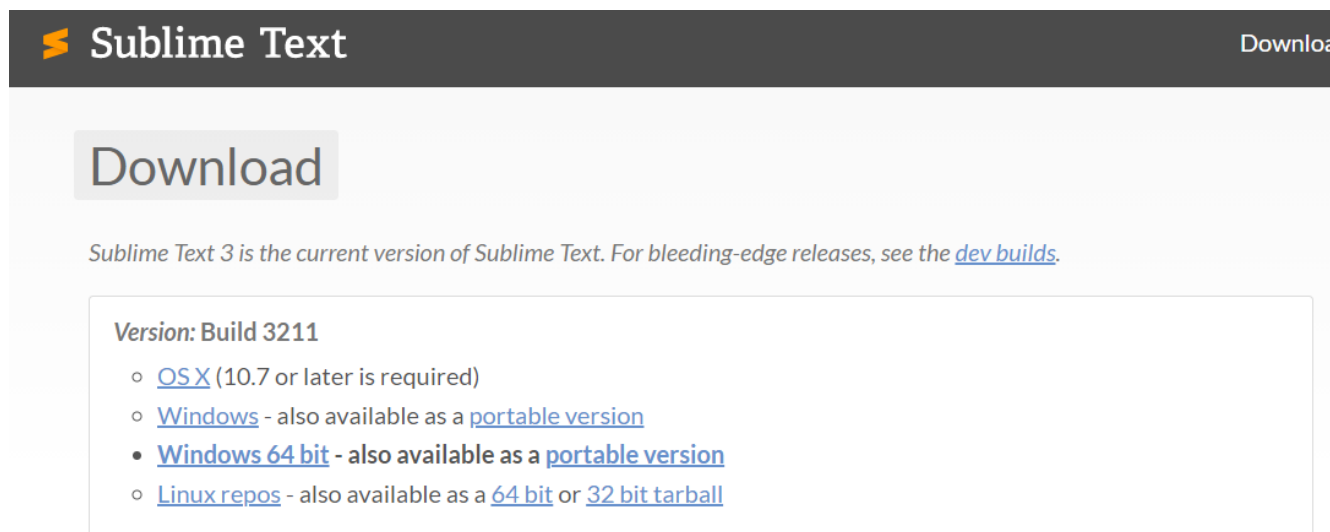
Python 설치(windows) 및 에디터 설정

- 설치 창에서 Add Python 3.x to PATH는 무조건 체크해주세요. 체크하지 않으면 나중에 코딩할 때 문제가 생겨서 다시 설치해야 합니다.



Python 설치(windows) 및 에디터 설정

- 파이썬 에디터로는 pycharm, jupyter, vs code, sublime text 등 다양한 환경에서 파이썬을 실행할 수 있습니다. 만약 사용하고 있는 에디터가 있다면 그것을 사용하면 됩니다.
- 본 강의에서는 sublime text3을 사용할 예정이므로, sublime text3 설치 방법을 설명합니다.
- <https://www.sublimetext.com/3> 에서 pc(Window, Linux...)에 맞는 설치 파일로 설치하면 됩니다.



Python 설치(windows) 및 에디터 설정

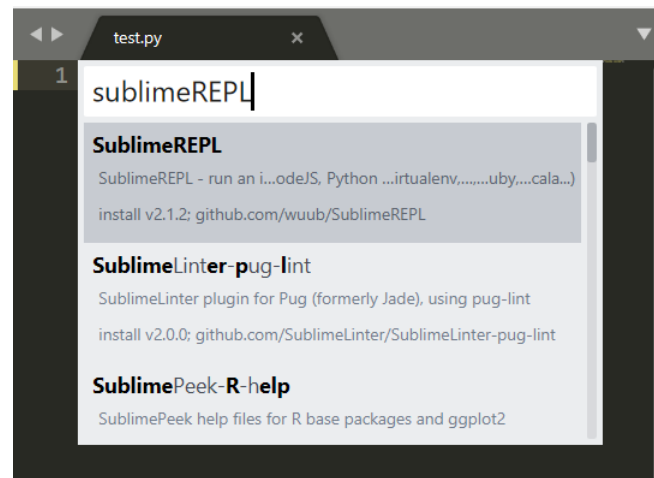
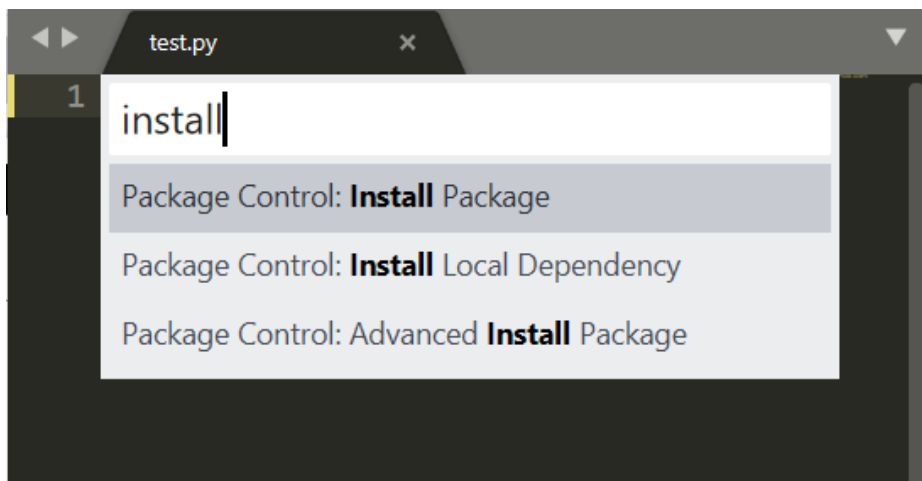
- Sublime text3을 실행하고 아래의 코드를 컴파일해서 python과 sublime text가 잘 설치되었는지 확인해보세요. Ctrl + b를 누르면 컴파일이 됩니다. 정상적으로 설치되었다면 Hello World가 출력될 것입니다. (컴파일 하기 전에 코드를 저장해야 컴파일이 정상적으로 진행됩니다.)
- 파일 명에 .py를 붙여야 파이썬 코드로 인식하니 새 파일을 만들 때 .py 확장자를 지정해주세요.



The screenshot shows the Sublime Text 3 interface. The title bar indicates the file path is C:\Users\Wqjatn\Desktop\test.py. The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The editor window shows a single line of Python code: `print("Hello World")`. Below the editor, the console output displays "Hello World" and "[Finished in 0.4s]". The status bar at the bottom shows "ASCII, Line 1, Column 21" and "Tab Size: 4".

Python 설치(windows) 및 에디터 설정

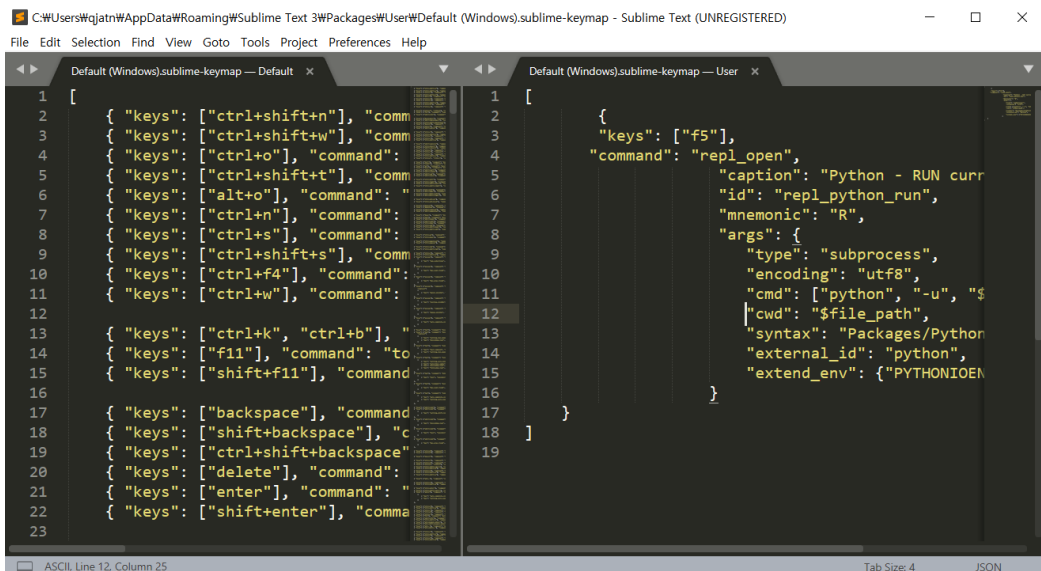
- 주어진 환경에서는 프로그램에 입력을 할 수가 없습니다. 따라서 입력을 할 수 있는 창인 sublimeREPL을 설치해야 합니다. Ctrl+shift+p 를 누르면 아래와 같이 창이 뜨는데, install을 입력하고 enter를 누릅니다. 조금 기다리면 새로운 창이 뜨는데, 그 창에서 sublimeREPL을 검색한 후 enter를 누르면 설치가 됩니다. (만약 sublimeREPL이 뜨지 않는 경우, 몇 분 정도 기다리고 sublime을 껐다가 키면 보입니다.)



- Sublime을 사용하면 한글이 깨지는 문제가 발생합니다. 위와 같은 방법으로 ConvertToUTF8을 설치하면 문제가 해결됩니다.

Python 설치(windows) 및 에디터 설정

- sublimeREPL을 실행하기 위해서는 Tools > SublimeREPL > Python > RUN current file을 통해 실행해야 합니다. 이는 아주 귀찮기 때문에 단축키를 등록해주어야 합니다. Preference > Key bindings를 누르면 2개의 탭이 뜨는데, 오른쪽 탭에 아래의 내용을 복붙한 후 저장하면, F5가 단축키가 되어 F5를 누르면 파이썬 SublimeREPL이 작동합니다.

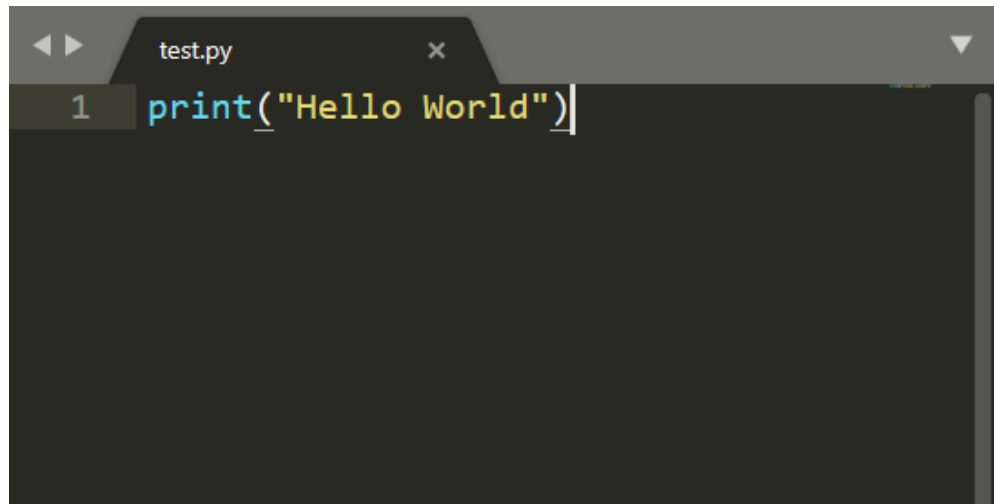


```
[
  {
    "keys": ["f5"],
    "command": "repl_open",
    "caption": "Python - RUN current file",
    "id": "repl_python_run",
    "mnemonic": "R",
    "args": {
      "type": "subprocess",
      "encoding": "utf8",
      "cmd": ["python", "-u", "$file_path"],
      "cwd": "$file_path",
      "syntax": "Packages/Python/Python.tmLanguage",
      "external_id": "python",
      "extend_env": {"PYTHONIOENCODING": "utf-8"}
    }
  }
]
```

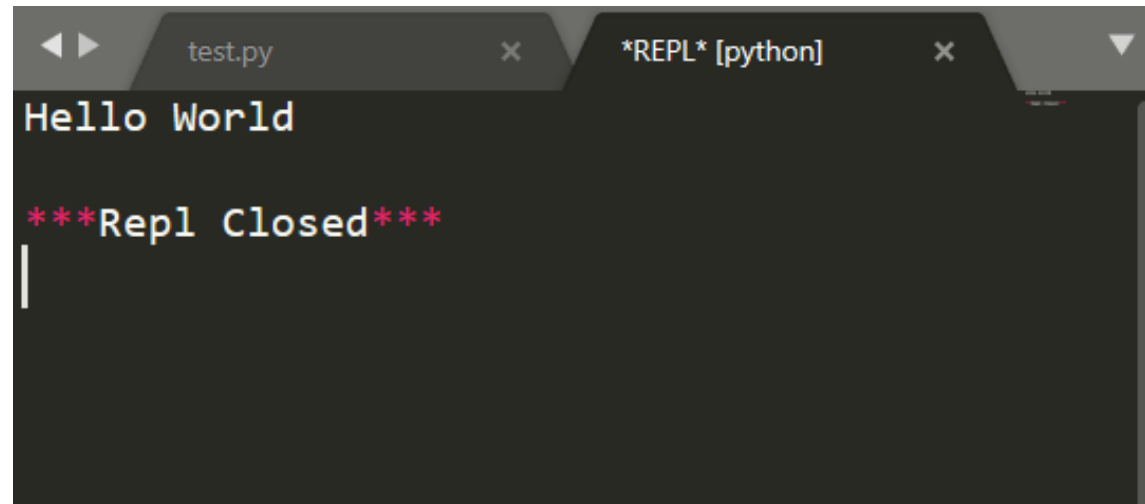

Python 설치(windows) 및 에디터 설정



- Sublime Text 3를 한 번 꺾다가 킨 후, F5를 눌러 아까의 코드를 컴파일해보세요. 아래와 같이 새 창에서 Hello World가 출력된다면 파이썬 설정은 끝난 것입니다. 그럼 이제 본격적으로 파이썬의 기초부터 시작하겠습니다.



```
test.py
1 print("Hello World")
```



```
*REPL* [python]
Hello World
***Repl Closed***
```

변수와 리스트



- 앞에서 봤듯이 print 함수를 이용하면 원하는 내용을 출력할 수 있습니다. 아래 그림처럼 숫자를 주면 숫자를, 문장을 주면 문장을 출력하는 함수이지요.

```
1 print("Hello World")
2 print(123)
3 print("!")
```

Hello World
123
!

- 간단한 사칙연산도 진행할 수 있습니다.

```
1 print(5 + 3)
2 print(5 - 3)
3 print(5 * 3)
4 print(5 / 3)
5 print(5 % 3)
6 print(5 ** 3)
```

8
2
15
1.6666666666666667
2
125

변수와 리스트

- 변수는 간단히 말하면 “값을 저장하는 공간” 입니다. 앞으로는 변수를 사용하여 다양한 계산을 진행할 예정입니다.
- 변수에는 숫자나 문자를 저장할 수 있고, print를 이용하여 변수를 출력할 수도 있습니다. 또한 변수끼리도 사칙연산을 할 수 있습니다.
- 변수에도 다양한 종류가 존재합니다. 정수를 저장하는 int형, 실수를 저장하는 float형, 문자열을 저장하는 string 형이 존재합니다. 파이썬에서는 type() 함수를 이용하여 변수형을 확인할 수 있습니다.

```
1 a = 1
2 b = 12.34
3 c = "abc"
4 print(a + b)
5 print(type(a))
6 print(type(b))
7 print(type(c))
```

```
13.34
<class 'int'>
<class 'float'>
<class 'str'>
```

변수와 리스트

- 만약 여러 개의 변수를 한 줄에 출력하고 싶다면 어떻게 해야 할까요? 이 때는 format() 함수를 사용하면 문자열을 예쁘게 만들 수 있습니다.
- format 함수에서는 중괄호로 포매팅을 지정하고 인자로 중괄호 안에 들어갈 값을 전달합니다. 아래의 예제를 통해 format 함수를 어떻게 사용하는 지 살펴보겠습니다.

```
a = 123
b = 456
c = 789
string = "a = {}, b = {}, c = {}".format(a, b, c)
print(string)
```

```
a = 123, b = 456, c = 789
```

- 실수 출력의 경우, {:.자리수f}와 같은 형태로, 소수점 몇 자리까지 출력할 지 정할 수 있습니다.

```
a = 12.3456789
print("a = {:.2f}".format(a))
print("a = {:.4f}".format(a))
```

```
a = 12.35
a = 12.3457
```

- 이외에도 format() 함수를 다양하게 활용할 수 있습니다. 이는 뒤의 예제를 풀어보면서 하나씩 알아보겠습니다.

변수와 리스트



- 만약 다른사람이 내 코드를 읽는다면 내 코드가 어떤 내용인지 설명해야겠죠? 이 때 사용하는 것이 주석입니다. 파이썬에서는 #(삽)으로 주석을 작성하는 방법과, 따옴표 3개로 주석을 작성하는 법, 두 가지가 있습니다. 아래와 같이 사용하면 됩니다.
- #은 한 줄 주석에, 따옴표 3개는 여러 줄 주석에 사용됩니다. 한글이 깨지면 ppt 8번 페이지를 참고하시면 됩니다.

한 줄 주석

"""

여러줄주석

"""

'''

여러줄주석

'''

'''

이렇게
섞어쓰면

"""

"""

절대
안돼요

'''

변수와 리스트

- 만약 입력을 받고 싶다면 어떻게 해야 할까요? 파이썬에서는 input()함수를 이용하여 입력을 받고, 입력받은 값을 변수에 저장할 수 있습니다. 하지만, input() 함수는 숫자를 입력으로 주어도 그것을 문자열로 인식합니다. 따라서 입력을 숫자로 바꾸기 위해서는 int() 를 사용해야 합니다. int()를 사용하면, 문자열을 그에 해당하는 숫자로 바꿔주게 됩니다.

```
1 a = input()
2 print(a)
3 print(type(a))
4 a = int(a)
5 print(a)
6 print(type(a))
```

```
123
123
<class 'str'>
123
<class 'int'>
```

직접 입력으로 넣은 숫자

- 지금까지 배운 내용을 바탕으로 간단한 연습 문제 하나를 풀어보겠습니다. 학생 세 명의 성적을 입력받아 그 평균을 출력하는 문제를 풀어보겠습니다.

변수와 리스트



- 연습문제 1
- 학생 세 명의 성적($0 \leq x \leq 100$)이 한 줄에 하나씩 총 세 줄에 걸쳐 주어집니다. 이 때, 세 학생의 평균 성적을 소수점 2자리 까지 반올림해서 출력해봅시다.
- Average score : (평균성적) 형태로 출력해봅시다.

Example Input



96

87

91

Example Output



Average score : 91.33

변수와 리스트



- 만약 여러 개의 변수를 저장하고 싶다면 어떻게 해야 할까요? 예를 들어 앞의 연습문제에서 학생이 100명이라면? 1000명이라면? 당연하게도 1000개의 변수를 일일이 만드는 건 매우 비효율적입니다. 따라서 파이썬에서는 리스트를 제공하여 여러 개의 변수를 저장할 수 있도록 합니다.
- 리스트는 대괄호를 사용하여 정의하며, 리스트 안에 저장하는 변수들을 리스트의 원소라고 합니다. 이 때 리스트의 원소는 정수, 실수, 문자열, 또는 리스트까지 무엇이든 가능합니다.

```
List1 = [1, 2, 3]
List2 = [1, 2.34, "567"]
print(List1)
print(List2)
print(type(List1))
```

```
[1, 2, 3]
[1, 2.34, '567']
<class 'list'>
```

변수와 리스트

- 리스트의 각 원소는 index 숫자를 통해 접근할 수 있습니다. 이 때 index 번호는 0부터 시작합니다. 즉 첫 번째 원소는 0을 통해 접근할 수 있고, 두 번째 원소는 1... 이런 식으로 index가 매겨집니다.

```
List1 = [1, 2, 3, 4]
List2 = [2, 3, 4, 5]
List2[0] = List2[0] + List1[1]
print(List2)
print(List2[0])
```

- 더하기를 통해 두 리스트를 합칠 수도 있고, sort 함수를 통해 리스트 내의 원소를 정렬할 수도 있습니다.

```
List1 = [1, 2, 3, 4]
List2 = [2, 3, 4, 5]
print(List1 + List2)
```

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

```
List1 = [5, 3, 1, 4, 2]
List1.sort()
print(List1)
```

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

- 왜 sort함수를 sort(List1) 가 아니라 List1.sort() 형태로 사용하는지는 다음 2강에서 class에 대한 내용을 배워야 이해할 수 있으니 지금은 사용 방법만 알아둡시다.

조건문(if)과 반복문(for, while)

- 앞에서 int, float 등의 자료형에 배웠는데, 사실 bool이라는 자료형이 하나 더 존재합니다. Bool 자료형은 True와 False 값 중 하나를 가지는 아주 단순한 자료형입니다.
- 두 수를 비교하고 싶을 때, 등호나 부등호를 사용해서 비교할 수 있습니다. 그리고, 비교 결과를 bool 형으로 반환하게 됩니다.

```
print(1 == 2)    False
print(1 == 1)    True
print(1 > 1)      False
print(1 >= 1)     True
```

- 여러 개의 조건을 표현하고 싶다면, 즉 A 이고 B, A 또는 B와 같은 형태로 쓰고 싶다면, and와 or을 사용하면 됩니다.
- A and B는 A와 B가 모두 True일 경우에만 True가 되고, A or B는 A와 B중 하나라도 True라면 True가 됩니다.

```
a = 3
print((a == 3) and (a > 3))    False
print((a == 3) or (a > 3))     True
print((a == 3) and (a >= 3))   True
print((a > 3) or (a < 3))      False
```

조건문(if)과 반복문(for, while)

- 성적이 90점 이상이면 S, 90점 미만이면 U를 출력하고 싶은 경우에는 어떻게 하면 될까요? if문을 사용하면 조건에 따라 원하는 것을 실행할 수 있습니다.
- 오른쪽 그림의 형태로 조건문을 사용하며, 수행할 문장은 tab으로 들여쓰기를 해야 합니다.
<조건문>이 True일 경우, 문장1, 문장2...이 실행되며, False일 경우, 문장A, 문장B...가 실행됩니다.
- 콜론(:) 이 없거나 들여쓰기가 일정하게 되지 않은 경우 오류가 발생하니 주의하세요.
- 위에서 조건이 True라면 내용1이 실행되고, 그렇지 않다면 내용2가 실행이 됩니다. 이 때, 내용1과 내용2는 tab을 이용하여 들여쓰기를 해서 표현해야 합니다.

```
if <조건문>:  
    수행할 문장1  
    수행할 문장2  
    ...  
else:  
    수행할 문장A  
    수행할 문장B  
    ...
```

```
score = int(input())  
if score >= 90:  
    print("S")  
else:  
    print("U")  
print("Your score is {}".format(score))
```

```
90  
S  
Your score is 90
```

```
80  
U  
Your score is 80
```

조건문(if)과 반복문(for, while)

- 이번에는 오른쪽 점수-학점 표를 따라서 학점을 출력하는 문제를 생각해봅시다.
- if-else만 사용한다면 계속 들여쓰기를 하게 되어 왼쪽의 코드처럼 보기 난잡해지게 됩니다.
따라서 파이썬에서 제공하는 else와 if를 합친 elif를 사용하면 오른쪽의 코드처럼 간결해집니다.

Score	Grade
[90, 100]	A
[80, 90)	B
[70, 80)	C
[60, 70)	D
[0, 60)	F

```
score = int(input())
if score >= 90:
    print("A")
else:
    if score >= 80:
        print("B")
    else:
        if score > 70:
            ...
            ...
```

```
score = int(input())
if score >= 90:
    print("A")
elif score >= 80:
    print("B")
elif score >= 70:
    print("C")
...
```

조건문(if)과 반복문(for, while)

...

- 특히 파이썬에서는 `x in s`, `x not in s`과 같은 조건문을 사용할 수 있습니다. `s` 리스트, 문자열 안에 `x`라는 원소가 존재하는지를 쉽게 확인할 수 있습니다.

<pre>print(1 in [1, 2, 3]) print(2 in [3, 4, 5]) print('p' in "python") print('P' in "python")</pre>	<pre>True False True False</pre>
--	--

- 만약 조건문을 여러 줄로 적는 게 귀찮다면 아래의 조건부 표현식을 이용해 한 줄에 적을 수도 있습니다. 주로 코드를 간결하게 작성하기 위해 가끔 사용됩니다. 이전의 S/U 문제를 조건부 표현식으로 풀면 아래와 같습니다.

<조건문이 참인 경우> `if` <조건문> `else` <조건문이 거짓인 경우>

```
score = int(input())  
print("S") if score >= 90 else print("U")  
print("Your score is {}".format(score))
```

조건문(if)과 반복문(for, while)

- 이제 반복문에 대해 알아보시다. 반복문은 문장을 반복해서 수행할 때 사용합니다. while문은 오른쪽의 구조로 사용됩니다. if와 마찬가지로 콜론(:)과 들여쓰기에 주의합니다.

- 조건문 확인 -> 문장 수행 -> 조건문 확인 -> 문장 수행 -> ... 의 순서로 실행됩니다. [Code1]의 1~10까지 출력하는 예제를 적용해보면

count(= 1) <= 10 -> count(= 1) 출력 및 count 1 증가 -> count(= 2) <= 10 -> count(= 2) 출력 및 count 1 증가
-> ... -> count(= 10) <= 10 -> count(= 10) 출력 및 count 1 증가 -> count(= 11) <= 10 -> while문 종료
의 순서로 이루어지게 됩니다.

- [Code2]와 같이 break를 사용해서 while문을 빠져나올 수도 있습니다.
- [Code3]에서 처럼 continue를 사용하면, while문의 맨 앞으로 가서 처음부터 실행합니다. 따라서 [Code3]을 실행하면 홀수만 출력됩니다.

```
while <조건문>:  
    수행할 문장1  
    수행할 문장2  
    ...
```

```
count = 1  
while count <= 10:  
    print(count)  
    count += 1 # count = count + 1
```

[Code1]

```
count = 1  
while True:  
    print(count)  
    count += 1 # count = count + 1  
    if count > 10:  
        break
```

[Code2]

```
count = 1  
while count <= 10:  
    if count % 2 == 0:  
        count += 1  
        continue  
    print(count)  
    count += 1
```

[Code3]

조건문(if)과 반복문(for, while)

- 이제 반복문 중 for에 대해서 알아보시다. for문은 오른쪽의 구조로 사용됩니다.
마찬가지로 콜론(:)과 들여쓰기를 주의해서 사용합시다.
- for문은 변수에 리스트의 첫 원소를 대입 -> 문장 수행 -> 변수에 리스트의 다음 원소를 대입 -> 문장 수행 -> ...
의 순서로 실행됩니다. [Code1]처럼 for문을 이용해 리스트 안의 모든 원소를 출력할 수 있습니다.
- while문과 유사하게 break를 써서 for문을 빠져나올 수도 있고, continue를 써서 for문 처음으로 돌아올 수도 있습니다.
[Code2]에서는 x가 짝수일 때 for문을 빠져나와 1, 3만 출력이 되고,
[Code3]에서는 x가 짝수일 때 continue로 인해 for의 맨 처음으로 돌아가 홀수인 1, 3, 5, 9만 출력됩니다.

```
for <변수> in <리스트 등>:  
    수행할 문장1  
    수행할 문장2  
    ...
```

```
List = [1, 3, 4, 5, 6, 9]  
for x in List:  
    print(x)
```

[Code1]

```
List = [1, 3, 4, 5, 6, 9]  
for x in List:  
    if x % 2 == 0:  
        break  
    print(x)
```

[Code2]

```
List = [1, 3, 4, 5, 6, 9]  
for x in List:  
    if x % 2 == 0:  
        continue  
    print(x)
```

[Code3]

조건문(if)과 반복문(for, while)

...

- for문은 숫자 리스트를 자동으로 만들어주는 range(x, y) 함수와 같이 사용됩니다. range(x, y) 함수에서 x는 시작 숫자, y는 마지막 숫자(x, y는 정수)를 의미하는 데, 이 때 끝 숫자는 포함되지 않습니다. 따라서 1~10까지 출력하는 예제를 range 함수를 이용하면 [Code1]과 같이 사용할 수 있습니다.
- 리스트 내포를 사용하여 보다 간편하게 리스트를 만들 수 있습니다. 아래의 구조로 사용되며 우리는 거의 사용하지 않으니 이런 것이 있구나 정도만 알아두시면 될 것 같습니다. [Code2]처럼 사용하여 List의 수 중 짝수만 골라서 2를 곱해 새로운 리스트를 만들 수 있습니다.

[표현식 for 변수 in 반복가능객체<리스트 등> if 조건문]

```
for x in range(1, 11):  
    print(x)
```

[Code1]

```
List = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
newList = [x * 2 for x in List if x % 2 == 0]  
print(newList)
```

[Code2]

함수



- 함수는 입력값 -> 계산 -> 결과값 으로 구성됩니다. $f(x) = 2x$ 라는 함수를 예로 들면, 입력값($x = 2$)를 주면, 이를 계산($2 * 2 = 4$)하고, 그 결과값(4)을 전달하게 됩니다. 이 때, f 를 함수명, x 와 같이 함수에 전달하는 변수를 매개변수라 합니다. 파이썬에서는 오른쪽의 구조로 함수를 정의합니다. (매개변수와 결과값은 상황에 따라 존재하지 않을 수도 있습니다).
- 함수는 주로 같은 계산을 여러 번 수행해야 할 때 사용됩니다. 예를 들어 세 개의 분반 성적의 평균을 각각 계산하려고 할 때, 세 분반 성적의 평균을 계산하는 코드를 직접 작성할 수도 있지만, 매개변수로 성적 리스트를 받고, 평균을 계산하여 반환하는 함수를 만들어 보다 편리하게 계산할 수 있습니다.
- 오른쪽의 코드는 리스트를 입력받아 그 평균을 반환하는 함수를 이용한 예제입니다.

```
def 함수이름(매개변수):  
    수행할 문장  
    ...  
    return 결과값
```

```
def average(score_list):  
    Sum = 0  
    for score in score_list:  
        Sum += score  
    return Sum / len(score_list)  
  
List = [1, 2, 3, 4, 5]  
print(average())
```

함수



- 연습 문제2
- 세 분반의 성적이 주어질 때, 리스트를 매개변수로 받아 평균값(average), 중앙값(median), 최대값(maximum), 최소값(minimum)을 계산하는 함수를 각각 만드세요.
- 리스트와 분반 번호가 입력으로 주어질 때, 위에서 만든 함수를 이용하여 분반 성적 정보를 출력하는 함수(print_info)를 만드세요 (출력 형식은 아래와 같습니다.)

```
(분반 번호)분반  
학생 수 : (학생 수)명  
평균 점수 : (평균 점수)점  
중앙값 : (중앙값)점  
최고 점수 : (최고 점수)점, 최저 점수 : (최저 점수)점
```

- 평균 점수는 마찬가지로 소수점 2자리까지 반올림하여 출력합니다.
- 1번째 줄에는 1분반 학생의 성적, 2번째 줄에는 2분반 학생의 성적, 3번째 줄에는 3분반 학생의 성적이 주어집니다. 세 분반의 성적을 입력으로 받아 위 출력형식에 맞게 분반 성적 정보를 출력하면 됩니다.
(줄 단위가 아니라 띄어쓰기 단위로 입력이 주어지는 문제는 split 함수를 사용해서 문장 전체를 입력받고, 띄어쓰기 단위로 나눠 리스트에 저장할 수 있습니다.)
- 중앙값은 학생 수 N이 홀수인 경우, $(N + 1) / 2$ 번째 학생의 점수, N이 짝수인 경우, $N / 2$ 번째 학생과 $(N / 2) + 1$ 번째 학생의 성적의 평균으로 정의합니다.

Example Input



```
98 75 68 45 78
68 95 74 82 46 48
91 87 82 76 43 66 71
```

Example Output



1분반

학생 수 : 5명

평균 점수 : 72.80점

중앙값 : 75.00점

최고 점수 : 98점, 최저 점수 : 45점

2분반

학생 수 : 6명

평균 점수 : 68.67점

중앙값 : 70.50점

최고 점수 : 95점, 최저 점수 : 46점

3분반

학생 수 : 7명

평균 점수 : 73.71점

중앙값 : 76.00점

최고 점수 : 91점, 최저 점수 : 43점