

Ch01_게임_시작

2020년 4월 7일

1 파이썬과 놀기 (맞보기)

파이썬과 처음 만나는 시간입니다. 간단하게 파이썬 문법의 이모저모를 ‘쭉’ 살펴 보겠습니다. 이리저리 돌아다니면서 파이썬의 기초 문법을 익히도록 하겠습니다. 2-3일 동안에 아래 내용을 쭉 따라 해 보세요. 단계마다 마지막에 퀴즈도 나갑니다. 재미나게 퀴즈도 풀면서, 내가 정말 이해하고 넘어가는지를 한번 체크 해 보세요.

※ 새로운 단어 : 자료형, 연산자, 메소드, 빌트인 함수

1.1 배울 내용

목표 :

- 1) 자료에는 여러가지 종류(type)가 있습니다.
- 2) 각 자료형 마다 연산자가 있습니다.
- 3) 각 자료형 마다 메소드가 존재합니다.
- 4) 빌트인 함수(표준함수)가 있습니다.

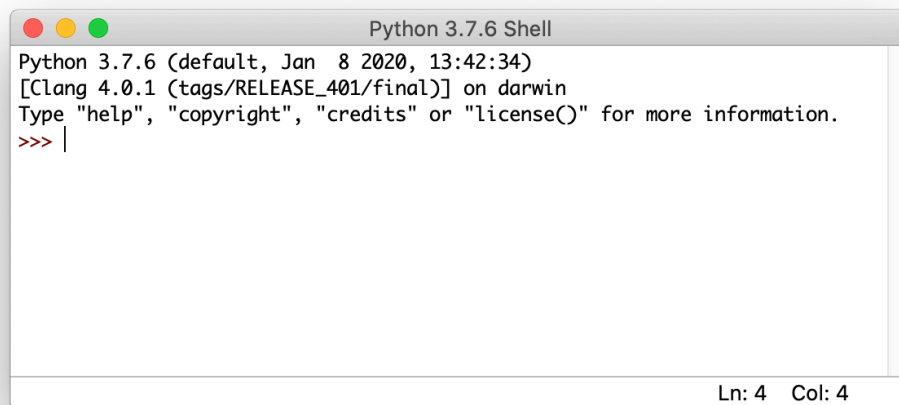
구분	배울 내용	함수	자료형	연산자
실행 문자	IDLE 사용법 print,type, 따옴표, 연산자, 주석	print(),type()	str	+, *
숫자	print,type, 연산자, 숫자와 문자 차이, 에러 이해	print(),type()	int,float	+, -, *, /, //, %, +=, ==
변수개념 이해				
집짓기	단독, 아파트 차이	len()	list	
연산자	자료형과 연산자			
기초 함수		dir(),help()		
제어문	for, while, if continue, break	range(),enumerate()list()		in

구분	배울 내용	함수	자료형	연산자
파일에 명령어 저장, 실행 실습 : 과제 수행하기				

2 실행

파이썬의 기본 IDE인 ‘아이들(IDLE)’ 단축키를 클릭하거나, 터미널에서 `idle` 을 입력하면, 아래 화면이 등장합니다. 이를 파이썬 셸(shell)이라고 합니다. 기호 `>>>` 는 파이썬의 명령 프롬프트입니다. 모든 명령은 `>>>` 이후에 입력하게 됩니다. 인터랙티브하게 셸과 주거나 받거나 하면서 프로그램을 만들 수 있습니다. 빠져 나올 때는 `exit()` 입력합니다.

제 주위에 계신분들은 주로 ‘주피터 노트북’을 많이 사용하고 있습니다만, 이번 챕터에서는 가장 기본이 되는 IDLE 부터 시작해 보도록 하겠습니다. 각자 선호하는 IDE 를 선택하셔서 사용하셔도 됩니다.



참고로 현재 사용중인 파이썬 버전을 확인하기 위하여 아래와 같이 입력해 보세요.

```
In [1]: import sys
        sys.version_info

Out[1]: sys.version_info(major=3, minor=7, micro=6, releaselevel='final', serial=0)
```

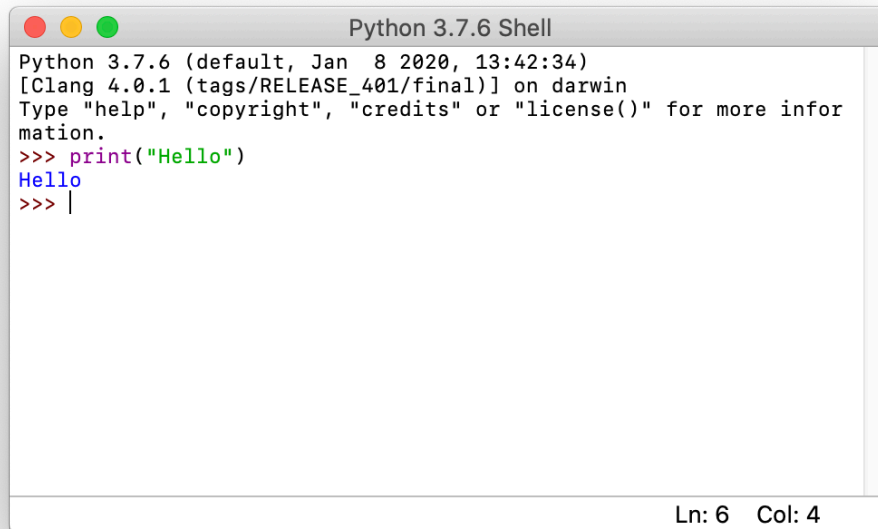
3 문자

파이썬 IDLE 또는 주피터 노트북을 실행하고 난 다음 제일 먼저 뭐 부터 해볼까요. 많은 분들이 ‘Hello’ 문자를 모니터에 출력하는 작업을 합니다. 우리도 이런 전통을 즐기면서, 한번 해 보겠습니다.

3.1 프린트 방법

먼저, 아래와 같은 명령어를 입력해 봅시다.

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

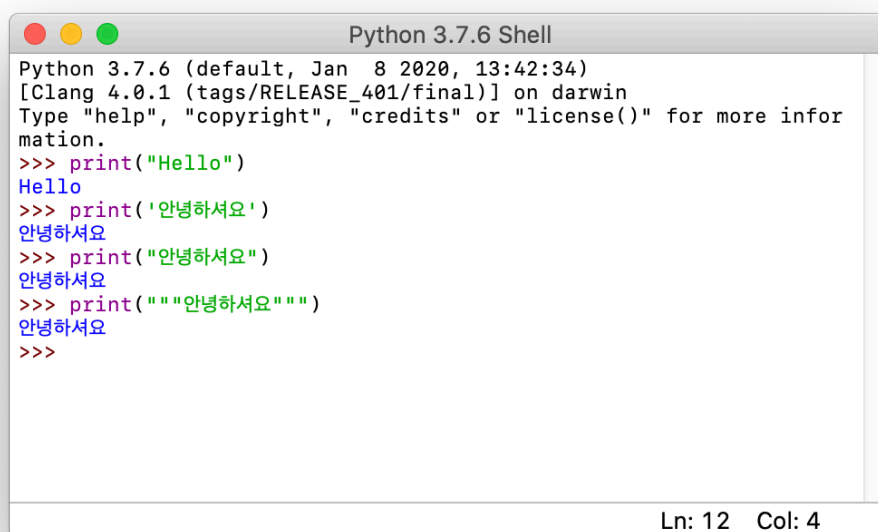


```
Python 3.7.6 Shell
Python 3.7.6 (default, Jan 8 2020, 13:42:34)
[Clang 4.0.1 (tags/RELEASE_401/final)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello")
Hello
>>> |
```

Ln: 6 Col: 4

파이썬에서 문자를 표시할 때는, 작은따옴표(""), 큰따옴표(“,”), 세겹따옴표(“”“,””)를 모두 사용할 수 있습니다.

```
print('안녕하세요')
print("안녕하세요")
print("""안녕하세요""")
```



```
Python 3.7.6 Shell
Python 3.7.6 (default, Jan 8 2020, 13:42:34)
[Clang 4.0.1 (tags/RELEASE_401/final)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello")
Hello
>>> print('안녕하세요')
안녕하세요
>>> print("안녕하세요")
안녕하세요
>>> print("""안녕하세요""")
안녕하세요
>>>
```

Ln: 12 Col: 4

여기까지 성공했다면, 입문자 과정의 1/4을 완성하신 것입니다. 축하합니다.

3.2 문자의 타입

'타입(type)'이라는 것을 설명합니다. 처음 부터 타입 이야기를 하는 것이 이해하기 힘들 수 있는데, 모든 데이터는 타입을 가집니다. 그래서 타입을 알아야 하는데, 타입의 종류가 많습니다. 처음에는 어려울 수 있지만, 조금 지나면 그냥 알게 됩니다.

파이썬에서 제공하는 `type()` 함수를 이용하면, 해당 데이터의 종류를 알 수 있습니다. 아래와 같이 한번 해 봅시다.

```
type('abc')
```

문자는 `str` 타입입니다. 이것은 문자열(string)을 나타냅니다.

```
In [2]: type('abc')
```

```
Out[2]: str
```



```
Python 3.7.6 Shell
>>>
>>>|
>>>
>>>
>>>
>>>
>>>
>>> type('abc')
<class 'str'>
>>>
>>>
Ln: 50 Col: 3
```

3.3 문자 이어주기 : 연산자(+, *)

'연산자(operator)'라는 단어가 낯설 수도 있지만, 일단 진행해 보겠습니다. 수학 시간에 배운 연산자와 비슷하지만, 컴퓨터 키보드에 등장하는 여러가지 기호 들에는 어떤 특별한 의미를 부여해 놓았습니다. 그런 의미를 하나씩 배워 나갈 것입니다.

프린트 명령어를 다양하게 알아보겠습니다.

- 문자만 출력하기

```
print("I am Dr. Who !")
```

- 문자 + 문자

```
print("I am" + "Dr. Who !")
```

- 문자 * 정수

```
print("ABC"*10)
```

- 플러스(+) : 문자를 연결시켜주는 기능입니다.
- 스타(*) : 문자열을 *배 만들어 줍니다.
- 마이너스(-)는 ? 그런데 문자를 빼는 기능은 지원하지 않습니다. 각자가 테스트 해 보세요.

In [3]: # Example

```
print("I am Dr. Who !")
print("I am " + "Dr. Who !") # + 기능
print('abc'*3)               # * 기능
```

```
I am Dr. Who !
I am Dr. Who !
abccabccabc
```

- 아무런 연산자 없이 그냥 문자열만 나열하면 문자를 자동으로 연결시켜 줍니다.

In [4]: print('Red' 'Green' 'Blue')

```
RedGreenBlue
```

3.4 [Quiz]

[Q-1] 다음 중 맞지 않는 방법은 ? (문자열; 따옴표)

- ① `print('\"해맞이 도서관\"에 가다')`
- ② `print(\"\"해맞이 도서관\"에 가다\")`
- ③ `print(\"\"\"해맞이 도서관\"에 가다\"\"\")`

[Q-2] 다음의 결과는 ? (문자열; 연산자)

```
print("*****5")
```

- ① *5 ② ***** ③ *55555

4 숫자(정수, 실수)

이번에는 숫자를 출력하는 방법을 알아 보겠습니다. 숫자는 정수, 실수, 복소수 등이 있습니다. 파이썬은 이런 모든 숫자를 다룹니다. 그런데 우리는 정수, 실수만 다룰 것입니다.

4.1 프린트 방법

정수는 플러스, 마이너스 값을 가지며, 실수는 소수점을 가집니다.

- 정수 : 0, -1, 1, ...
- 실수 : 1.0, 1., 0.1, 소수점이 있는것

```
In [5]: # (example)
```

```
print(100) # 이건 정수이죠.  
print(0.8) # 이건 실수 이죠.  
print(0.)  # 이것도 실수 이죠.  
print(1.)
```

```
100
```

```
0.8
```

```
0.0
```

```
1.0
```

4.2 type 확인

앞에서 문자 타입 확인할 때 사용한 `type()` 함수를 이번에도 사용해 보겠습니다. `int` 는 정수 (integer) 이며, `float` 은 실수 입니다.

(참고) C언어에서 실수는 `float`, `double` 타입이 있는데, 파이썬은 `float` 만 있습니다. 여러분이 사용 중인, 라임을 얼핏보니, I, D로 표시된 부분이 있는데, I는 정수, D는 `double` 인 실수를 나타내죠, 같은 용어입니다.

```
In [6]: type(1)
```

```
Out[6]: int
```

```
In [7]: type(0.)
```

```
Out[7]: float
```

4.3 연산자

이제 숫자의 연산자를 알아 보겠습니다. 더하기, 빼기, 곱하기, 나누기 등등에 대해서 살펴 보겠습니다. 아마도 뺄, 나머지, 지수, 비교, 할당 연산자는 낫설수 있습니다. 익숙해 지면 쉬워지니, 한번 더 눈길을 보내 주세요

구분	기호	설명
더하기	+	-
빼기	-	-
곱하기	*	곱하기 \times 가 키보드에 없습니다. 그래서 * 을 사용합니다
나누기	/	나누기 \div 도 키보드에 없습니다. 그래서 / 을 사용합니다
몫연산자	//	몫만 돌려주고 나머지는 버리는 연산자 $x // y$

구분	기호	설명
나머지연산자	%	몫은 버리고 나머지만 돌려주는 연산자 $x \% y$
지수 연산자	**	$a**b$ (a의 지수 b)
비교 연산자	==	
할당 연산자	+=	

사칙연산

여러가지 연산자 중에서 먼저 사칙연산 연산자인 더하기, 빼기, 곱하기, 나누기를 하는 방법 부터 알아보겠습니다. 곱하기, 나누기 기호를 눈여겨 봐주세요.

```
In [8]: print(1+2)
        print(3-1)
```

곱하기는 \times 가 키보드에 없습니다. 그래서 '*' 을 사용합니다. 나누기 \div 도 키보드에 없습니다. 그래서 '/' 을 사용합니다

```
In [9]: print(3*9)
        print(27/2)
```

27

13.5

※ 참고로, 파이썬 2.* 에서는 정수 나누기 정수는 정수로 취급하여, 나머지가 출력이 되지 않았습니다. 그러나 파이썬 3.* 에서 이 문제가 해결되었습니다.

자, 이제 변수명을 이용해서 더하기 빼기를 하는 방법을 배워보겠습니다.

```
a = 100
b = 10.1
c = a+b
```

```
print(c)
110.1
```

```
In [10]: # 참고
```

```
a = 'abc'
b = '123' # 외형상 숫자같지만, 실제로는 문자임

print(type(a))
print(type(b))
```

```
<class 'str'>
<class 'str'>
```

[연습] 파이썬 명령창에서 다음을 계산하고 결과값을 제시 해보세요

① 22×10 ② $33 \div 20$

// (몫 연산자)

몫을 돌려줍니다. 이게 언제 쓰일까 쉽지만, 종종 사용합니다. 아래에서 10 나누기 3을 해서 몫만 구하는 방법을 표시해 놓았습니다.

```
In [11]: 10//3
```

```
Out[11]: 3
```

```
In [12]: 10.1//3
```

```
Out[12]: 3.0
```

% (나머지 연산자)

나머지 연산자인 퍼센트 기호도 언제 사용하나 싶겠지만, 이게 종종 사용합니다. 이런 것이 있구나 하고 익혀 둡시다.

```
In [13]: 10%3 # 나머지를 구합니다.
```

```
Out[13]: 1
```

```
In [14]: 11%3
```

```
Out[14]: 2
```

```
In [15]: 11.1%3
```

```
Out[15]: 2.0999999999999996
```

** (지수 연산자)

- 지수 값을 구해 줍니다.

```
In [16]: 2**3
```

```
Out[16]: 8
```

```
In [17]: 2.1**2
```

```
Out[17]: 4.41
```

+= (할당 연산자)

- $x += y$ 는 $x = x + y$ 를 나타냅니다.
- 아래 예를 보면 이해가 될 것입니다. 처음에는 낯설지만, 나중에는 이런 표기법이 편해지게 됩니다. 매주 자주 사용하게 될 연산자입니다.

```
In [18]: a = 100
```

```
        a = a + 200
```

```
        print(a)
```


300

위의 예와 아래는 같은 결과를 보이고 있다.

```
In [19]: a = 100
         a += 200
         print(a)
```

300

== (비교 연산자)

$x == y$ 는 x 와 y 의 값이 같은지를 비교하여 같으면 True, 다르면 False 를 돌려줍니다.

```
In [20]: a = 10
         b = 20
         c = 10

         print(a == b)
         print(a == c)
```

False

True

4.4 숫자와 문자의 차이점

이게 중요합니다. 겉으로 보기에는 숫자로 보여도, 컴퓨터가 인식하는 것은 따옴표 사이에 있는 숫자는 글자로 인식을 합니다.

```
print(10)    #<-- 숫자로 인식합니다.
10
```

```
print('10')  #<-- 글자로 인식합니다.
10
```

※ 참고, 엑셀 파일의 경우도, 시트에서 숫자로 보이지만, 왼쪽에 있으면 문자, 오른쪽에 있으면 숫자

```
In [21]: # Example

         print('100' + '9')
         print(1000 + 9 )
```

1009

1009

```
In [22]: type('100'+'9')
```

```
Out[22]: str
```

```
In [23]: type(1000+9)
```

```
Out[23]: int
```

4.5 형이 다르면 에러 발생

타입이 다른면, 연산자 적용할 수 없습니다. 아래와 같이 str과 int 형을 덧셈(+) 연산자로 이어주면, 에러가 발생합니다.

```
'10' + 10
```

```
TypeError: can only concatenate str (not "int") to str
```

타입이 다른 것을 덧셈(+) 연산자를 적용시키면, 에러가 발생합니다. 그러기에 Type을 거의 일치해 줘야 합니다. 정수, 실수 타입은 같이 사용 가능하지만, 문자, 숫자는 같이 사용할 수 없습니다.

4.6 [Quiz]

[Q] 다음 결과는 ?

```
type(1000+9)
```

①int ② str ③ list

[Q] 다음 결과는 ?

```
type('100'+'9')
```

①int ② str ③ list

[Q] 다음의 결과를 맞춰 보세요.

```
a = 100
```

```
b = 'km'
```

```
c = 1.
```

```
print(a, b, c)
```

① 100, km, 1.0,

② km 100 1.0,

③ 100 km 1.0

[Q] 다음 결과는 무엇일까요?(형일치)

10 + '1'

① 11,

② 에러가 발생한다

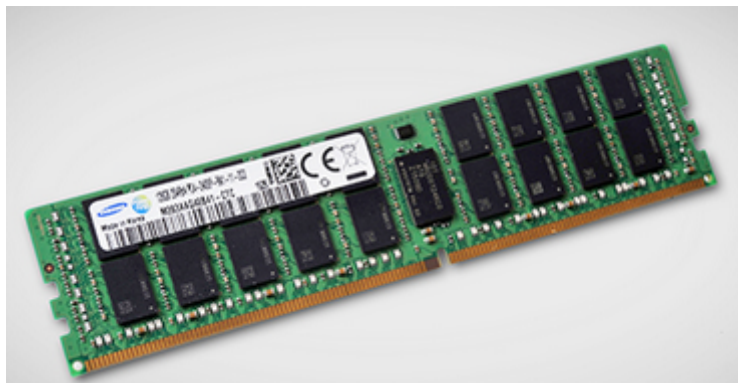
③ 10

5 변수 개념

- 변수(variables)가 뭔지를 소개해 주려고 합니다.
- 컴퓨터 메모리(RAM)에 만든 집에 비유하여 설명을 하고자 합니다.

5.1 컴퓨터 구성 부품

- 컴퓨터 본체는 메모리 + 하드디스크 + CPU + 메인보드 + 그래픽 카드 등등으로 구성되어 있습니다.
- 이런 부품을 모아서 컴퓨터 본체를 만들 수가 있습니다.
- 지금 우리는 이 부품 중에서 램(RAM)에 대해서 살펴볼 것입니다.
- 램이 변수들이 들어갈 방을 저장하는 곳이기 때문입니다.



삼성전자 128 GB 램

<http://www.samsung.com/sec/news/local/samsung-electronics-the-worlds-first-d-128-gigabytes-of-ram-modules-production>

5.2 램 위에 집짓기

- 프로그램을 돌리면, 일단 모든 정보가 램(Ram) 안으로 들어갑니다. 이곳에 정보를 넣을 집을 만들어 줍니다. 이집을 어떻게 짓는가가 중요합니다. 아래와 같이 코딩을 하게되면,

```
name1=100
```

```
name2='abc'
```

- 100 이라는 숫자가 메모리 상에 생성됩니다. 그리고 이 생성된 장소의 이름이 name1이 됩니다. 이제, name1이라는 변수명만 호출하면 그 속에 담겨있는 100이라는 숫자를 호출할 수가 있게 됩니다.

6 집짓기 (단독, 아파트)

램위에 집을 지어 보겠습니다. 데이터의 다양한 구조를 단독집, 집단집(아파트)에 비유하여 설명하였습니다.

6.1 단독집

변수의 종류도 다양한데, 이를 어떻게 설명할까 하다가 메모리에 지어진 집에 비유해 보았습니다. 사람들이 살아가는 집의 종류가 다양합니다. 단독집, 연립, 아파트 등의 집형태가 있습니다. 그리고 이런 집들이 모여서 마을과 도시를 이룹니다. 변수들 집도 비슷합니다. 먼저, 가장 기본이 되는 단독집을 짓는 방법을 알아보겠습니다.

메모리에 단독집 a를 만듭니다. 이때 a를 변수라고 불러줍니다. 집이름은 문자(영어, 한글) 또는 언더스코어(_)로 시작해야해야 합니다. 그러나 특수문자는 안되니 조심해야 합니다.

```
a = 100
```

위와 같이하면, 램 내부에 a라는 집이 생깁니다. 그리고 그 집에는 100 라는 숫자가 들어갑니다. 그래서 이제 print 해주면, 글자 a가 아니라 a 집에 들어가 있는 100을 출력해 주게 됩니다.

```
print(a)
100
```

집에다 값 넣기

메모리에 지은 집 a에 100과 200을 차례대로 넣어보시다. 결과 값은 제일 마지막에 넣은 값만 저장하게 됩니다.

```
a = 100
a = 200
print(a)
```

```
200
```

변수를 여러 개 만들어 보겠습니다. 그리고 이들 변수를 서로 더해 보겠습니다.

```
a = 100
b = 200
c = 300
print(a+b+c)
```

```
600
```

6.2 아파트

이제는 메모리에 집을 짓는데, 이제 단독주택이 아니라, 아파트를 만들어 보겠습니다.

아파트 만들기 : 울타리 조심 { } [] ()

아파트는 울타리를 갖고 있습니다. 사각형 울타리([]) (브래킷(bracket))로 묶어 줍니다. 참고로 (과) 은 튜플 데이터를 만드는 데 나중에 소개하도록 하겠습니다.

```
a = [100,200,300]
print(a)
```

```
[100, 200, 300]
```

형 확인

type() 함수를 이용하여, 아파트의 타입 을 확인해보면, 'list' 타입인 것을 알 수 있습니다. 즉 리스트는 여러개의 값을 저장할 수 있는 데이터 구조입니다. 향후 이 리스트 타입을 매우 자주 사용하게 될 것입니다.

```
In [24]: a = [100,200,300]
         type(a)
```

```
Out[24]: list
```

len() 함수 소개

len() 함수는 리스트 내에 몇개의 요소가 들어 있는지를 알려줍니다.

```
In [25]: a = [1,2,3]
         len(a)
```

```
Out[25]: 3
```

문자열의 크기를 살펴 봅시다.

```
In [26]: a = 'abcd'
         len(a)
```

```
Out[26]: 4
```

아파트 찾아가기 (리스트 번호 읽고 이해하기)

- 0 부터 시작 (1부터 시작이 아님)

아파트는 0층부터 집이 있다고 생각합시다. a 아파트 0층에 사는 아파트 주민을 불러내려면, a 아파트 0층에 사는 사람이라고 지정해 줘야 합니다. 즉, a[0] 이라고 호출해 주면, 100이 불러지게 됩니다. a[1] 은 두번째 집을 불러오게 되며, a[2] 는 세번째 집을 불러오게 됩니다.

```
print a[0]
print a[1]
print a[2]
```

100
200
300

아파트 단지

아파트 구조가 조금 더 복잡해 지는 경우를 생각해 봅시다. 첫 동에는 'Hong', 'Park', 'Sohn'이 살고 있습니다. 다음 동에는 'Go', 'Lee', 'Kim'이 살고 있습니다. 이때 각각의 사람들에게 접근하는 방법은 아래와 같습니다.

변수이름 [동번호][층번호]

```
a = [ ['Hong', 'Park', 'Sohn'], ['Go', 'Lee', 'Kim'] ]
```

```
print(a[0])    # 0동 전체를 불러옵니다.
```

```
print(a[0][0]) # 0동 0층에 살고 있는 사람을 불러옵니다.
```

뒤에서부터 읽기 (-1의 의미?)

- -1은 제일 끝을 의미합니다. 그런데 a 아파트의 마지막 층에 있는 사람을 어떻게 불러 올까요 ?

```
a = [100,200,300,500,400]
```

이 있다고 합시다. 이때 a[4]라고 해도 됩니다. 그런데 아래의 리스트 a에서 제일 마지막 값을 어떻게 불러올까요.

```
a = [1,3,1,1,3,111,333,1113,3333,55555,1111,881,200,100,10]
```

- a안에 몇 개 있는지를 하나하나 세어서 그 갯수에서 1을 빼 주면 마지막 집에 들어있는 값을 구할 수 있습니다. 그런데 이런 방법은 시간이 많이 소요되고, 좀 헷갈립니다. 그래서 “뒤에서 부터 세어 주세요” 라고 파이썬에게 명령을 내리고 싶습니다. 이럴 때 -1을 사용하면 됩니다.

```
print(a[-1]) ---> 10
```

```
print(a[-2]) ---> 100
```

```
print(a[-3]) ---> 200
```

- -2는 -1보다 1 더 앞에 있는 것입니다.
- -3은 -2보다 1 더 앞에 있는 것입니다.
- [Example] 제일 마지막 요소를 알아내는 방법

```
a = [1,3,1,1,3,111,333,1113,3333,55555,1111,881,200,100,10]
```

```
print(a[-1]) #---> 10
```

```
print(a[-2]) #---> 100
```

```
print(a[-3]) #---> 200
```

6.3 [Quiz] 리스트 이해

[Q] 다음의 값은 ?

```
a = [1,10,100]
print(len(a)*10)
```

① 3 ② 30 ③ 300

[Q] 아래 결과를 맞춰 보세요.

```
a = [100,200,300,500,400]
print(a[0] + a[1])
```

① 100 ② 300 ③ 200

[Q] 아래 결과를 맞춰 보세요.

```
b = [[1,100,10],[2,3]]
print(b[0][1] + b[1][0])
```

① 102 ② 103 ③ 104

[Q] 리스트의 -1의 의미를 이해했는지 확인해 볼까요. 아래 결과값을 맞춰 보세요.

```
data = [100,200,10,20,111,222,333,111,333,1111,0,-1,11,1]
print(data[-1])
```

① 1 ② 11 ③ 111

[Q] 리스트의 -1의 의미를 이해했는지 확인해 볼까요. 아래 결과값을 맞춰 보세요.

```
data=[1,3,5,7]
print(data[-1] + data[-3])
```

① 0 ② 10 ③ 22

7 자료형 vs 연산자

[<https://docs.python.org/3/library/operator.html#module-operator>]

- 기본적인 더하기 빼기 이외의 연산자들의 목록을 보고싶으면 아래 사이트 방문하시기 바랍니다.

구분	문자(str)	숫자(int, float)	리스트(list)
+	O	O	O

구분	문자(str)	숫자(int, float)	리스트(list)
-	-	O	-
	O	O	-
/	-	O	-
// (몫연산자)	-	O	-
% (나머지연산자)	-	O	-
(지수 연산자)	-	O	-
==(비교 연산자)	O	O	O
+=(할당 연산자)	-	O	O

- 몫연산자(몫만 돌려주고 나머지는 버리는 연산자) $x // y$
- 나머지연산자(몫은 버리고 나머지만 돌려주는 연산자) $x \% y$
- 지수 연산자 $a**b$ (a의 지수 b)

8 기초함수 소개

앞에서 아래 3개 함수에 대해 간단하게 살펴보았습니다.

- `print()`
- `type()`
- `len()`

이제는 다음 함수의 사용법을 살펴 보겠습니다.

- `dir()`
- `help()`
- `list()`
- `range()` : 다음 섹션에서
- `enumerate()` : 다음 섹션에서

8.1 `dir()`, `help()`

- 객체에 대해서 잘 모를 때는 `dir()`, `help()` 함수를 불러 주세요.

```
a = 100
```

```
dir(a)
```

```
help(a)
```

```
In [27]: a=100
```

```
print(dir(a))
```

```
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__', '__delattr__', '__dir__',
```

```
In [28]: # print(help(a))
```


8.2 list()

- 문자열을 리스트형으로 바꿀수 있습니다.

```
In [29]: a = list('abc')
         print(a)
         print(type(a))
```

```
['a', 'b', 'c']
<class 'list'>
```

9 제어(for, while, if 문)

9.1 for 문

반복문이라고 합니다. for 가 영어로는 .. 하는 동안 이라는 뜻이 있습니다. 무엇을 하는 동안 “반복하세요”라는 기능을 수행합니다.

for ... in : 의 기능은 리스트의 값을 차례대로 임의의 변수에게 전달해줍니다.

for문을 만들 때는 다음에 주의하세요.

- 첫번째 : for 와 in 사이에 들어가는 변수명은 마음대로 하셔도 됩니다. 여기서는 each를 만들었습니다.
- 두번째 : in 다음에 리스트, 튜플 등 이터러블 한 시퀀스를 입력 합니다. 여기서는 [100,200]을 입력했습니다.
- 세번째 : for 가 있는 줄 제일 끝에는 콜론(:)을 적어줘야 합니다.
- 네번째 : for문안에 속하는 속하지 않는가는 들여쓰기로 구분합니다. 들여쓰기 중요합니다.
- 예제 1

```
In [30]: for each in [100,200]: # 마지막에 콜론(:) 적어 줌
         print(each) #
```

```
100
200
```

- 예제 2

```
In [31]: for a in [100,200]: # for 와 in 사이에 들어가는 변수명은 ?
         print(a) #
         print("----")
```

```
100
----
200
----
```

- for문의 작동 순서를 알아 봅시다.

```
In [32]: for each in [10,20,30]:    # each=1, each=2, each=3\n",
        print('One', each)
        print('Two', each*each)
        print("*"*50)
```

```
One 10
Two 100
*****
One 20
Two 400
*****
One 30
Two 900
*****
```

- 예제 3

```
In [33]: for each in ['choi','sohn','noh','han','park']:
        print("Family name :", each)
```

```
Family name : choi
Family name : sohn
Family name : noh
Family name : han
Family name : park
```

- 예제 4 다음은 어떻게 작동하는지를 생각해 보세요. len() 함수는 문자 갯수를 알려줍니다.

```
In [34]: words = ['아침', '햇살이', '좋아요']
        for w in words:
            print(f"{w}, {len(w)}")
```

```
아침, 2
햇살이, 3
좋아요, 3
```

for 문 중첩

- for문 속에 for을 다시 만들 수도 있습니다.

```
In [35]: for i in range(2,4):
        print("-"*10)
```

```
for j in range(1,5):
    print(f'{i}*{j}={i*j}')
```

2*1=2

2*2=4

2*3=6

2*4=8

3*1=3

3*2=6

3*3=9

3*4=12

in 연산자

멤버인지 아닌지 알려줍니다.

```
In [36]: 'a' in ['a','b','c']
```

```
Out[36]: True
```

```
In [37]: 1 in range(10)
```

```
Out[37]: True
```

```
In [38]: 1000 in range(10)
```

```
Out[38]: False
```

범위 함수 range()

for 문과 함께 많이 사용되는 함수가 있습니다. range() 함수입니다. range(숫자) 형식을 취하며, 숫자 하나만 입력하면 0부터 입력한 숫자-1 까지의 정수를 리턴합니다. 즉 range(5)는 0,1,2,3,4까지 값을 가집니다. 5까지가 아니라는 점을 주목해주세요.

```
In [39]: for each in range(5):
          print(each)
```

0

1

2

3

4

enumerate()

for 문과 함께 많이 사용되는 함수에는 enumerate()가 있습니다. 인덱스도 함께 리턴해 줍니다.

```
In [40]: names = ['hong', 'kim', 'choi', 'lee']
         for each in names:
             print(each)
```

```
hong
kim
choi
lee
```

```
In [41]: for index, each in enumerate(names):
         print(f"{index}, {each}")
```

```
0, hong
1, kim
2, choi
3, lee
```

9.2 while 문

조건이 참이면 계속 그 내용을 수행하게 됩니다.

```
while 조건:
    실행1
```

아래에서 조건이 참이면 계속 반복하는 순환문을 볼 수 있는데요, 증감식을 두어서 일정 조건을 벗어날 수 있도록 하였습니다.

```
In [42]: a = 5
```

```
while a>1:
    print(f">>> {a}")
    a -= 1
```

```
>>> 5
>>> 4
>>> 3
>>> 2
```

break

순환문에 등장하는 아래 3가지 명령어가 있습니다. 이중에서 break 사용법만 살펴 봅시다.

- continue : 신중
 - break : 적극. 루프 탈출한다 (루프가 많으면 가장 안쪽 먼저 탈출한다)
 - pass : 침묵.
- break는 루프를 탈출하게 합니다. 즉 a가 5가 될때 순환문을 탈출합니다. 아래에서 *이 5개까지 출력된 것을 확인하세요.

```
In [43]: a = 10
```

```
while a > 1:
    if a == 5 :
        print(f">>> a = {a}")
        break
    a -= 1
    print("*"*a,a)

***** 9
***** 8
***** 7
***** 6
***** 5
>>> a = 5
```

9.3 if 문

if의 의미는 if 다음의 x가 참이라면 if문 아래에 있는 문장을 실행하라는 의미입니다. 만약 그렇지 않을때는 else 이하를 실행합니다. 이때 0, 빈리스트 등은 False 입니다.

```
if x :
    액션
else:
    액션
```

- 예제 1

```
In [44]: if True :
          print("apple > orange")
```

apple > orange

- 예제 2

```
In [45]: a = True
```

```
if a :  
    print("a is True")
```

```
a is True
```

Reference ***

<https://docs.python.org/3/library/stdtypes.html>

<https://docs.python.org/3/tutorial/controlflow.html#for-statements>

10 파일 실행

이젠 마지막 단계입니다. IDLE 메뉴에서 파이썬 소스코드를 저장하고, 파일을 불러와서 실행하는 방법을 알아보도록 하겠습니다.

10.1 파일 저장과 종료

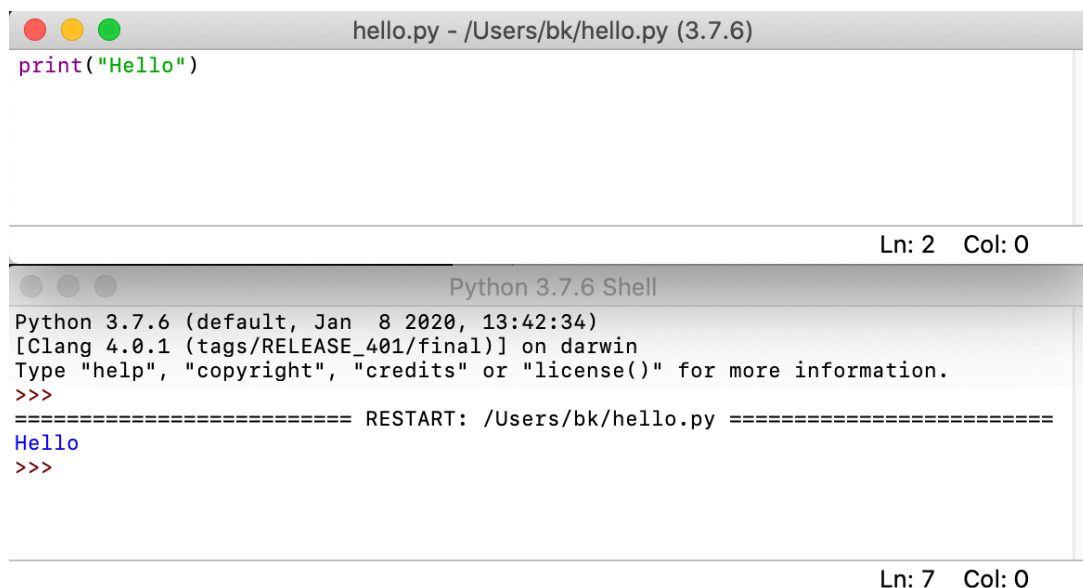
IDLE의 File 메뉴에서 New File 을 선택하여 비어있는 파일을 만든 다음에 아래와 같이 입력해 보세요.

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

그런 다음에 저장하세요. File 메뉴의 Save를 클릭하십시오. 파일명을 입력하라는 윈도우가 등장하면 파일명을 입력하십시오. 파일 확장자는 .py 또는 .pyw 로 해야 합니다. 보통 *.py로 하면 됩니다. 그럼 파일 저장이 완료되었습니다.

- 확장자 : .py, .pyw

IDLE 창에서 File 메뉴의 Exit을 클릭하면 작업이 종료되면서, IDLE 편집기가 사라집니다..



10.2 파일 불러오기

IDLE을 호출하여, 저장한 파일을 불러옵니다. File 메뉴의 Open을 클릭하셔서, 조금 전에 저장한 파일을 불러옵니다. 그리고 Run 메뉴 아래에 있는 Run module을 클릭하면 실행이 됩니다. 단축키가 F5로 설정된 경우, F5만 클릭하면 실행이 됩니다.

축하합니다. 1장을 마칩니다.