

Bbong 뽑는 Python

1강 출력, 자료형, 입력

권태형@TaeBbong
github.com/TaeBbong

출력

Output

출력 Output

프로그래밍에서 가장 기본적인 코드인 출력
출력 함수의 다양한 사용법을 익혀보자

출력 Output

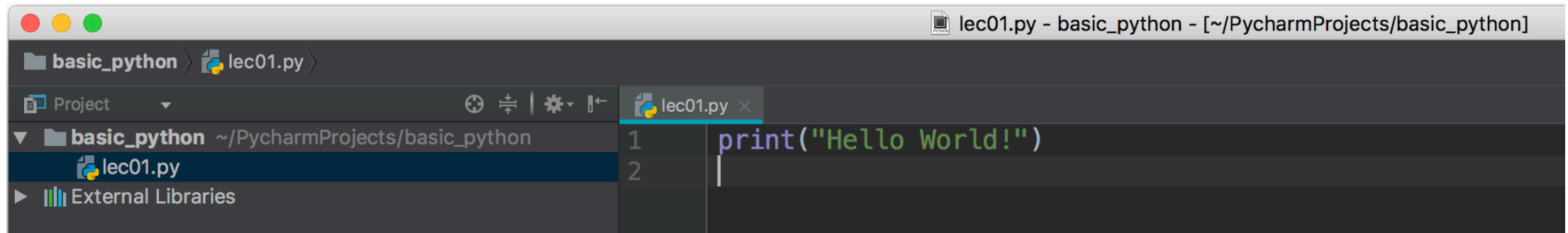
프로그래밍에서 가장 기본적인 코드인 출력
출력 함수의 다양한 사용법을 익혀보자

```
1 Hello World!
```

=> 가장 많이, 처음 보게 되는 문장 Hello World!

출력 Output

1. Open Pycharm, New Project name “basic_python”
2. New Python file “lec01.py”
3. Write follow codes.

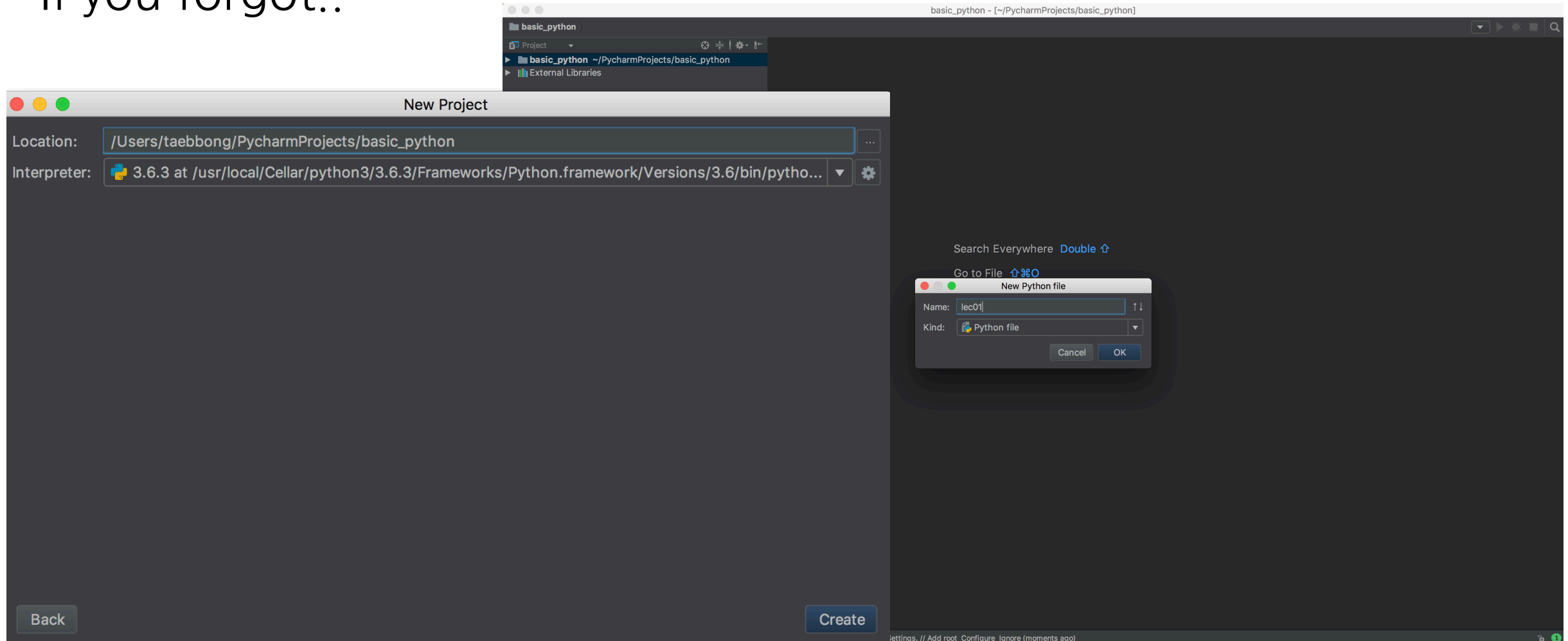


The screenshot shows the PyCharm IDE interface. The title bar indicates the file is 'lec01.py' within a project named 'basic_python' located at '~/PycharmProjects/basic_python'. The left sidebar shows the project structure with 'basic_python' and 'lec01.py' listed. The main editor area displays the following Python code:

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

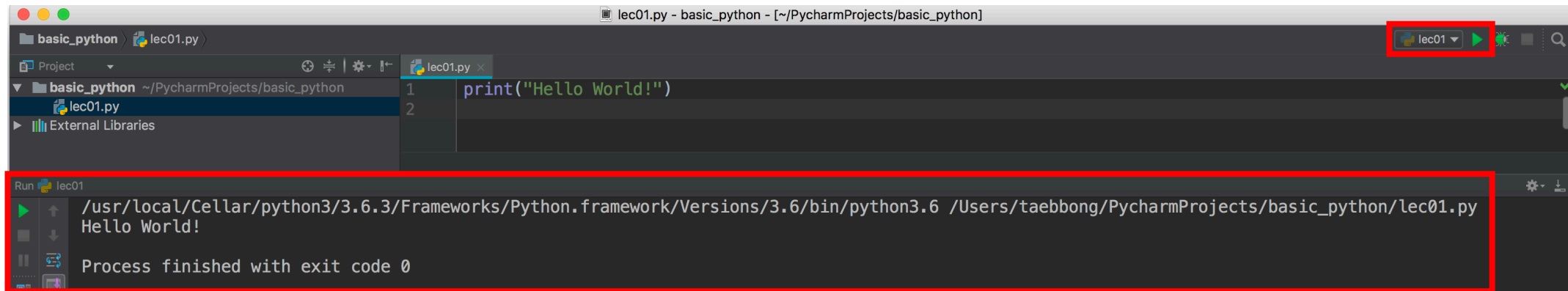
출력 Output

If you forgot..



출력 Output

4. Run!!!
(Run - Run - lec01.py)

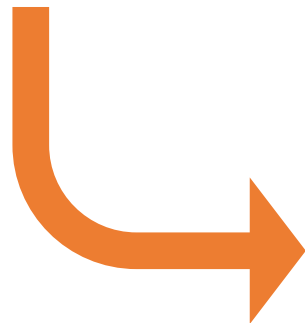


The screenshot shows the PyCharm IDE interface. The top toolbar has a red box around the 'Run' button (a green play icon) and the 'lec01' dropdown menu. The main editor window shows a file named 'lec01.py' with the following code:

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

The bottom 'Run' console is also highlighted with a red box. It shows the command executed and the output:

```
Run lec01  
/usr/local/Cellar/python3/3.6.3/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/taebbong/PycharmProjects/basic_python/lec01.py  
Hello World!  
Process finished with exit code 0
```



Can see results here

출력 Output

출력의 기본 개념

출력은 “화면”에 “내가 원하는 무언가를” 찍는 일

print()은 Python에 내장된 하나의 함수이다. => 함수란? 2~3강 쯤 제대로 배워보자

프로그래밍에서 기본적으로 함수는 “일련한 작업이나 기능의 단위체” 로 정의 가능
그리고 하나 더 알고 가자면 함수는 입력이 있고(parameter), 연산을 해서(function), 반환(return)해준다.

출력 Output

출력하는 방법

- print("Hello!") # Hello!
- print(1004) # 1004
- print(5 + 10) # 15
- print("3" + "A") # 3A
- print('Hello!') # Hello!

기본적으로 출력을 담당하는 print 함수는 () 안에 출력하고 싶은 문자열(String)을 담는다. => 문자열?? 곧 배울테니 걱정 ㄴ ㄴ!

“ ”, ‘ ’ 둘다 문자열을 감싸는 방법! 편한걸로 사용하면 된다. (섞어쓰면 안됨)

또한 print는 변수, 상수를 출력하기도 한다. 변수와 상수란??

출력 Output

변수와 상수, 그리고 연산자

변수: 어떤, 변하는 값을 **저장**하기 위한 그릇

상수: **고정**되어 있는 값(ex: 4, "hello", ..)

Step 1: 내가 어떤 그릇을 만들면, 그릇에 이름을 지을 수 있다. (x, a, ...)

Step 2: 그 그릇에 원하는 값을 저장할 수 있다. (x = 3, a = 'hihi' ...)

Step 3: 그 그릇의 이름을 부르면, 내가 원하는 연산에 사용할 수 있다.

연산자: 변수와 변수끼리 **계산할 때** 쓰는 기호들

+ - * /, //(나눈 몫) *(제곱) %(나머지) -> 이항 연산자

= -> 대입 연산자

출력 Output

너무 많은 이론을 배워버렸다..

실습으로 내용들을 복습해봅시다:)

Q) 내 나이와 아버지의 나이, 어머니의 나이를 각각 변수를 만들어 값을 저장하고,

아버지 나이 - 내 나이 = ‘ ’

어머니 나이 - 내 나이 = ‘ ’

위와 같이 출력 되도록 해보자!

출력 Output

Q1) 부모님과 나

1) 변수 선언

me = 21

dad = 52

mom = 51

2) 출력하기

print("아버지 - 나 = ", dad - me) => print 안에서 여러가지를 출력하고 싶을 땐

print("어머니 - 나 = ", mom - me) => 이렇게 쉼표(,)를 사용해 출력합니다!

* 혹은, print("나는 " + "권태형입니다") 와 같이 +를 이용하기도 함!

출력 Output

Detail: 심화학습

1) 더 많은 연산자들 by 우선순위

** : 제곱

*, /, %, // : 곱하기, 나누기, 나눈 나머지, 나눈 몫

+, - : 더하기, 빼기

<=, >=, <, > : 비교

== : 같다, != 다르다 (*주의할 것! '==' 이 같다는 의미함, '=' 는 대입!)

+=, -= : 대입 연산자(a = a + 3 equals a += 3)

2) 출력 포맷

앞에서 배웠던 + 로 연속 출력, 심표로 연속 출력하는 것 외에 다른 방법도 있음

print("I'm {} and work in {}".format(name, work)) 와 같이 {}로 자리를

잡아놓고, .format 을 사용해서 원하는 변수를 {} 자리에 순서대로 넣어 출력!

출력 Output

HW1

비트코딩 교재 Prob1 ~ Prob 37

각 문제를 풀어서 '1강_HW1_이름' 폴더에 각 문제 풀이 파일을 저장
Slack HW Channel 에 Upload!

자료형과 자료구조

Data Type & Data Structure

자료형 Data Type

자료형이란

데이터가 생겨먹은 모양!

우리가 알고 있는 데이터는??

=> 정수, 실수, 문자(열), ... 이것들이 자료형이다!

기본적으로 같은 자료형끼리만 연산을 해야한다(할 수 있다)
하지만 그렇지 않은 경우도 존재한다.

자료형 Data Type

다른 자료형, 같은 결과 - 강제 형변환

1) 3.5라는 값이 있는데 이 중에 정수 값만 가져오고 싶어요!

=> `round(3.5)`

2) 내가 원하는 정수 값을 문자로 바꿔서 내 이름 사이에 넣고 싶어요!

=> `"권" + str(10) + "태" + str(12) + "형"`

3) 문자로 변해버린 정수를 정수로 다시 바꾸고 싶어요!

=> `int(str(10))`

자료형 Data Type

문자열

문자들의 **나열**!

파이썬에서는 모든 글자들을 문자라고 생각할 수 있어, 모든게 문자열일 수 있다!

ex: “3”, “123123”, “A”, “asdf qwer *&^%” 모두 문자열이다!

파이썬에서는 모든 문자열의 **인덱싱(indexing)**을 지원한다.(0~)

즉, 각 문자가 어느 **위치**에 있는지를 쉽게 접근할 수 있다는 뜻!

ex: s = “나는 권태형 입니다” 에서 ‘나’는 0번째, ‘권’은 3번째 위치에 존재한다.

```
print(s[0]) # 나
```

```
print(s[3]) # 권
```

자료형 Data Type

문자열의 여러 기능

문자열 덧셈, 곱셈

특수한 문자 기호(Escape Code)

문자열 슬라이싱(Slicing)

자료형 Data Type

문자열의 여러 기능

문자열 덧셈, 곱셈

특수한 문자 기호(Escape Code)

문자열 슬라이싱(Slicing)

덧셈: `print("hihi " + "my name is " + "TaeHyung")` # 아하 print의 더하기가 문자열 덧셈이었구나!

곱셈: `print("I love you " * 5)` # I love you I love you I love you ...

자료형 Data Type

문자열의 여러 기능

문자열 덧셈, 곱셈

특수한 문자 기호(Escape Code)

문자열 슬라이싱(Slicing)

Escape Code: 미리 정의해둔 특수한 문자열들

\n: 엔터(줄바꿈)

\t: 탭

\': 작은 따옴표 출력

\": 큰 따옴표 출력

\\: \ 출력

자료형 Data Type

문자열의 여러 기능

문자열 덧셈, 곱셈

특수한 문자 기호(Escape Code)

문자열 슬라이싱(Slicing)

슬라이싱이란? 문자열의 원하는 부위를 **잘라내는** 것!

파이썬에서는 범위를 지정하여 원하는 범위를 편하게 잘라낼 수 있다.

`a = "Hi, My name is TaeHyung"`

`a[4:6]` # "My" # index 기준 4이상 6미만의 범위를 잘라냄

`a[4:]` # "My name is TaeHyung" # 상한값,

`a[:3]` # "Hi," # 혹은 하한 값을 설정하지 않으면 자동으로 반대쪽 끝으로 설정됨

`a[:-1]` # "Hi, My name is TaeHyun" # index 에는 음수도 적용된다!

자료형 Data Type

문자열의 내장 함수

내장 함수란 파이썬에 기본적으로 제공되는 기능.

게임에서 **캐릭터**를 하나 생성하고 **직업**을 고르면 내가 고른 직업의 **기본 스킬**이 있다.
그런 기본 스킬은 **별도의 설치 없이** 해당 직업에 대해 사용할 수 있다.

이것이 파이썬의 내장 함수!

우리는 문자열이라는 직업을 고르고 그 문자열 직업의 기본 스킬을 사용해 볼거다.

자료형 Data Type

문자열의 내장 함수

string.lower() # string의 모든 문자를 **소문자**로 변경

string.upper() # string의 모든 문자를 **대문자**로 변경

a = string.count('A') # string에서 'A'의 **개수**를 세어서 a에 저장

isA = string.startswith('A') # string이 A로 시작하는지 **참/거짓 여부**를 저장

str_split = string.split() # 공백으로 문자열을 **잘라서**, 각 조각들을 리스트에 저장

str_split2 = string.split(',') # 쉼표로 문자열을 **잘라서**, 각 조각들을 리스트에 저장

n = len(string) # string의 **길이**를 n에 저장

자료형 Data Type

Boolean 형

참과 거짓을 저장하는 멋진 자료형

`a = True(== 1)`

`b = False (== 0)`

이런 느낌으로..

자료구조Data Structure

자료구조란?

자료를 저장하는 방식 != 자료형

자료형은 자료 하나하나의 종류였다면, 자료구조란 그런 자료들을 어떻게 모아서 저장할 건지에 대한 얘기:)



자료구조 Data Structure

내장된 자료구조

자료구조는 사실 알고리즘을 배우기 전에 주로 배운다.
(스택, 큐, 트리, .. 직접 구현해야..)

하지만 파이썬은 기본적으로 **내장**하고 있는 자료구조가 몇 개 있다!
하나하나 각 기능들과 함께 익혀보자.

- 1) 튜플
- 2) 리스트
- 3) 딕셔너리
- 4) 셋

자료구조 Data Structure

튜플 (Tuple)

튜플이란 여러가지 값들을 가장 간단한 형태로 저장하는 자료구조
`x = (3, 5, 'hi')` # 이런식으로 저장할 수 있다!

튜플의 성질은 한번 저장한 **값을 바꿀 수 없다**는 것.
그래서 튜플은 변하면 안되는 값들을 저장할 때 사용한다.

자료구조 Data Structure

튜플 (Tuple) - Example

```
tup_1 = (2, 1, 7, 8, "hi hello")
```

```
tup_2 = (1, 3, 5, (2, 4, 6))
```

```
print(tup_1[2]) # 7
```

```
print(tup_1[4][4:6]) # el
```

```
print(tup_1.count(2)) # 1
```

```
print(tup_1 + tup_2) # (2, 1, 7, 8, "hi hello", 1, 3, 5, (2, 4, 6))
```

```
print(tup_2 * 2) # (1, 3, 5, (2, 4, 6), 1, 3, 5, (2, 4, 6))
```

```
print(sorted(tup_1)) # (1, 2, 7, 8, "hi hello")
```

```
print(tup_2.index(5)) # 2
```

```
print(tup_1 + 6) # (2, 1, 7, 8, "hi hello", 6)
```

자료구조 Data Structure

리스트 (List)

리스트도 여러가지 값들을 **목록 형태**로 저장하는 자료구조
`x = [3, 5, 12, 'hihi']` # 이런식으로 저장할 수 있다.

이 친구는 대신 값의 수정이 가능하다!

가장 많이 쓰는, 가장 **일반적**인 자료구조이니 꼭 익혀두자.
그런 만큼 수 많은 내장 함수가 존재하니 이들의 사용법을 익혀보자.

자료구조 Data Structure

리스트 (List) - Example1

```
list_1 = [1, 3, 5, 7, 9]
```

```
list_2 = ['apple', 'pear', 'pineapple', 'banana']
```

```
print(list_1[3]) # 7
```

```
print(list_1[1:3]) # [3, 5]
```

```
list_1[4] = 99 # [1, 3, 5, 7, 99]
```

```
list_2.insert(2, 'new fruit') # ['apple', 'pear', 'new fruit', 'pineapple', 'banana']
```

```
list_2.remove('new fruit') # ['apple', 'pear', 'pineapple', 'banana']
```

```
list_1.append(12) # [1, 3, 5, 7, 99, 12]
```

```
list_1.sort() # [1, 3, 5, 7, 12, 99]
```

```
list_1.reverse() # [99, 12, 7, 5, 3, 1]
```

```
list_1.pop() # [99, 12, 7, 5, 3]
```

자료구조 Data Structure

리스트 (List) - Example2

```
list_1 = [1, 3, 5, 7, 9]
```

```
list_2 = ['apple', 'pear', 'pineapple', 'banana']
```

```
len(list_1) # 5
```

```
max(list_1) # 9 , min(list_2) # 'apple'
```

```
sum(list_1) # 25
```


자료구조 Data Structure

딕셔너리 (Dictionary)

앞서 배운 튜플, 리스트가 원하는 위치의 값을 가져오고 싶을 때 index 를 사용하였다.
(a[3], a[0], ..)

딕셔너리는 우리가 어렸을 때 사용했던 사전을 생각하면 이해가 쉽다.
‘사’ 를 찾아서 ‘삼’ 이라는 단어를 찾듯,
특정 ‘키(Key)’ 값으로 내가 원하는 ‘값(Value)’ 를 찾을 수 있다.

딕셔너리에서 기억해야 할 것은 Key-Value 이다! 꼭 기억하자.

x = {'태형': 21, '주형': 20}

자료구조 Data Structure

딕셔너리 (Dictionary) - Example

```
dict_1 = {'Amy': 70, 'Bob': 89, 'Carl': 55, 'David': 90}
```

```
dict_1['Amy'] # 70
```

```
dict_1['Emily'] = 100 # {'Amy': 70, 'Bob': 89, 'Carl': 55, 'David': 90, 'Emily': 100}
```

```
dict_1['Carl'] = 44 # {'Amy': 70, 'Bob': 89, 'Carl': 44, 'David': 90, 'Emily': 100}
```

```
dict_1.keys() # ['Amy', 'Bob', 'Carl', 'David', 'Emily']
```

```
dict_1.values() # [70, 89, 44, 90, 100]
```

```
n = dict_1.pop('Emily') # n = 100, dict_1 = {'Amy': 70, 'Bob': 89, 'Carl': 44, 'David': 90}
```

```
del(dict_1['Bob']) # {'Amy': 70, 'Carl': 44, 'David': 90}
```

자료구조 Data Structure

셋 (Set)

셋을 한국어로 하면 집합이다.

중, 고등학교 때 배웠던 집합, 집합의 성질이 기억나는가?

집합은 중복 원소가 없다!

즉, 셋이라는 자료구조는 **중복인 값이 존재하지 않는** 자료구조이다.

그냥 우리가 수학 시간 배웠던 집합의 성질을 모두 포함하는 자료구조이다 라고 이해해보자:) (교집합, 합집합, 차집합..)

(혹시 집합을 배우지 않았다 해도 걱정말자 셋은 잘 안쓰인다ㅠㅠ)

자료구조 Data Structure

셋 (Set) - Example

```
set_1 = {1, 2, 3, 4, 5, 6}
```

```
set_2 = {2, 4, 6, 8, 10, 12}
```

```
print(set_1.union(set_2)) # 합집합, {1, 2, 3, 4, 5, 6, 8, 10, 12}
```

```
print(set_1.intersection(set_2)) # 교집합, {2, 4, 6}
```

```
print(set_1.difference(set_2)) # 차집합, {1, 3, 5}
```

```
set_1.add(7) # {1, 2, 3, 4, 5, 6, 7}
```

```
set_2.remove(12) # {2, 4, 6, 8, 10}
```

자료구조 Data Structure

Special: Set으로 List 중복 제거하기

```
list_1 = [1, 2, 3, 1, 2, 5, 7]
```

```
list_1 = list(set(list_1)) # [1, 2, 3, 5, 7]
```

자료구조 Data Structure

HW2

비트코딩 교재 Prob49 ~ Prob 83, Prob 95 ~ Prob 131

각 문제를 풀어서 '1강_HW2_이름' 폴더에 각 문제 풀이 파일을 저장
Slack HW Channel 에 Upload!

입력

Input

입력 Input

입력

입력은 간단하다!

사용자 키보드로부터 문자열을 입력받는 것(엔터를 기준으로)

`n = input("입력해보세요: ")` => “”: 안내를 위한 문자열

n에 저장되는 값은? **무조건 문자열!**

- Tip: 어떤 변수의 자료형을 알고 싶다면?

`print(type(n))`

입력 Input

입력

n = input("정수를 입력해보세요: ")
n에 저장되는 값은? 무조건 문자열!

엇 근데 난 정수를 입력받고 싶은데??

=> 앞에서 배웠던 강제 형변환 사용!

n = int(input("정수를 입력해보세요: "))

입력Input

HW3

비트코딩 교재 Prob84 ~ Prob 94

각 문제를 풀어서 '1강_HW3_이름' 폴더에 각 문제 풀이 파일을 저장
Slack HW Channel 에 Upload!

감사합니다:)

Thank you