Python Language

최종 목표물

❖ 각 관계식 찾고, 프로그래밍 하기(머신러닝 개념 이해 도움)

X	Y
0	0
1	2
2	4
3	6

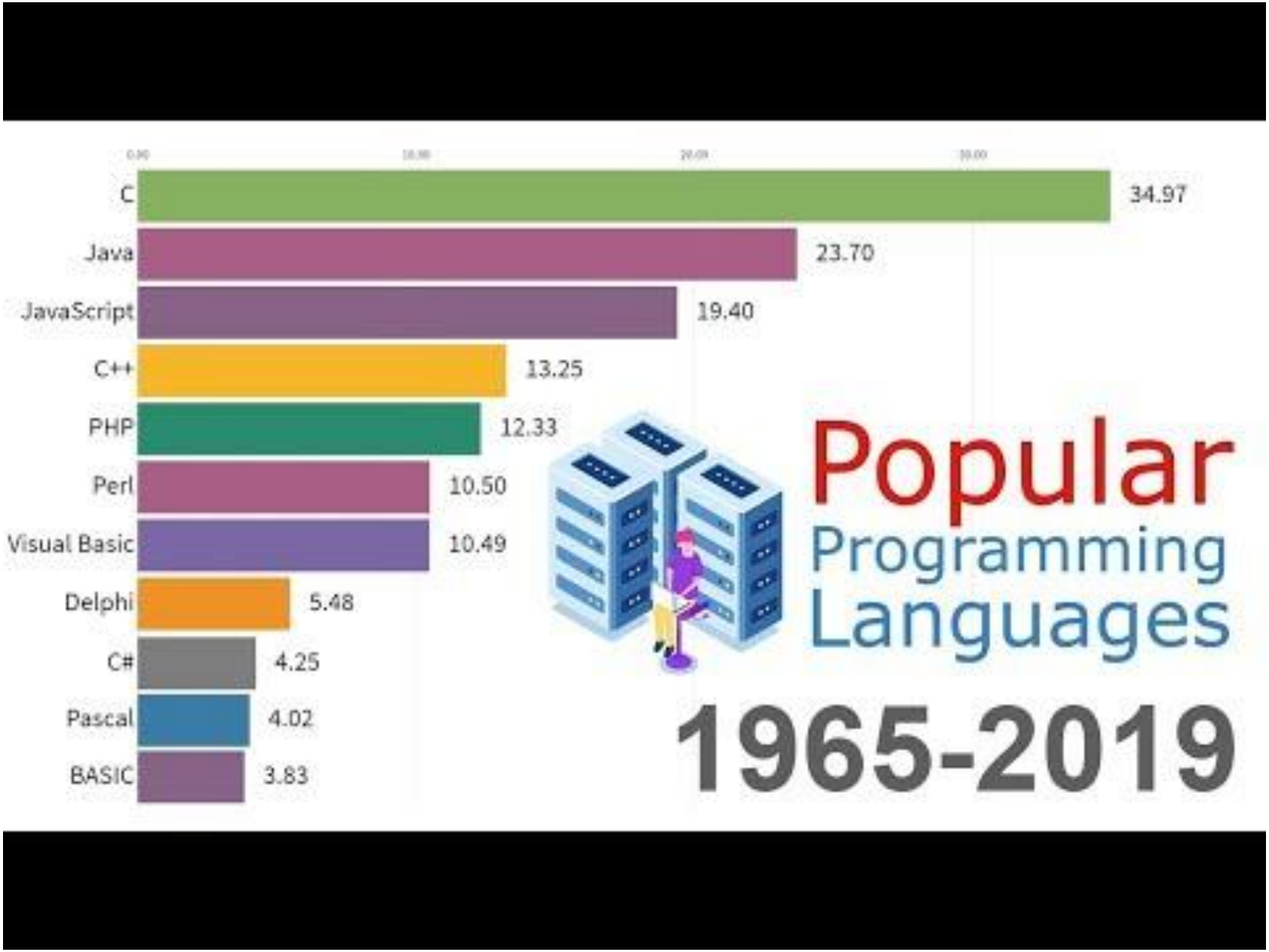
X	Y
0	2
1	2.5
2	3
3	3.5

X1	X2	Y
0	2	2
1	3	4
2	4	6
3	5	8

X1	X2	Y
0	2	6
1	3	9
2	4	11.5
3	5	14.5

Why Python

- [Most Popular Programming Languages 1965 - 2019]



Setup Visual Studio Code

- Use Jupyter With Colab : googling with ‘google colab’
  - https://colab.research.google.com/
- Install Anaconda For Python
  - https://www.anaconda.com/
  - add Path in Windows OS → path : c:\anaconda\bin;
- Install Visual Studio Code For Tool.
  - https://code.visualstudio.com/
  - Plug in useful package

ANACONDA®

Visual Studio Code

Setup Anaconda - Linux

```
# Go to home directory
cd ~
# You can change what anaconda version you want at
# https://repo.continuum.io/archive/
$ wget https://repo.continuum.io/archive/Anaconda3-4.2.0-Linux-x86_64.sh
$ bash Anaconda3-4.2.0-Linux-x86_64.sh -b -p ~/anaconda
$ rm Anaconda3-4.2.0-Linux-x86_64.sh
$ echo 'export PATH="~/anaconda/bin:$PATH"' >> ~/.bashrc
# Reload default profile
$ source ~/.bashrc
$ conda update conda
$ jupyter notebook --ip=0.0.0.0 --port=8080
```

Hello World with Python3

- **print()**

```
>>> print("Hello, World!")
>>> str = "Hello, World!"
>>> print(str)
```

- 주석 달기 : # or “””...””” or ‘’’...’’’

```
‘’’ Unlike other C based languages ’’’
    # Python IDLE allows you to put your code
```

- **Tab(들여쓰기)과 : 중요**.
- 식별자(변수) 형 미리 선언 필요 없음 : shell 과 유사 → 뒤에 설명

```
ex) myage = input('What is your age? ')
    age = int(myage)
    print("Your age is "+str(age))
```

→ 변수와 상수 구별.

Standard Data Types(1)

- Numbers : int, long(L), float, complex(3.14j,3e+26j), etc
- String

```
str = 'Hello World!'        # 재사용 위해 변수 사용
str          # Prints complete string
str[0]       # Prints first character of the string
str[2:5]     # Prints characters starting from 3rd to 5th
str[2:]      # Prints string starting from 3rd character
str * 2      # Prints string two times
str + "TEST" # Prints concatenated string
```

- **List**

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']
list          # Prints complete list
list[0]       # Prints first element of the list
list[1:3]     # Prints elements starting from 2nd till 3rd
list[2:]      # Prints elements starting from 3rd element
tinylist * 2  # Prints list two times
list + tinylist # Prints concatenated lists
```

Standard Data Types(2)

- **Dictionary** : Key, Value 묶음, kind of hash table type

```
dict = {}
dict['one'] = "This is one"
dict[2]     = "This is two"
tinydict = {'name': 'john','code':6734, 'dept': 'sales'}
dict['one']       # Prints value for 'one' key
dict[2]           # Prints value for 2 key
tinydict          # Prints complete dictionary
tinydict.keys()   # Prints all the keys
tinydict.values() # Prints all the values
```

- Tuple : **read-only**

```
tuple = print( 'abcd', 786 , 2.23, 'john', 70.2  )
tinytuple = print(123, 'john')
tuple           # Prints complete list
tuple[0]        # Prints first element of the list
tuple[1:3]      # Prints elements starting from 2nd till 3rd
tuple[2:]       # Prints elements starting from 3rd element
tinytuple * 2   # Prints list two times
tuple + tinytuple # Prints concatenated lists
```

String

- Special Operators : +, *, [:], in, not in, %
  var1 = 'Hello World!'; ("Updated String :- ", var1[:6] + 'Python')
- Formatting Operator : %c, %s, %i, %d, %u, %o, %x, %f
  ("My name is %s and weight is %d kg!" % ('Zara', 21))
- Methods
  - capitalize(), count(str, beg = 0,end = len(string)),
  - decode(encoding = 'UTF-8',errors = 'strict'), encode(encoding = 'UTF-8',errors = 'strict')
    Str = "this is string example....wow!!!"; Str.encode('base64','strict');
  - find(str, beg = 0 end = len(string)), index(str, beg = 0, end = len(string))
  - isalnum(), isalpha(), isdigit(), islower(), isnumeric(), isspace(), isupper()
    str = "This Is String Example...Wow!!!"; (str.istitle())
  - join(seq), len(string), ljust(width[, fillchar]),lower(), lstrip()
    str = "this is string example....wow!!!"; str.ljust(50, '*')
  - max(str), min(str), replace(old, new [, max]), split(str="", num=string.count(str)),upper()
- Codec : 다른 언어간 의사 전달 위한 약속
  - https://docs.python.org/3/library/codecs.html#standard-encodings

Data Type - List

- 여러 데이터 형태 집합 가능 : **help(list) or dir(list) 참조**
- len(list), max(list), min(list), list(seq)
  cast = ["Cleese", "Palin", 34.52, 'Idle', "Cleese"]
  print(cast[2], cast[-2], cast[1:])
- append(obj) : Appends object obj to list
  cast.append('Gilliam')
- count(obj) : Returns count of how many times obj occurs in list
  cast.count('Cleese')
- extend(seq) : Appends the contents of seq to list
  cast.extend(['Idle', 'Gilliam'])
- pop(obj = list[-1]) : Removes and returns last object or obj from list
  cast.pop()
- insert(index, obj) : Inserts object obj into list at offset index
  cast.insert(0, 'Cleese')
- **remove(obj)** : Removes object obj from list
  cast.remove('Cleese')
- reverse() : Reverses objects of list in place
  cast.reverse()
- **isinstance()** : 구분 데이터형 기술 필요
  ex) isinstance(movies, list)

Try - List 자료형 구현과 이해

```
>>> movies = ["The Holy Grail", "The Life of Brian", "The Meaning of Life"]
>>> print(movies)
['The Holy Grail', 'The Life of Brian', 'The Meaning of Life']
>>> print(movies[1])
The Life of Brian
>>> print(len(movies))
3
>>> print(len(movies[1])+" - "+movies[2])
                        → Try : print(str(len(movies[1]))+" - "+movies[2])
                        → Try : print(len(movies[1])," - ",movies[2])
17
>>> del movies[2]
```

❖ 해보기

- ➢ 3종류 동물 이름과 분류 List 작성
- ➢ 두번째에 새 동물 이름과 분류 추가.
- ➢ 세번째 동물 정보 출력
- ➢ 세번째 동물 정보 삭제

Try - 복합 자료형 List 구현과 이해

❖ 같이 하기

```
>>> movies = ["The Holy Grail", "The Life of Brian", "The Meaning of Life"]
>>> print(movies)
['The Holy Grail', 'The Life of Brian', 'The Meaning of Life']
>>> movies.insert(1, 1975)
>>> movies.insert(3, 1979)
>>> movies.insert(5, 1983)
>>> print(movies)
['The Holy Grail', 1975, 'The Life of Brian', 1979, 'The Meaning of Life', 1983]
>>> movies.remove(1983)
>>> print(movies)
['The Holy Grail', 1975, 'The Life of Brian', 1979, 'The Meaning of Life']
```

❖ 해보기 : 위 예제 이용 아래와 같이 결과 출력

```
>>> movies = ['The Holy Grail', 1975,
                  'The Life of Brian', 91, ['Graham Cahpman',
                                ['Michael Palin', 'John Cleese', 'Terry Gilliam', 'Eric Idle']]]
>>> print(movies[4][1][2])
Terry Gilliam
>>> isinstance(movies, list)
True
```

Data Type - Tuples

- 리스트와 유사한 불변 데이터(Sequence of immutable like lists)
- len(tuple), max(tuple), min(tuple), tuple(seq)

```
cast =("Cleese", "Palin", 34.52, 'Idle', "Cleese")
print(cast[2], cast[-2], cast[1:])
len(cast), max(cast), min(cast)
```

- cmp(tuple1, tuple2) : Compares elements of both tuples
  - ➢ 함수 맛보기

```
def cmp(a, b):                                  → ':' 와 Tab 주의
  return (a > b) - (a < b)

tuple1, tuple2 = (123, 'xyz'), (456, 'abc')
cmp(tuple1, tuple2)
cmp(tuple2, tuple1)
tuple3 = tuple2 + (786,);
cmp(tuple2, tuple3)
```

- 알아가기
  - ➢ set(반복 가능한 객체) : 집합 형식, 중복 데이터 제거

ex) distances = set(cast)

Try - Tuple

❖ 같이 하기

```
tup1 = ()
tup2 = (50,)                                    → 선언 권장. 함수 단계 설명
tup3 = ('physics', 'chemistry', 1997, 2000)
tup4 = (1, 2, 3, 4, 5 )
tup5 = "a", "b", "c", "d"
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5, 6, 7 )
print("tup1[0]: ", tup1[0])
print("tup2[1:5]: ", tup2[1:5])
tup1 = (12, 34.56)
tup2 = ('abc', 'xyz')
tup3 = tup1 + tup2
print(tup3)
 tup = ('physics', 'chemistry', 1997, 2000);
 print(tup)
 del tup;
 print("After deleting tup : ",tup)             → Error Traceback
```

Data Type - Dictionary

- 여러 데이터 형태 집합 가능, key, value 묶음.
- len(dict), str(dict), type(variable)
  dict1 = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
  print("dict1['Name']: ", dict1['Name'])
- dict.copy() : Returns a shallow copy of dictionary dict
  dict2 = dict1.copy()
- dict.get(key, default=None) : returns value or default if key not in dictionary
  dict1.get('Age')
- dict.keys() : Returns list of dictionary dict's keys
  dict.keys()
- dict.setdefault(key, default = None) : Similar to get(), but will set dict[key] = default if key is not already in dict
- dict.update(dict2): Adds dictionary dict2's key-values pairs to dict
  dict3 = {'Sex': 'female', 'Age': 9 }
  dict1.update(dict3)
  dict1
- dict.values() : Returns list of dictionary dict's values
  dict.values()

Try - Dictionary

❖ 같이 하기

➢ **결과 예상 후 실습**

```
x = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
x.update(a=100, b=200);
del x['a'];
x.get('a');
x.keys();
x.clear()

x = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
y = x.copy();
print(x is y)
print(x == y)
print('a' not in x);
print('d' in x);
for key, value in x.items():
    print(key, value)
```

```
x.pop('a')
x.setdefault('f', 100)      → Try x['f'] = 200
x.items()
x.values()
→ del 와 차이 있음.

y['a'] = 99
# False
# True
# False
# True
→ for 맛보기
```

Try - 복합 Dictionary

❖ 같이 하기

```
terrestrial_planet = {                    → 데이터 형 관련 없음.
    'Mercury': {
        'mean_radius': 2439.7,
        'orbital_period': 87.969
    },
    'Venus': {
        'mean_radius': 6051.8,
        'orbital_period': 224.70069
    },
    'Earth': {
        'mean_radius': 6371.0,
        'mass': 5.97219E+24
    },
    'Mars': {
        'orbital_period': 686.9600
    }
}
print(terrestrial_planet)
print(terrestrial_planet['Venus']['mean_radius'])
```

Operator - Feature

- Arithmetic (산술) : *, /, %, //
- Comparison (비교) : <=, <, >, >=, <>, ==, !=
- Assignment : =, %=, /=, //=, -=, +=, *=, **=

```
c *= a        # c = c * a
c /= a        # c = c / a
c %= a        # c = c % a
c **= a       # c = c ** a
```

- Bitwise : >>, <<, &, |, ~

```
a = 60; b = 13;          # 60 = 0011 1100; 13 = 0000 1101
a&b                      # 12 = 0000 1100
a|b                      # 61 = 0011 1101
a^b                      # 49 = 0011 0001
a << 2;       # 240 = 1111 0000
```

- Identity : is, is not

```
if ( a is b ):
if ( a is not b ):
```

- Membership : in, not in

```
if ( b not in list ):
```

- Logical : not, or, and

```
if ( (b not in list) and not( a is not b ) ):
```

Statement - if else

if condition :
    indented Statement Block
elif condition2 **:** → **else if** condition2 가능
    indented Statement Block
else :
    indented Statement Block

ex) temperature = **int**(**input**('What is the temperature? '))
    if temperature > 70:
        print('Wear shorts.')
    else:
        print('Wear long pants.')
    print('Get some exercise outside.')

Try - if else 이해

❖ 실행결과

What is your score? 87
your score is B

❖ 같이 하기

```
score = int(input('What is your score? '))
if score >= 90:
    letter = 'A'
elif score >= 80:
    letter = 'B'
elif score >= 70:
    letter = 'C'
elif score >= 60:
    letter = 'D'
else:
    letter = 'F'
print('your score is ' + letter)
```

❖ 맛보기

```
while 1:
    위 문장 넣기
```

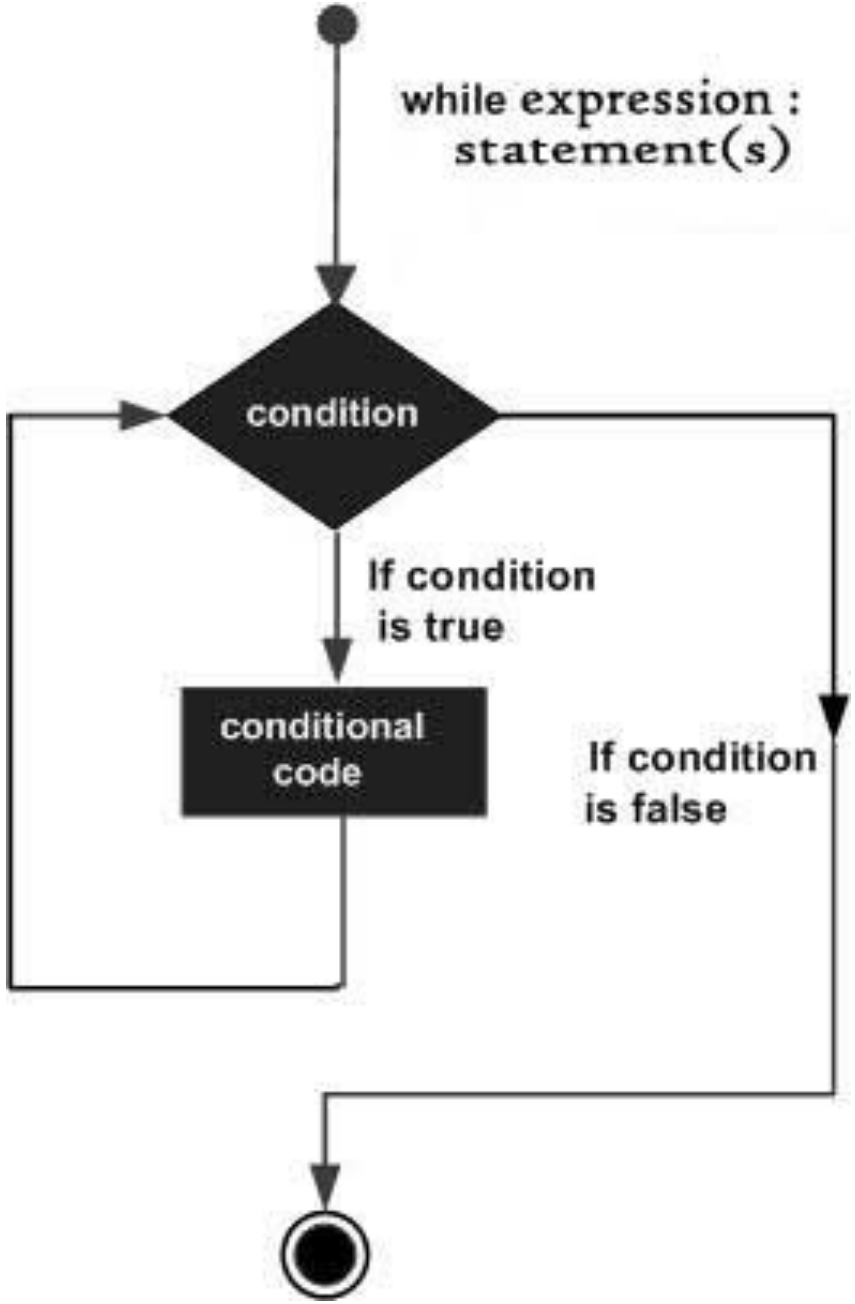
❖ 생각해 보기 : 반복문 빠져 나오기

Statement - while

```
while expression:
    statement(s)

ex) count = 0
    while (count < 9):
        'The count is:', count)
            count = count + 1
        print("Good bye!")

ex) count = 0
    while count < 5:
        (count, " is  less than 5")
        count = count + 1
    else:
        (count, " is not less than 5")
```



Try - while 구현과 이해

❖ 실행결과

```
The Holy Grail
1975
The Life of Brian
1979
The Meaning of Life
1983
```

❖ 같이 하기

```
movies = ["The Holy Grail", "The Life of Brian", "The Meaning of Life"]
for each_flick in movies:
    print(each_flick)

count = 0
while count < len(movies):
    print(movies[count])
    count = count+1
```

Statement - for

for **iterating_var** in **sequence**:
    statements(**iterating_var**)

ex) for letter in 'Python':
        print('Current Letter :', letter)

ex) fruits = ['banana', 'apple', 'mango']
    for fruit in fruits:
        print('Current fruit :' + fruit)

    for **index** in **range**(**len**(fruits)):
        print('Current fruit :', **fruits[index]**)

    movies = ["The Holy Grail", "The Life of Brian", "The Meaning of Life"]
    for **each_flick** in movies:
        print(each_flick)

Try - For 구현과 이해

❖ 실행결과

10 equals 2 * 5

11 is a prime number

...

18 equals 2 * 9

19 is a prime number

❖ 같이 하기

```
for num in range(10,20):  #to iterate between 10 to 20
    for i in range(2,num): #to iterate on the factors of the number
        if num%i == 0:
            j=num/i
            print('%d equals %d * %d' % (num,i,j))
            break
    else:                  # else part of the loop
        print(num, 'is a prime number')
```

→ Try : 콤마 적용

Try - For 구현과 이해(nested List)

❖ 실행결과

```
The Holy Grail
1975
The Life of Brian
91
Graham Cahpman
['Michael Palin', 'John Cleese', 'Terry Gilliam', 'Eric Idle']
```

❖ 같이 하기

```
movies = ['The Holy Grail', 1975,
      'The Life of Brian', 91,
      ['Graham Cahpman',['Michael Palin', 'John Cleese', 'Terry Gilliam', 'Eric Idle']]]
for each_item in movies:
      print(each_item)
```

❖ 해 보기

➢ nested List 아래 결과 추가 출력

```
Michael Palin
John Cleese
Terry Gilliam
Eric Idle.py)
```

Useful Method

❖ enumerate() : iterate로 사용 가능.

```
some_list = ['foo', 'bar', 'baz']
for index, value in enumerate(some_list):
    print('i: {}, v: {}'.format(index,value))
```

❖ zip() : 동일한 자리끼리 묶어줌

```
seq1 = ['foo', 'bar', 'baz']
seq2 = ['one', 'two', 'three']
seq3 = ['apple', 'raspberry', 'banana']
for a, b, c in zip(seq1, seq2, seq3):
    print('a: {}, b: {}, c: {}'.format(a, b, c, ))
    print('b: {1}, c: {2}, a: {0}'.format(a, b, c, ))
```

→ Try seq3 = ['apple', 'raspberry',]

Function(1)

- statement

  def **function_name(parameters, ...)**:

  function_suite

  return [expression]

- And Then Call below

  result = **function_name(parameters, ...)**

ex) def **print_info(name, age = 35)**:

```
    print("Name: ", name)
    print("Age ", age)
    return;
print_info( age=50, name="miki" )
print_info( name="miki" )
```

ex) def **sum(arg1, arg2)**:

```
    result = arg1 + arg2
    print("Inside the function : ", result)
    return result
total = sum( 10, 20 )
print("Outside the function : ", total )
```

Try - Function 구현과 이해

```
ex) def changeme( fmylist ):
        fmylist = [1,2,3,4];                    → Try : fmylist.append([1,2,3,4])
        print("Values inside the function: ", fmylist)
        return
    mylist = [10,20,30];
    changeme( mylist );
    print("Values outside the function: ", mylist)

ex) def changeme( fmylist ):
        fmylist = [1,2,3,4];
        print("Values inside the function: ", fmylist)
        return fmylist
    mylist = [10,20,30];
    mylist = changeme( mylist )
    print("Values outside the function: ", mylist)

ex) def printinfo( fname, fage = 35 ):
        return fname, fage;
    name, age = printinfo( fage=50, fname="miki" )
    print(name + " ", " + str(age))             → Try : print(name, age)
```

Try - Function 구현과 이해

❖ 실행결과

The Holy Grail
The Life of Brian
The Meaning of Life

❖ 해보기

➢ movies = ["The Holy Grail", "The Life of Brian", "The Meaning of Life"]

➢ 출력 function 작성

Try - Sum function 구현과 이해

❖ 실행결과

input number : 20, 10
Inside the function : 30
Outside the function : 30

❖ 해보기

➢ 입력 : 두 숫자

■ **first, second** = input("Input two Number ? ")**.split(',')**

➢ 덧셈 Function 작성.

❖ 더 해 보기

➢ 사칙 연산 적용

Function(2)

❖ Variable-length Arguments

def function_name(**[formal_args,] *var_args_tuple**):
  function_suite
return [expression]

ex) def **print_return_tuple(arg1, *vartuple)**:
    print("Inside arg1 is : ",arg1)
    for var in vartuple:
        print("Inside vartuple is : ",var)
    return vartuple

print_return_tuple(10)
out_tuple = **print_return_tuple**(70, 60, 50)
print(out_tuple)

Function(3)

❖ 선택 인자 : 기본값 추가(ex. level=0 or **var_args_dict)

```
def pwe_print_lol(the_list, level=1):
    for each_item in the_list:
        if isinstance(each_item, list):
            pwe_print_lol(each_item, level+1)
        else:
            for tab_stop in range(level):
                print("\t", end='')
            print(each_item)
>>> movies = ['The Holy Grail', 1975,
                'The Life of Brian', 91, ['Graham Cahpman',
                        ['Michael Palin', 'John Cleese', 'Terry Gilliam', 'Eric Idle']]]
>>> pwe_print_lol(movies)                          → Error
>>> pwe_print_lol(movies, level=0)
```

Try - 선택 인자

❖ 알아가기 - 인자 *와 ** 차이

```
>>> def functionA(*var_args01, **var_args02):
>>>     print(var_args01, type(var_args01))
>>>     print(var_args02, type(var_args02))
>>> functionA(1, 2, 3, 4, 5, 6, a=2, b=3, c=5)
(1, 2, 3, 4, 5, 6)
{'a': 2, 'c': 5, 'b': 3}
>>> args_list=[1, 2, 3, 4]
>>> args_dict={'a': 10, 'b':20}
>>> functionA(*args_list, **args_dict)                    → Not is (args_list, args_dict)
(1, 2, 3, 4)
{'a': 10, 'b': 20}
```

❖ 해보기 - 앞 예제 참조

- ➢ indent 선택 가능 함수 만들기
  def pwe_print_lol(the_list, indent=False, level=0):
- ➢ dictionary type 인자로 함수 구성
  def pwe_print_lol(**var_args):

Function - Lambda

❖ 간편 함수 선언

lambda [arg1 [,arg2,.....argn]] **:** expression

ex1) **sum** = **lambda** arg1, arg2 : arg1 + arg2;
print("Value of total : ", sum(10, 20))

ex2) nums = [1,2,3]

```
def is_greater_than_one(x):
    return x > 1
>>> more_than_nums = filter(is_greater_than_one, nums)
>>> print(list(more_than_nums))
[2,3]
>>> more_than_nums = filter(lambda x : x > 1, nums)
>>> print(list(more_than_nums))
```

❖ 해 보기

➢ ex1)을 함수 방식으로 수정

Try - 사칙계산기 Function 구현과 이해

❖ 실행결과

Enter first operands: 30.5
Enter an operator (+, -, *) -
Enter second operands: 50.6
30.5 - 50.6 = -20.1
Have a good time !

→ Try : 나눗셈 시도

❖ 해보기

➢ 숫자, 연산기호, 숫자 순차 입력

➢ ‘q’ 입력 하면 "Have a good time !" 출력 후 종료.

❖ 궁금해 보기

➢ 이상 입력 시 에러 처리

➢ 아래와 같은 방식 처리

■ (3*2) + (5*3) =

Python(First in 1991, https://docs.python.org/3/library/index.html)

- feature
  - Created by Guido van Rossum
  - interpreted language
- Be available for Many operating systems
- install python

  ~$ apt-get update; apt-get upgrade;

  ~$ python3 -V
- 각종 IDE :
  - Python Shell

    ~$ apt-get install python3-tk idle3

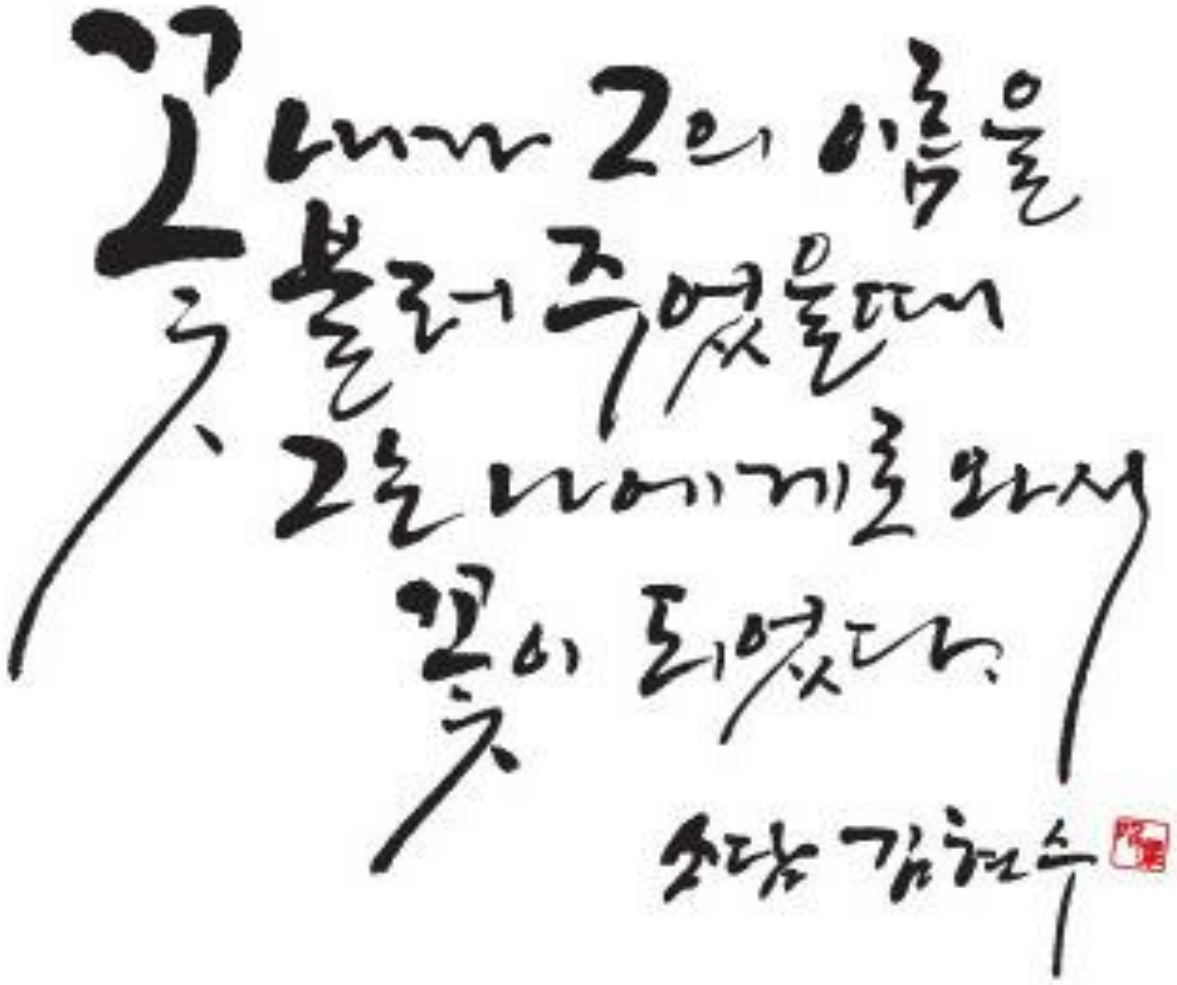
    ~$ python3 idlelib
  - WingIDE
    - Be designed specifically for the Python
  - Eclipse
    - plugin pyDev
  - **Visual Code**
  - **Jupyter notebook**
- Run python

  ~$ python3

  >>> _

Memory

- **불러주었을 때 인스턴스**가 되었다.
- 인스턴스(=프로시저)는 메모리 올리는 것.
- 식별자(변수) 형 미리 선언 필요 없음 : shell 과 유사
- **변수가 중요 : 동사와 명사를 알면 프로그래밍이 보인다.**

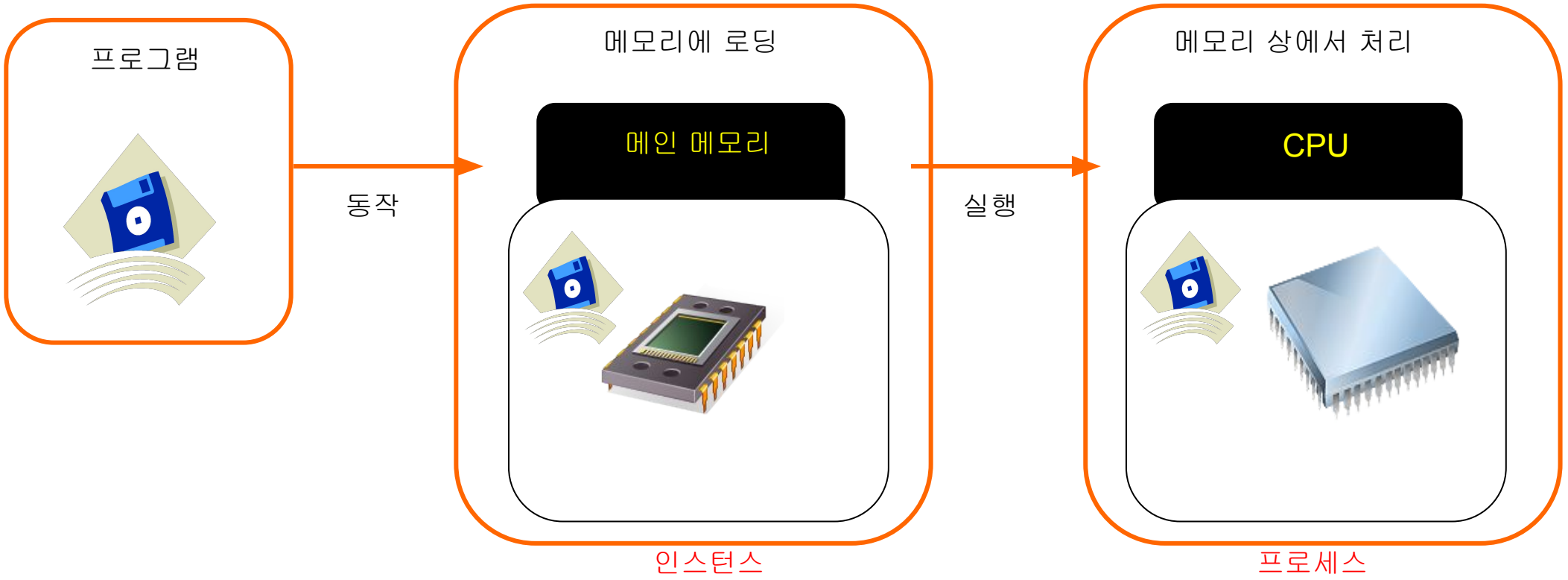


영혼은 존재하는가 ?

- 우린 잘 죽기 위해 태어났다.
- 죽음 ←→ 소프트웨어.

Memory와 프로그램

- CPU Register 한계 극복
- 스택영역 : 호출된 함수의 종료후 원래 함수의 실행이던 위치로 돌아오기 위해 복귀 주소를 저장
- 데이터 영역 : 전역변수등 프로그램이 사용하는 각종 데이터가 저장되는 공간
- 코드 영역 : 프로그래머가 작성한 코드가 기계어 명령형태로 저장되는 영역



Directory & File

❖ 폴더 다루기

➢ os.mkdirprint("newdir") : Directory 생성

```
import os
os.mkdir("./dir01")
```

➢ os.getcwd() : 현재 위치

➢ os.chdir(directory) : 디렉토리 이동.

```
os.chdir(os.getcwd()+'/dir01')
print(os.getcwd())
```

➢ os.rmdir('dirname') : Directory 삭제

```
os.mkdir("./dir01")
os.rmdir("./dir02")
```

❖ 파일

➢ os.path.exists(file_name) : file 또는 Directory 유무

```
print(os.path.exists('./test/hellow.py'))
```

➢ os.remove(file_name) : 파일 삭제

```
os.remove(‘hellow.py’)
```

File I/O(1)

- file object = **open**(file_name [, **access_mode**][, buffering]) : 파일 열기
  - **access_mode :** 파일 접근 모드
    - r : 읽기 모드, r+ : 쓰기도 가능, 없을 시 Error
    - w : 쓰기 모드, w+ : 읽기도 가능, 새로 파일 생성
    - a : 추가 모드, a+ : 읽기도 가능, 파일 끝 추가, 없으면 생성.
    - 입출력 모드 : t - text(ASCII) file, b - binary file
- fileObject.**close**(); : 반드시 파일 닫기

ex) make file ./foo1.txt

```
fo = open("foo1.txt", "wb")
print("Name of the file: ", fo.name)
print("Closed or not : ", fo.closed)
print("Opening mode : ", fo.mode)
fo.close()
```

File I/O(2)

❖ fileObject.**write**(string); : 파일 입력

ex) fo = open("foo2.txt", "wb")
fo.write(**b**"Python is a great language.\nYeah its great!!\n")
fo.write(**br**'C:\\nowhere'**)** → **r'expression**
fo.close()

❖ fileObject.**read**([count]); : 파일 출력

ex) fo = open("foo3.txt", "rb")
str = fo.read(10);
print("Read String is : ", str)
fo.close()

❖ fileObject.seek(offset[, from]) : from 서 offset 만큼 이동.(0-시작, 1-현재,2-끝)
data.seek(50,1) : 현재 위치서 50만큼 이동.

❖ fileObject.readline() : 줄단위 읽기

❖ fileObject.find(':') : 해당 문자열 찾기

❖ fileObject.strip() : 양끝 공백 삭제

❖ Str.split(sep=None, maxsplit=-1) : sep 문자열을 maxsplit 반복만큼 분리

Try - 파일 다루기 구현과 이해

❖ 실행결과

```
Man: Is this the right room for an argument?
Other Man: I've told you once.
ven't!
Other Man: Yes I have.
(pause)
Man: Yes it is!
```

❖ 같이 하기

```
def f_open(filename):
    fdata = open(filename)
    return fdata
data = f_open('sketch.txt')
print(data.readline(), end = '')
print(data.readline(), end = '')
data.seek(90)
for each_line in data:
    print(each_line, end = '')
data.close()
```

→ 파일 위치 확인

→ Try : 주석 처리

Try - 파일 다루기 구현과 이해

❖ 실행결과

```
...
Man said:  You most certainly did not!
Traceback (most recent call last):   ...
ValueError: too many values to unpack (expected 2)
```

❖ 같이 하기

```
def f_compared(str):
    result = str.find(':')
    return result
data = open('sketch.txt')
for each_line in data:
     if not f_compared(each_line) == -1:     → (pause) 제거
          (role, line_spoken) = each_line.split(':')
                                → Try : (role, line_spoken) = each_line.split(':', 1)
          print(role, end = '')
          print(' said: ', end = '')
          print(line_spoken, end = '')
data.close()
```

❖ 해 보기

➢ Function Wrapping each_line.**split**(':')

Handling Exceptions

❖ Statement

try:
    You do your operations here;
except Exception1:
    If there is Exception1, then execute this block.
else:
    If there is no exception then execute this block.
finally:
    This would always be executed.

ex) for each_line in data: → 에러 복구 정의 코드
    try:
        … (role, line_spoken) = each_line.split(':') ...
    except:
        pass → 무시하고 진행.
    finally :
        data.close()
except ValueError or IOError: → 종류 명기
except IOError as err: → 내용 출력
    print("File Error' + str(err))

Try - 다른 방식 처리(예외)

❖ 알아가기 : **print**(**objects*, *sep=' '*, *end='\n'*, *file=sys.stdout*, *flush=False*)

```
print(x, end=" ")  # Appends a space instead of a newline
print("fatal error", file=sys.stderr)
```

❖ 실행결과

```
...
Man said:  You most certainly did not!
...
```

❖ 같이 하기

```
data = open('sketch.txt')
for each_line in data:
    try:
        (role, line_spoken) = each_line.split(':')
        print(role, end='')
        print(' said: ', end='')
        print(line_spoken, end='')
    except:
        pass
data.close()
```

Try - 파일 복사와 예외(finally) 구현과 이해

```
try:
    data = open('sketch.txt');  man=[];     other=[];
    for each_line in data:
        try:
            (role, line_spoken) = each_line.split(':')
            if role == 'Man':
                man.append(line_spoken)
            elif role == "Other Man":
                other.append(line_spoken)
        except ValueError:
            pass
    data.close();         print(man);          print(other);
    try:
        man_out = open('man_data.txt', 'w')
        other_out = open('other_data.txt', 'w+')
        print(man, file=man_out)       print(other, file=other_out)
    except IOError:     print('The script file is missing.')
    finally:      man_out.close();        other_out.close()
except IOError:     print('The datafile is missing')
```

Try - 파일 다루기 위해 with 활용

```
try:
    man_out = open('man_data.txt', 'w')
    print(man, file=man_out)
except IOError as err:
    print('The script file is missing. '+ str(err))
finally:
    if ‘man_out’ in locals():
        data.close()
```

❖ with 적용

```
man = 'Its a good day'
try:
    with open('man_data.txt', 'w') as man_out:
        print(man, file=man_out)
except IOError as err:
    print('The script file is missing. '+ str(err))
```

Try - list sort()

❖ 실행결과

['2:58', ' 3:08', ' 2:23', ' 2:59', ' 2-12', ' 3-11', ' 2:34']
[' 2-12', ' 2:23', ' 2:34', ' 2:59', ' 3-11', ' 3:08', '2:58']
[' 2-12', ' 2:23', ' 2:34', ' 2:59', ' 3-11', ' 3:08', '2:58']

❖ 같이 하기

```
try:
    with open('james.txt','r') as james_file:
        james_data = james_file.readline()
        james = james_data.strip().split(',')
        print(james)
        james_sorted = sorted(james)
        print(james_sorted)
        james.sort()
        print(james)
except IOError as err:
    print('IO error : '+err)
```

Try - 시간 형태 다른 리스트 처리

```
def sanitize(time_string):
    if '-' in time_string:
        splitter = '-'
    elif ':' in time_string:
        splitter = ':'
    else:
        splitter = '.'
    (mins, secs) = time_string.split(splitter)
    return (mins.strip()+'.'+secs.strip())
try:
    james_item = []
    with open('james.txt','r') as james_file:
        james_data = james_file.readline()
        james = james_data.strip().split(',')
        for each_item in james:
            james_item.append(sanitize(each_item))
        james_sorted = sorted(james_item)
        print(james_sorted)
except IOError as err:
    print('IO error : '+err)
```

Class

- 코드와 데이터 통합 통한 복잡도 낮춤 → 유지보수 유리
- 객체 지향 개념(OOP-Object-Oriented Programming) : 목적별 상속, 맞춤 공정
  - ➢ 코드 = 메서드
  - ➢ 데이터 = 속성
  - ➢ 데이터 객체 = 인스턴스
- 클래스 선언 형식

```
class 클래스명:
        __init__(self, …):
        class_suite :
            # …
```

```
ex) class Athlete:
        def __init__(self, value='Jane'):
            self.thing = value;
        def getAthlete(self):
            return self.thing
```

- 클래스 통한 인스턴스 객체 생성

인스턴스명 = 클래스명()

a = Athlete() → Athlete.__init__(a)

b = Athlete('Holy') → Athlete.__init__(b,'Holy')

Try - Class 구성 구현과 이해

❖ 실행결과

Point01 Class 0 , 0
Point02 Class 3 , 5

❖ 같이 하기

```
class Point:
    def __init__( self, x=0, y=0):
        self.x = x
        self.y = y

pt1 = Point()
print('Point01 Class', pt1.x, ',', pt1.y)
pt2 = Point(3,5)
print('Point02 Class', pt2.x, ',', pt2.y)
```

Try - Class 정보 구현과 이해(ClassAthlete.py)

❖ 실행결과

```
>>> james
<__main__.Athlete object at 0x1023ab780>
>>> sarah
<__main__.Athlete object at 0x1055d99e8>
>>> james.name
'James Jones'
>>> sarah.times
['2:58', '2.58', '1.56']
```

❖ 같이 하기

```
class Athlete:
    def __init__(self, athlete_name, athlete_dob=None, athlete_times=[]):
        self.name = athlete_name
        self.dob = athlete_dob
        self.times = athlete_times

sarah = Athlete('Sarah Sweeney', '2002-6-17', ['2:58', '2.58', '1.56'])
sarah.times
james = Athlete('James Jones')
james.name
```

Try - Class Data Hiding 구현과 이해(ClassEmployee.py)

❖ 실행결과

Name : Zara ,Salary: 2000
Name : Manni ,Salary: 5000
Total Employee 2

❖ 같이 하기

```
class Employee:
    empCount = 0            → Shared among all the instances in this class
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1
    def displayCount(self):
        print("Total Employee ",Employee.empCount)
    def displayEmployee(self):
        print("Name : ", self.name,  ", Salary: ", self.salary)

emp1 = Employeeprint("Zara", 2000); emp2 = Employeeprint("Manni", 5000)
emp1.displayEmployee();       emp2.displayEmployee()
print("Total Employee “, Employee.empCount)
```

Try - import Class 구현과 이해(ImportClass.py)

❖ 실행결과

```
Name :  Zara ,Salary:  2000
Name :  Manni ,Salary:  5000
Total Employee 2
```

❖ 같이 하기 : 다른 파일 Class 사용

```
from ClassEmployee import Employee as emp
emp1 = empprint("Zara", 2000)
emp2 = empprint("Manni", 5000)
# import ClassEmployee as emp
# emp1 = emp.Employeeprint("Zara", 2000)
# emp2 = emp.Employeeprint("Manni", 5000)
emp1.displayEmployee()
emp2.displayEmployee()
```

❖ 해보기

- ➢ make ‘packages’ folder
- ➢ move ClassAthlete.py, ImportClass.py in packages
- ➢ import Athlete class

Class Inheritance(ClassInheritanceList.py)

- Syntax

```
class SubClassName (ParentClass1[, ParentClass2, ...]):
    class_suite
```

- 실행결과

```
John Paul Jonesis a Composer. - 2017.10.10
...
```

- 같이 하기

```
class NamedList(list):
    def __init__(self, a_name):
        list.__init__([])
        self.name = a_name
        self.dob = None

johnny = NamedList('John Paul Jones')
dir(johnny)                          → Debugging Mode
dir(list)                            → Debugging Mode
johnny.dob = '2017.10.10'
johnny.extend(['Composer', 'Arranger', 'Musician'])
for attr in johnny:
    print(johnny.name + 'is a ' + attr + '. - ' + johnny.dob)
```

Try - Class Inheritance 구현과 이해(ClassInheritanceDict.py)

❖ 실행결과

Composer : 2017.10.10
Arranger : 2018.10.10
Musician : 2019.10.10

Composer : 2017.10.10
Arranger : 2018.10.10
Earth : 2030.10.10

❖ 같이 하기

➢ dict 상속 받는 Class 정의
➢ 인스턴스 통한 값 입력
➢ For문 통한 출력.(순서 무관)
➢ 키,값 한 쌍 추가 후 출력.(순서 무관)

Try - Class Inheritance 구현과 이해(ClassInheritance.py)

❖ 실행결과

```
Calling child constructor          Calling child method
Calling parent method              Parent attribute : 200
```

❖ 같이 하기

```
class Parent:        # define parent class
   parentAttr = 100
   def __init__(self):       print("Calling parent constructor")
   def parentMethod(self):   ('Calling parent method')
   def setAttr(self, attr):  Parent.parentAttr = attr
   def getAttr(self):        print("Parent attribute :", Parent.parentAttr)

class Child(Parent):      # define child class
   def __init__(self):       print("Calling child constructor")
   def childMethod(self):      ('Calling child method')

c = Child()          # instance of child
c.childMethod()      # child calls its method
c.parentMethod()     # calls parent's method
c.setAttr(200)       # again call parent's method
c.getAttr()
```

Overriding Methods

❖ 실행결과

Calling child method

❖ 같이 하기

```
class Parent:        # define parent class
   def myMethod(self):
      ('Calling parent method')
class Child(Parent): # define child class
   def myMethod(self):
      ('Calling child method')
c = Child()          # instance of child
c.myMethod()
```

Overloading Methods

- Base Overloading Methods
  - __init__ (self [,args...]): Constructor (with any optional arguments)
  - __del__(self) : Destructor, deletes an object
  - __repr__(self) : Evaluatable string representation
  - __str__(self) : Printable string representation
  - __cmp__ (self, x) : Object comparison
- 실행결과

Vector(7,8)

- 같이 하기

```
class Vector:
   def __init__(self, a, b):
      self.a = a
      self.b = b
   def __str__(self):
      return 'Vector (%d, %d)' % (self.a, self.b)
   def __add__(self,other):
      return Vector(self.a + other.a, self.b + other.b)
v1 = Vector(2,10)
v2 = Vector(5,-2)
(v1 + v2)
```

Modules

❖ import module1[, module2[,... moduleN]

ex) import sys, os

sys.path

import math

from math import pi, sqrt → Try from math import *

sqrt(4.0)

❖ 내장함수(https://docs.python.org/3/library/functions.html)

>>> dir(__builtins__) : 약 70개

➢ help(내장함수명) : 정보 확인

- list() : 빈 리스트 생성
- range() : 범위의 일련 숫자를 생성하는 나열자 반환
- enumerate() : 0부터 시작해 번호 매겨진 리스트 생성
- int() : 문자열이나 숫자를 정수로 변환
- id() : 객체 고유한 식별자 반환
- next() : 나열된 데이터 구조체에서 다음 항목 반환

ex) help(list())

Pickle(00.HowToUserPickle.ipynb)

- 영구 저장, 반드시 이진 모드 접근
- dump() - 쓰기, load() - 읽기

ex) import pickle
favorite_color = { "lion": "yellow", "kitty": "red" }

pickle.dump(favorite_color, open("save.pkl", "wb"))

favorite_color_load = pickle.load(open("save.pkl", "rb"))

favorite_color_load

- 해보기

james_item = [sanitize(each_item) for each_item in james] → 지능형 리스트
james_item[1:3] → 리스트 슬라이스

➢ pickle로 dump와 load

Try - Pickle 구현과 이해

❖ 실행결과

[1, 2, 'three']

❖ 같이 하기

```
import pickle
try:
    with open('mydata.pickle', 'wb') as mysavedata:
        pickle.dump([1,2,'three'], mysavedata)
    with open('mydata.pickle', 'rb') as myrestoredata:
        re_list = pickle.load(myrestoredata)
    print(re_list)
except pickle.PickleError as err:
    print('Pickle Error : '+err)
```

최종여행물(1)

❖ 실행결과

- Copy.txt

1 Man: Is this the right room for an argument?
1 Other Man: I've told you once.
2 Man: No you haven't!
2 Other Man: Yes I have.

- Man.txt

1. Man: Is this the right room for an argument?
2. Man: No you haven't!
3. Man: When?

- Other.txt

1. Other Man: I've told you once.
2. Other Man: Yes I have.
3. Other Man: Just now.

❖ 해보기

➢ sketch.txt file 적용
➢ function 사용과 class file 분리
➢ try: except: 사용
➢ pickle dump / load

최종 목표물(2)

❖ 각 관계식 찾고, 프로그밍 하기

X	Y
0	0
1	2
2	4
3	6

X	Y
0	2
1	2.5
2	3
3	3.5

X1	X2	Y
0	2	2
1	3	4
2	4	6
3	5	8

X1	X2	Y
0	2	6
1	3	9
2	4	11.5
3	5	14.5


수고하셨습니다.