04_Python(기타)

1. 날짜(date, time, datetime)

1.1 현재 날짜

```
from datetime import date, time, datetime

# 현재 날짜
today = date.today()
print("오늘의 날짜 : " + str(today)) # 오늘의 날짜 : 2020-03-01

print("연도 : {0}, 월 : {1}, 일 : {2}".format(today.year, today.month,
today.day))
# 연도 : 2020, 월 : 3, 일 : 1

today = datetime.today() # 1/1000000 초까지
print(today)    # 2020-03-01 22:50:42.454521
```

1.2 날짜 연산

```
from datetime import date, time, datetime, timedelta
from dateutil.relativedelta import relativedelta  # years, months 사용 하기 위한
패키지

today = date.today() # 오늘의 날짜
days = timedelta(days = -1)
# days = timedelta(months = -1) years, months 사용 불가
print(today + days)     # 2020-02-29

today = datetime.today() # 오늘 날짜와 시간
delta = timedelta(hours = -3)
print(today + delta)    # 2020-03-01 19:51:43.960696

today = date.today()
days = relativedelta(months = -2)
print(today + days)     # 2020-01-01
```

2. print 문

```
for tmp in range(10):
    print("tmp : {}".format(tmp), end=", ") # 한줄 출력

# print 함수는 기본적으로 println 처럼 작동
# print 함수의 마지막 인자로 출력에 대한 제어가 가능
```

3. for 문

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

```
my_sum = 0

# for tmp in my_list:
#    my_sum += tmp

for tmp in range(len(my_list)):
    my_sum += my_list[tmp]


print("합계 : {}".format(my_sum)) # 합계 : 15

my_list = [1, 2, 3, 4, 5]
new_list = [tmp * 2 for tmp in my_list] # my_list에서 요소를 뽑아서 2배를 하고 tmp에 삽입
print(new_list)     # [2, 4, 6, 8, 10]

new_list = [tmp * 2 for tmp in my_list if tmp % 2 == 0]
print(new_list)     # [4, 8]
```

4. 함수

4.1 내장 함수

4.1.1 내장 함수의 종류

내장 함수	설 명
len()	길이 연산
abs()	숫자에 대한 절대값 반환
all()	반복 가능한 자료형에 대해서 모두가 참이면 True
any()	반복 가능한 자료형에 대해서 하나라도 참이면 True
eval()	실행 가능한 문자열을 입력받아 수치연산 수행(문자열 => 숫자열)
int()	정수로 변환
list()	list로 변환
tuple()	tuple로 변환
str()	문자열로 변환

4.1.2 정렬에 대한 함수

```
my_list = [7, 3, 9, 2, 8, 5] # 정렬이 안된 리스트를 준비
my_list.sort()               # instance의 method를 호출
                             # instance를 제어하고 return : None
print(my_list)

my_list = [7, 3, 9, 2, 8, 5]
result = sorted(my_list)     # 정렬된 결과가 리턴되고
                             # 원본객체는 변현되지 않음
print(result)
```

4.2 사용자 정의 함수

- 인자의 개수를 아는 경우

```
def my_sum(a, b, c):
    return a + b + c

result = my_sum(10, 20, 30)
print(result)
```

- 인자의 개수를 모르는 경우

```
def my_sum_1(*args):
    result = 0;
    for tmp in args:
        result += tmp
    return result;
result = my_sum_1(10, 20, 30, 40, 50)
print(result)
```

- python은 함수의 리턴값이 2개 이상처럼 보일 수 있음 하지만 실제로는 tuple형태로 반환

```
def my_func(a, b):
    result1 = a + b
    result2 = a * b
    return result1, result2 # tuple을 리턴

res1, res2 = my_func(10, 20)
print(res1, res2)
```

- 사용자 정의 함수의 scope

```
# 전역변수(global variable) 와 지역변수(local variable)
tmp = 100  # 전역변수

def my_func(x):
    #tmp    # 지역변수
    global tmp  # 전역변수 호출
    tmp += x
    return tmp
print(my_func(3))   # 103
```

5. Python의 객체지향

- class
  - 현실 세계의 개체를 프로그램적으로 모데링하기 위해서 사용하는 수단
  - 새로운 데이터 타입을 만드는 수단

```
class Student:
    #property(field)
    s_nation = "국적" # class variable
    # Constructor
    def __init__(self, n, nation):
```

```
        Student.s_nation = nation   # class variable
        self.s_name = n             # instance variable

    # method
    def display(self):
        print("국적 : {0}, 이름 : {1}".format(self.s_nation, self.s_name))

# instance 생성
stu1 = Student("홍길동", "한국")
stu1.display()
```

```
## Python - Module
## Module : 함수, 변수, class를 모아놓은 파일
## 다른 python 프로그램에서 불러와 사용할 수 있도록
## 만들어진 python 파일을 지칭
## Module을 만들면 호출하여 사용
## 이때 사용하는 keyword가 import

################################################
# 1. 먼저 my_module.txt를 jupyter notebook 에서 생성
# 2. 다음 값 입력
    def my_sum(a, b):
    return a + b

    PI = 3.1415926535
# 3. my_module.py 로 저장
################################################

# import my_module
# print(my_module.my_sum(10, 20))

# import my_module as m1  # alias
# print(m1.my_sum(10, 20))

# from my_module import my_sum
# print(my_sum(10,20))

# import my_package.my_module
# print(my_package.my_module.my_sum(10,20))

# from my_package import my_module
# print(my_module.my_sum(10,20))

from my_package.my_module import my_sum
print(my_sum(10,20))
```