Chapter. 03

파이썬 프로그래밍 언어

함수와 모듈

FASTCAMPUS
ONLINE
금융공학/퀀트

강사. 서찬웅

이번 시간에 배울 내용

1. 사용자 함수 작성 방법
2. 함수 인자값 전달
3. 지역변수, 전역변수
4. 람다함수
5. 모듈이란
6. 함수를 모듈로 만들기

I 함수란?

- 함수의 정의
  - 컴퓨터 프로그래밍에서 사용하는 개념 중에 같은 반복은 하지 말자라는 의미로 DRY(Don’t Repeat Yourself)라는 말이 있다고 합니다.
  - 특정 작업을 수행 할 때마다 코드를 여러 번 작성하는 대신 해당 코드를 가진 함수를 만들어 호출합니다.

- 함수의 생성 규칙
  - 함수는 def 키워드 뒤에 함수를 식별할수 있는 이름을 적어서 생성합니다. 함수 이름 뒤에는 항상 괄호가 붙습니다.
    - def 함수이름() :
  - () 안에 인자값을 받는다면 괄호 안에 인자 이름을 넣고 여러 개일 경우 쉼표로 분리합니다.

I 함수 예제

- 함수의 생성 예제
  - 아래 코드를 작성하고 실행하면 아무것도 출력이 되지 않습니다. 정의와 호출은 별개의 이야기입니다.

```
def say():
    print ("안녕하세요")
```

<함수의 정의>

```
say()
```

```
안녕하세요
```

<함수의 호출>

  - 함수를 호출 할 때는 항상 소괄호를 붙어야 합니다.
  - 함수 사용은 함수 생성 뒤에 사용할 수 있습니다. 함수를 생성 전에 호출하면 다음과 같은 오류가 발생합니다.

```
say2()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-88b0a69d2425> in <module>
----> 1 say2()

NameError: name 'say2' is not defined
```

I 함수의 인자값(매개변수)

- 함수는 인자(매개변수)를 전달 받아서 사용할 수 있습니다.
- 인자는 함수 안에서 사용할 수 있는 변수 이름입니다. (지역변수)

```
def say2(name):
    print ("{} 안녕하세요".format(name))
```

```
say2('수강생님')
```

```
수강생님 안녕하세요
```

  - name은 say2 사용자 함수 안에서 사용하는 변수 이름입니다.
- 사용자 함수 정의할 때 인자를 정의하면 함수 호출 때는 반드시 인자값을 전달해야 합니다. 그렇지 않으면 아래와 같은 오류가 발생합니다.

```
say2()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-88b0a69d2425> in <module>
----> 1 say2()

TypeError: say2() missing 1 required positional argument: 'name'
```

I 함수에 여러 개의 인자값을 전달하기

- 함수를 정의할 때 인자 값을 여러 개를 정의할 수 있습니다.
- 정의할 때 소괄호() 안에 여러 인자 이름을 콤마(,)로 구분하여 입력하시면 됩니다.
- 호출 할 때도 인자 값을 콤마(,)로 분리해서 호출하면 같은 위치에 있는 값들이 매핑되어 실행이 됩니다.

```
def stock(name, start):
    print ("{}의 시가는 {}입니다.".format(name, start))
```

```
stock('삼성전자','43000')
```

```
삼성전자의 시가는 43000입니다.
```

  - name = 삼성전자, start = 43000이 대입되어 함수가 실행이 되었습니다.
- 인자 이름을 각각 지정해서 함수에 전달하면 이름에 맞게 대입되어 값이 전달이 됩니다.

```
stock(start=43000, name='삼성전자')
```

```
삼성전자의 시가는 43000입니다.
```

  - 인자값 이름을 지정하여 전달할 때는 오타등의 이유로 이름을 잘못 부르면 오류가 발생합니다.
- 하나의 인자에 여러 개의 값을 전달 할 수도 있습니다. 즉 가변인수를 지원합니다. 인자 값에 * 기호를 붙입니다.

I 함수의 인자 값을 기본값을 설정하기

- 인자의 기본값을 원할 경우 함수 정의할 때 원하는 인자에만 설정할 수 있습니다.

```
def stock2(start, name = '삼성전자'):
    print ("{}의 시가는 {}입니다.".format(name, start))
```

```
stock2(43000, '삼성전자')
```

```
삼성전자의 시가는 43000입니다.
```

  - Q : stock(name, start)으로 정의했는데, 왜 기본 값이 있는 함수에는 인자값의 순서가 변경되었을까요?
  - A : 함수를 정의할 때 기본값이 없는 인자가 기본값이 있는 인자보다 무조건 앞에 나와야 합니다. 그렇지 않으면 오류가 발생합니다.

I 함수의 반환 값(return)

- 함수는 특정 작업을 수행하고 return 구문을 사용하여 처리된 데이터를 반환할 수 있습니다.
- 원하는 곳에서 어떤 데이터라도 반환할 수 있습니다.
- 그 역할을 하는 키워드가 return입니다. return을 호출하게 되면 값을 바로 반환하고, 그 함수는 종료가 됩니다.

```
def odd_or_even(number):
    if number % 2 == 0:
        return '짝수'
    else:
        return '홀수'
    print ('이 프린트 문이 실행이 될까요?')
```

```
odd_or_even(15)
```

```
'홀수'
```

```
def odd_or_even2(number):
    if number % 2 == 0:
        return number, '짝수'
    else:
        return number, '홀수'
    print ('이 프린트 문이 실행이 될까요?')
```

```
odd_or_even2(15)
```

```
(15, '홀수')
```

- return 값이 2개 이상일 때는 return 된 결과 값이 튜플로 구성되어 전달이 됩니다.
- 튜플로 구성된 값은 언패킹을 사용해서 각각의 변수에 입력할수 있습니다.

```
numb, text = odd_or_even2(15)
```

```
numb
```

```
15
```

```
text
```

```
'홀수'
```

I 지역변수란?

- 변수가 사용되는 프로그램의 영역을 유효범위라고 합니다.
- 아래 calculateTax 함수의 매개변수로 선언된 price는 함수 내부에서만 사용하는 변수입니다.
- 함수 외부에서 호출하게 되면 정의되지 않은 값이라고 오류가 발생합니다.

```
def calculateTax(price, tax_rate):
    total = price + (price * tax_rate)
    return total
```

```
my_price = float(input("Enter a price : "))
```

```
Enter a price : 20000
```

```
total_price = calculateTax(my_price, 0.06)
```

```
print ("price = {}, Total = {}".format(my_price, total_price))
```

```
price = 20000.0, Total = 21200.0
```

```
print (price)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-58-3b154833d11b> in <module>()
----> 1 print (price)

NameError: name 'price' is not defined
```

I 전역변수

- 프로그램에서 정의된 변수명은 사용자 함수에서 변경하지 않은 이상 전역변수로 선언됩니다.
- 전역변수로 선언되면 프로그램 전체에서 사용이 가능합니다.

```
def calculateTax(price, tax_rate):
    total = price + (price * tax_rate)
    print ("your price {}".format(your_price))
    return total
```

your_price가 전역변수로 선언되어 프로그램 전체에서 사용가능하다

```
your_price = 99999999
my_price = float(input("Enter a price : "))
```

```
Enter a price : 20000
```

```
total_price = calculateTax(my_price, 0.06)
```

```
your price 99999999
```

```
print ("price = {}, Total = {}".format(my_price, total_price))
```

```
price = 20000.0, Total = 21200.0
```

I 전역변수와 지역변수와 동일한 이름의 변수의 우선순위

- 프로그램이 생성할 때 프로그램 코드의 길이가 길어지고 나면 동일한 변수명을 전역, 지역에 사용할 때가 발생할 수 있습니다.
- 이럴 경우 꼭 기억하실 내용은 지역변수가 전역변수보다 우선 순위가 높다 라는 점을 기억하세요.
- 함수 내부 곧 지역변수에서 동일한 이름의 전역변수를 사용하고 싶다면 global 키워드를 사용하면 됩니다.

```
def test():
    x = 1
    print (x, id(x))

x = 10
test()
print (x, id(x))
```

```
1 140712295306048
10 140712295306336
```

```
def test():
    global x
    x = 1
    print (x, id(x))

x = 10
test()
print (x, id(x))
```

```
1 140712295306048
1 140712295306048
```

람다함수

- 람다 함수는 이름 없는 한 줄 짜리 함수입니다.
- 사용 방법은 lambda 변수명 : 표현식

```
def my_function(x):
    return x**2
```

```
my_function(5)
```

```
25
```

```
(lambda x : x**2)(5)
```

```
25
```

```
dict_sorted = {'f' : 10, 'b' : 5, 'a' : 15, 'd' : 1}
```

```
sorted(dict_sorted, key=lambda x : dict_sorted[x])
```

```
['d', 'b', 'f', 'a']
```

```
sorted(dict_sorted, key = lambda x : x)
```

```
['a', 'b', 'd', 'f']
```

구분	def로 정의되는 함수	lambda 함수
문/식	문(statement)	식(expression)
함수의 이름	def 다음에 지정된 이름으로 생성한 함수 객체를 치환한다	함수 객체만을 생성한다
몸체	한 개 이상의 문을 포함한다	하나의 식만 온다
리턴	return 문에 의해 명시적으로 리턴 값이 지정된다	식의 결과가 리턴된다
내부 변수 선언	지역 영역에 변수를 생성하고 사용하는 것이 가능하다	지역 영역에 변수를 생성하는 것이 가능하지 않다

I 모듈(module)

- 모듈은 서로 관련이 있는 프로그램 코드들을 모아놓은 파일 입니다.
- 파이썬에서는 모듈은 커다란 프로그램을 구성하는 작은 조각을 의미합니다.
- 파이썬 프로그램으로 된 파일이거나 혹은 c언어로 만들어진 파이썬 확장 파일일 수도 있습니다.
- 파이썬을 레고라고 생각하면 모듈은 레고 시리즈라고 생각하시면 편할거 같습니다.
- 레고는 단순하지만 레고 상품 마다 난이도도 다르고, 사용자가 원하는 상품만 구매하면 되듯이 파이썬도 모든 모듈을 다 사용할 필요 없이 원하는 모듈만 사용하면 됩니다.

<사용자가 원하는 시리즈만 구매할수 있는 레고사이트>
출처 : https://shop.lego.com/ko-KR/category/themes

I 모듈은 왜 사용할까?

- 모듈은 표준 모듈, 사용자 정의 모듈, Third Party 모듈이 존재합니다.
  - 표준 모듈은 파이썬 설치시 기본적으로 제공되는 모듈로 파이썬 기본 라이브러리라고도 합니다.
  - 사용자 정의 모듈은 사용자가 직접 만들어서 사용하는 모듈입니다.
  - Third Party 모듈은 Selenium같은 기본적으로 제공하는 모듈이 아닌 외부 제 3자가 만든 모듈입니다.
- 왜 모듈을 사용할까요?
  - 코드를 재 사용할 수 있습니다.
  - 파일의 크기가 더 작아져서 코드에서 원하는 것을 찾기가 쉽습니다.
  - 모듈은 시스템 구성 요소의 기본 단위입니다. 파이썬에서 모든 것을 모듈 단위로 분리해서 관리하고 있습니다. 하나의 시스템을 모듈 단위로 분리해서 설계하는 것은 여러 면에서 작업의 효율을 높일 수 있습니다.
  - 모듈은 별도의 Name Space를 제공합니다. 다른 모듈과 겹치지 않은 공간을 가지기 때문에 다른 모듈의 함수나 변수에 영향을 받지 않고 독립적인 작업을 할 수 있다.

I NameSpace

- A부터 F까지의 클래스가 있는 학교가 있습니다. 이 학교에는 신기하게도 A~F클래스에 마이클이라는 이름을 가진 학생이 존재한다고 합니다. 학교 전체에서 보면 6명의 마이클이 존재하는 거겠죠.
- 강당에 모든 학생들을 모아놓구 교장 선생님이 외칩니다. '마이클~~~ 자넨 퇴학이야'
- 6명의 마이클은 모두 깜놀할거 같습니다.
- 이렇게 교장이 전교생을 모아놓구 외치는 것을 Global NameSpace에 외치는 것과 같습니다.
- Global NameSpace에 외치면 위의 상황같이 교장이 원하는 한명을 지목하기 힘듭니다. 그래서 교장 선생님은 다시 외칩니다. 'A 클래스의 마이클 퇴학이야~'
- 순간 A클래스의 마이클만 반응합니다. 나머지 마이클들은 안도의 한숨을 쉽니다.
- 이렇게 교장 선생님이 A클래스라고 구역을 분리했기 때문에 A클래스의 마이클만 해당되는 것입니다.
- 이를 Local NameSpace라고 합니다.

I NameSpace

- 다시 파이썬으로 돌아와서 파이썬에서 모듈을 가져오는 방법이 2가지 존재합니다.
  - from <모듈> import <함수 or 클래스>
- import 'A클래스'
  - 이렇게 선언하게 되면 내가 원하는 마이클을 호출할때마다 A클래스.마이클 이라고 해야 합니다.

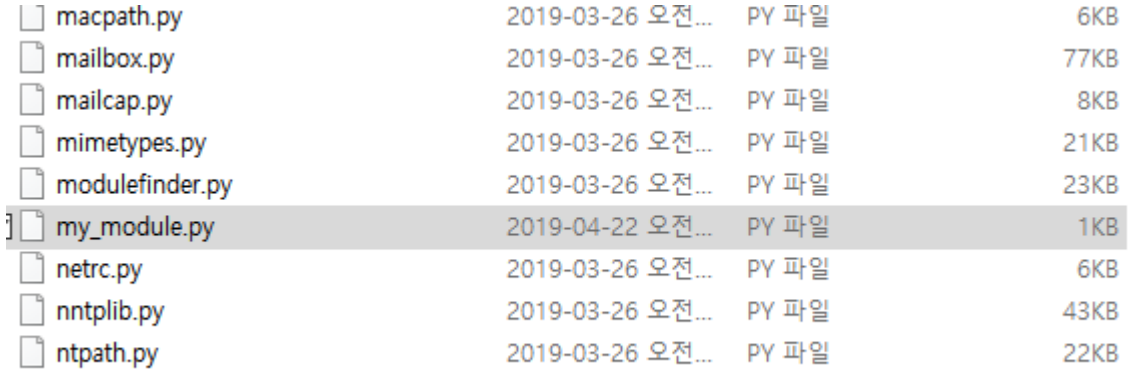
- from A클래스 Import '마이클'
  - A클래스의 있는 마이클만 호출하겠다는 의미입니다.
  - 이제부터 마이클을 호출하면 A클래스의 마이클만 대답을 할 것입니다.
  - '마이클'이라는 이름을 계속 부르면 나머지 5명에게 왠지 미안할거 같습니다. 그래서 '마이클'에게 '미남' 이라는 별명을 붙여줍니다.
    - from A클래스 import '마이클' as '미남'
  - 만약 A클래스나 특정 클래스의 학생들을 모두 호출하고 싶다면 ...
    - from A클래스 import *

I 사용자 모듈 만들기

- 아래와 같이 my_module.py를 작성합니다.

my_module.py

```
1  """모듈명 : my_module
2      c_to_f(celsius) : 섭씨를 화씨로 변환합니다.
3      f_to_c(Fahrenheit) : 화씨를 섭씨로 변환합니다."""
4
5  def c_to_f(celsius):
6      Fahrenheit = celsius * 9.0 / 5 + 32
7      return Fahrenheit
8
9  def f_to_c(Fahrenheit):
10     celsius = (Fahrenheit - 32) * 5 / 9
11     return celsius
```



- anaconda 설치 폴더의 Lib 밑에 모듈이 저장되어 있습니다. 해당 폴더에 my_module.py를 저장합니다.
- 저장 뒤에는 모듈을 호출 할수 있습니다.

```
import my_module

my_module.c_to_f(22)
```

```
71.6
```

```
from my_module import c_to_f

c_to_f(22)
```

```
71.6
```

I 정리

- 사용자 함수 만들기
- 인자 값 전달하기
- return 키워드
- 지역변수, 전역 변수
- 람다함수
- 모듈 및 from, import의 의미
- 사용자 모듈 만들기

감사합니다

FAST CAMPUS
ONLINE
서찬웅 강사.