Python 함수

키워드 : [Python 함수 (https://docs.python.org/3.7/reference/compound_stmts.html#def)](https://docs.python.org/3.7/reference/compound_stmts.html#def)

우리는 지난 챕터에서 자바스크립트에 대해 공부하며, 함수(Function)에 대해서 알아보았습니다. 함수는 특정 코드블록에서 실행되는 코드 구문으로, 전체적인 코드의 흐름을 제어하고, 중복되는 코드를 제거하기 위해 사용할 수 있었습니다. 이번 장에서는, 파이썬에서 함수를 정의하고 사용하는 방법에 대해서 알아보도록 하겠습니다.

함수 정의

파이썬에서는 함수를 정의하기 위해 def라는 키워드를 사용하며, 압력 값인 파라미터 값을 가질 수 있습니다. 또한, 반환(return) 값은 존재할 수도 있고, 없을 수도 있습니다.

아래 코드는 파이썬에서 함수를 정의하기 위해 사용되는 기본 형식의 예입니다.

```
# 기본 형식
def 함수명():
    # 실행 코드

def sum(a, b):
    return a + b
```

def 키워드를 이용해서 함수를 정의하고, 함수의 이름은 의미를 갖는 이름으로 정의합니다. 콜론(:)으로 코드 블록의 시작을 명시합니다.

파이썬에서 코드에서 작성 되는 모든 것은 객체라고 말씀드렸습니다. 파이썬에서 우리가 정의한 함수도 마찬가지로, 함수 객체가 됩니다. (당연하죠..?^^) 더 나아가서, 파이썬에서 함수는 1급 객체(First-class Object)라고 표현합니다. 이 1급 객체에 대한 내용에 대해 더 이야기해보도록 하겠습니다.

먼저, 1급 객체는 세 가지 특징을 만족해야 합니다.

1. 함수를 변수에 할당할 수 있어야 한다.
2. 함수를 반환 값으로 사용할 수 있어야 한다.
3. 함수를 함수의 인자로 입력할 수 있어야 한다.

위 내용을 하나하나 살펴보도록 하겠습니다.

```
def sum(a, b):
    return a + b

s = sum
print(s(1, 2)) # 3
```

함수는 선언만 된 상태에서는, 함수 구문 안에 정의된 구문이 실행되지 않습니다. 호출(Called)이라는 작업이 필요하며, 호출을 하기 위해서 함수의 이름을 통해 호출하게 됩니다. 이러한 함수를 우리가 임의로 선언한 변수에 할당하면, 해당 변수의 이름으로 함수를 호출할 수 있습니다.

변수에 할당할 수 있다면, 파이썬에서 사용되는 컨테이너 객체에도 할당할 수 있다는 말이 됩니다.

```
def sum(a, b):
    return a + b

def multiply(a, b):
    return a * b

func = [sum, multiply]
print(func[0](1, 1), func[1](2, 2)) # 2, 4
```

위 내용은 1급 객체를 만족하기 위한 첫 번째 조건인 함수를 변수에 할당할 수 있어야 한다.를 만족합니다. 계속해서 또 다른 코드 예제를 보도록 하겠습니다.

```
# 1번 코드
def sum(a, b):
    return a + b

# 2번 코드
def func(a, b):
    return sum(a, b)

print(func(1, 1)) # 2
```

1번 코드는 sum이란 함수를 선언하고 있습니다. 중점적으로 봐야 할 내용은, 2번 코드에서 sum 함수를 반환 값으로 사용하고 있다는 점입니다. 함수를 반환 값으로 사용할 수 있으므로, 두 번째 조건인 함수를 반환 값으로 사용할 수 있어야 한다.에 만족합니다. 계속해서, 마지막 조건에 대해서 이야기해보도록 하겠습니다.

```
# 1번 코드
from time import time

# 2번 코드
def run_time(f):
    # 시작 시간
    start = time()
    f
    print(time() - start) # 종료 시간 - 시작 시간(총 수행 시간)을 계산하여 정수 값으로
출력합니다.

# 3번 코드
def sum():
    sum = 0
    while sum < 100:
        sum += 1
        print(sum) # 1 2 3 ... 100

# 4번 코드
run_time(sum()) # 0, 코드 실행시간 출력
```

위 코드는 임의로 정의한 함수의 실행 시간을 구하기 위한 간단한 예시 코드입니다. 위 코드 내용을 하나씩 알아보도록 하겠습니다.

1. 우선, 1번 코드의 time 모듈은 파이썬 언어에서 기본으로 제공되는 모듈이며, 시간과 관련된 많은 함수가 정의되어 있는 모듈입니다. 자바스크립트 날짜 객체에서 이야기했던, 1970년 1월 1일 (UTC) 시간을 기준(0)으로, 흐른 시간을 반환합니다. time 모듈 안의 time 함수를 사용하기 위해 import한 구문입니다.

2. 2번 코드는 run_time이라는 이름으로 함수를 정의하고 있고, 정의된 함수의 인자로 f(이름은 임의대로 작성합니다.) 함수를 넘겨 받습니다. 함수의 실행 시간을 측정하기 위해서, 함수 코드가 실행되기 전에 시작 시간을 먼저 구합니다. 인자로 받은 함수의 코드 실행이 종료되면, 종료 시간 - 시작 시간 연산을 통해 총 실행 시간을 구하게 됩니다.

3. 3번 코드는 반복문을 통해 1 ~ 100의 수를 단순 더하는 연산을 하는 함수입니다.

4. run_time 함수의 인자로 sum 함수를 넘겨줍니다. (0 ~ 100 까지 더하는 연산은 1초가 걸리지 않습니다.)

위 내용은 1급 객체를 만족하기 위한 세 번째 조건인 함수를 함수의 인자로 입력할 수 있어야 한다.를 만족합니다. 아마도, 이런 생각이 드실 겁니다. "이거 자바스크립트 함수도 다 되는 거 아니야?" 그렇습니다. 자바스크립트의 함수도 1급 객체입니다. 위와 같은 내용이 해당하는 프로그래밍 언어도 존재하고, 그렇지 않은 언어도 있습니다.

지금 당장은 위 내용의 의미를 이해하기 어려울 수 있습니다. 다만, 자바스크립트와 파이썬 언어에서 함수는 1급 객체이고, 그 특징을 통해서 다양한 코드를 작성할 수 있다는 점만 기억해주시면 됩니다. 계속해서 파이썬 함수와 관련된 내용에 대해서 알아보도록 하겠습니다.

기본값(Default Argument Values)

우선 파이썬 함수에는 기본 인자라는 것이 존재합니다. 함수의 인자는 정의되어 있지만, 인자가 입력 값으로 주어지지 않을 경우에 기본 값으로 사용할 인자라고 할 수 있습니다. 기본 형식은 “파라미터 명 = 기본값” 형식으로 선언합니다.

```
def sum(x, y, z = 10):
    return x + y + z

print(sum(1, 1)) # 12
print(sum(1, 1, 1)) # 3
```

위 코드 예시를 통해 알 수 있듯이, 기본 인자는 입력 값으로 주어지지 않으면, 기본 값으로 설정되는 값을 이야기합니다. 첫 번째 sum 함수의 경우 인자가 따로 주어지지 않았기 때문에, 인자 z의 기본 값 10이 할당되게 됩니다. 반대로, 인자의 입력 값이 주어질 경우, 주어진 인자의 입력 값이 함수의 연산 값이 됩니다.

```
i = 0
def test(a = i):
    print(a)
i = 10
# 1번 코드
test() # 0
# 2번 코드
test(i) # 10
```

기본 값 인자를 이용한 함수 정의 시 주의해야 할 중요한 특징이 있습니다. 바로, 기본 값 인자의 값이 특정 값이 아닌, 변수로 할당될 수 있다는 점입니다. 변수의 이름으로 함수의 기본 값 인자가 할당된 경우엔 선언과 할당 순서가 중요합니다.

위 코드의 예시를 살펴보면, 1번 코드의 경우 기본 값으로 선언된 값 중에 test 함수보다 먼저 선언된 i = 0의 값이 기본 인자의 값으로 할당됩니다. 반대로, i 인자를 함수에 입력 값으로 명시적으로 할당할 경우엔, i = 10으로 선언된 변수의 값이 할당된 것을 확인할 수 있습니다.

이러한 특징 때문에, 기본 값 인자로 변수를 할당할 경우엔 변수를 선언하고 할당하는 순서에 유의해야 한다는 점 기억해주세요! :)

이름 있는 인자(Named Argument)

파이썬에서는 이름 있는(Keyword Argument 혹은, Named Argument) 인자라는 것이 존재합니다. 기본 형식은 “인자명 = 값”입니다. 가독성 있고 직관적인 코드를 작성할 수 있는 장점이 있습니다.

```
def sum(x, y, z = 5):
    return x + y + z

print(sum(x = 10, y = 10, z = 10)) # 30
print(sum(y = 10, x = 10)) # 25
print(sum(z = 10, y = 10, x = 10)) # 30
```

위 코드 예시는 기본 값 인자와 이름 있는 인자를 이용한 함수 선언의 간단한 예시입니다. 함수 선언 시에 사용한 인자의 이름을 특정 지어 입력 값을 할당하면, 입력 값의 순서에 상관없이 함수 내의 실행 코드에서 자동으로 해당 인자의 이름을 통해 연산을 진행할 수 있습니다. 또한, 기본 값 인자인 z의 경우 입력 값이 없으면, 자동으로 5가 할당된 것을 확인할 수 있습니다.

가변 길이 인자(Arbitrary Argument Lists)

우리가 함수를 정의할 때, 함수의 입력 값으로 할당된 인자의 갯 수가 정해져 있지 않을 수 있습니다. 이런 경우에 변수명 앞에 애스터리스크(*) 기호를 이용해 가변 길이 인자를 선언할 수 있습니다.

```
def sum(*number, fl):
  sum = 0
  for i in number:
    sum += i
  return sum + fl

print(sum(1, 2, 3, 4, 5, fl = 1.0)) # 16.0
print(sum(fl = 1.0, 1, 2, 3, 4, 5)) # SyntaxError 에러 발생!
```

위 코드 예시와 같이 애스터리스크(*)가 붙은 가변 인자의 경우 인자의 개수의 제한 없이 입력 값을 함수에 넘겨줄 수 있습니다. 단, 한 가지 주의할 점이 있습니다. 가변 길이 인자의 선언 순서인데요, 위 코드와 같이 `fl`라는 이름 있는 인자를 이용해서 인자를 특정 지어 값을 할당하고 싶은 경우에는, 가변 길이 인자보다 뒤에 인자를 선언해야 합니다.

반환 값(Return type)

강의에서도 말씀드렸듯이, 파이썬 언어는 하나의 함수에 여러 개의 반환 값이 존재할 수 있습니다. 여기서 반환되는 결과 값은 각각의 값을 따로 반환하는 것이 아닌, 하나의 튜플로 반환하는 것입니다.

```
def test(a, b):
    return a, b

print(test(1, 1)) # (1, 1)
t = test(1, 1)
print(t, type(t)) # (1, 1), <class 'tuple'>
a, b = test(1, 1)
print(a, b) # 1 1
t[0], t[1] = 10, 20 # 튜플의 값을 변경하려 했기 때문에, TypeError 에러 발생!
```

파이썬 튜플의 특성을 이용해 하나의 함수에서 동작한 여러 작업에 대한 반환 값을 여러 개 가지게 할 때 사용하면 유용합니다. 단, 튜플 컬렉션은 값이 변할 수 없는 불변(Immutable) 타입이라는 것을 컬렉션을 이야기할 때 알아보았습니다. 그렇기 때문에, 그 특징을 잘 생각하면서 반환 값을 결정해야 합니다.

이번 장에서는 파이썬 함수에 대해서 알아보았습니다. 파이썬 함수와 관련된 내용은 자바스크립트와 마찬가지로, 다룰 내용이 굉장히 많습니다. 예를 들어, Decorator (https://docs.python.org/3.7/glossary.html#term-decorator)와 같은 고급 코딩 스타일 등이 많이 존재합니다. 위에서 다룬 내용은 파이썬에서 함수를 선언하기 위한 최소한의 내용이라고 생각해주시길 바라겠습니다.

참고로, 파이썬 개발자들을 위해서 파이썬스럽게(Pythonic) 코드를 작성할 수 있도록, 다양한 내용을 담은 PEP8 (https://www.python.org/dev/peps/pep-0008/)라는 파이썬 코딩 스타일 가이드가 있습니다. 단순히 코드를 작성하는 것이 아닌, 많은 사람들에 의해 검증되고, 권장되는 표준적인 스타일로 코드를 작성하는 것도 고려해보세요!