고급 PYTHON PROGRAMMING

SSL SEMINAR

20190431 박규현

목차

01

Python 소개

Gudio Van Rossum

1989년 크리스마스에 만들어짐

영국 코미디 그룹 몬티 파이썬에서 이름을 땀

01

Python 소개 Features of Python

1. 플랫폼 독립적인 인터프리터 언어

- 코드와 인터프리터만 있다면 어디서든 실행 가능

2. 완전 객체 지향

- 모든 것이 객체

3. 동적 타이핑 언어

- 코드를 실행하던 중에 타이핑

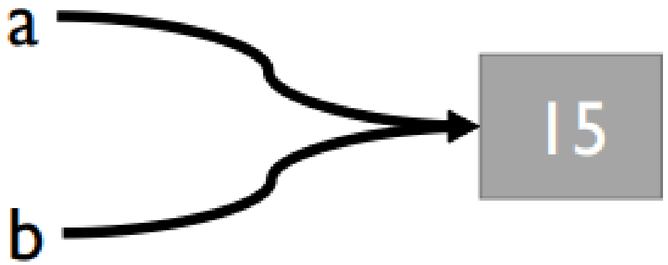
01

Python 소개 Features of Python Variable

모든 변수는 메모리 주소를 가리킴 !!

```
>>> a = 15
>>> b = 4
>>> a + b
19
```

```
>>> a = 15
>>> b = a
>>> b
15
```



Python 대입 == 메모리 주소 복사

- 원시 자료형도 결국 모두 포인터
- 변수 명 = 일종의 이름표 역할

01

Python 소개 Features of Python Variable

적당한 크기의 Primitive Data 대입은 기존 객체 할당

파이썬은 모두 메모리와 관련이 있고, 파이썬에서는
최적화를 알아서 잘 해줌

Text에서도 동일하게 작용함

```
>>> a = 1
>>> b = 1
>>> a is b
True

>>> a = 13453436
>>> b = 13453436
>>> a is b
False
```

02

Python Programming - Data Structure

Packing & Unpacking

Packing : 여러 데이터를 묶기

Unpacking : 묶인 데이터 풀기

```
>>> t = [1, 2, 3, 4, 5]
>>> a, b, c, _, _ = t
>>> c
3

>>> a, *b, c = t
>>> a, b, c
(1, [2, 3, 4], 5)
```

*(Asterisk)로 남는 요소 리스트 담기 가능

02

Python Programming - Function

Variable Capture

```
var = 1

def function():
    print (var)

var += 1
function()   # 2
```

이름표 선언 중

Capture : 함수 안에 변수를 넣게 되면 변수가 capture가 되어 해당 변수가 저장

함수의 순수성을 지키지 못함

02

Python Programming - Function

```
number = 10

def print_closure_factory(number):
    def print_closure():
        print(number)

    return print_closure

print_5 = print_closure_factory(5)
print_10 = print_closure_factory(10)

number += 10
print_5()
print_10()
```

Closure

Factory 형식으로 사용

Python에서는 함수도 일반 객체

-> 변수로 할당, 함수 반환이 가능

02

Python Programming - Function

```python
def add(var):
    return var + 2

def multiply(var):
    return var * 2

def factory(function, n):
    def closure(var):
        for _ in range(n):
            var = function(var)
        return var

    return closure

print(factory(add, 4)(10))
print(factory(multiply, 4)(3))
```

Closure Example

```
(base) PS C:\Users\GyuHyeon\Documents\랩 세미나\python programming>
s-python.python-2021.12.1559732655\pythonFiles\lib\python\debugpy\lau
랩 세미나\python programming\1.py'
18
48
```

02

Python Programming - Function

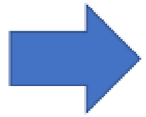
Decorator

```python
def print_closure_factory(function):
    def print_closure(var):
        print("Input:", var)
        out = function(var)
        print("Output:", out)

    return print_closure

def add(var):
    return var + 2

print_add = print_closure_factory(add)
print_add(10)
```



```python
def print_decorator(function):
    def print_closure(var):
        print("Input:", var)
        out = function(var)
        print("Output:", out)

    return print_closure

@print_decorator
def add(var):
    return var + 2

add(10)
```

- 함수 하나를 인자로 받아 같은 형태의 함수를 반환하는 함수
- '@'를 사용하여 함수를 꾸미는데 사용 가능

```
(base) PS C:\Users\GyuHyeon\Documents\랩 세미나\python
on programming'; & 'python' 'c:\Users\GyuHyeon\.vscode
ython\debugpy\launcher' '14802' '--' 'c:\Users\GyuHyeo
Input : 10
Output : 12
```

02

Python Programming Comprehension

List, Dcitionary 등을 빠르게 만드는 기법

```
result = []
for i in range(10):
    result.append(i * 2)
```

```
result = [i * 2 for i in range(10)]
```

```
result = {}
for i in range(10):
    result[str(i)] = i
```

```
result = {str(i): i for i in range(10)}
```

02

Python Programming Generator

range 함수의 경우 숫자를 하나씩 생성하여 반환 **요소를 하나씩 생성해서 반환하는 객체**

```
def my_range(stop):
    number = 0
    while number < stop:
        yield number
        number += 1

for i in my_range(5):
    print (i)
```

- 함수에 yield를 사용할 시 generator 가 됨
- yield 하는 위치에서 값을 반환
- 다시 값을 요청 받을 시 yield 다음 줄 부터 실행
- Return 될 시 StopIteration Error 발생으로 멈춤
- Sequence 전체 생성이 아니라 메모리 효율적

02

Python Programming - Object-Oriented

파이썬에서 제공되는 기능

예약어

- 일종의 문법적인 요소
- 괄호를 쓰지 않음
- 재정의 불가능
- if … else …, del, assert 등

내장함수

- 기본 정의된 함수
- 별개의 함수 사용
- 재정의 가능
- len( ), sum(), range() 등

메소드

- 객체 내에 정의된 함수
- .method( ) 으로 접근
- Overriding
- .append( ), .insert() 등

02

Python Programming - Object-Oriented

Magic Method

Initializer

```python
class Student:
    def __init__(self, name: str, sid: int):
        self.name = name
        self.sid = sid
        self.classes = set()
```

- 객체를 생성할 때 호출됨
- 일반적으로 객체의 속성을 초기화하는데 사용

Destroyer

```python
class Student:
    def __del__(self):
        self.classes.clear()
```

- 객체를 소멸할때 호출됨
- 파이썬은 Garbage Collection으로 메모리 관리

02

Python Programming - Object-Oriented

__getitem__ __setitem__

Magic Method

```python
class DoubleMapper(object):
    def __init__(self):
        self.mapping = {}

    def __getitem__(self, index):                # Indexing get
        return self.mapping.get(index, index * 2)

    def __setitem__(self, index, item):          # Indexing set
        self.mapping[index] = item

mapper = DoubleMapper()
print(mapper[10], mapper[1, 2])
mapper[10] = 15
print(mapper[10], mapper[1, 2])
```

```
(base) PS C:\Users\GyuHyeon\Documents\랩 /
on programming'; & 'python' 'c:\Users\GyuH
ython\debugpy\launcher' '14363' '--' 'c:\U
20 (1, 2, 1, 2)
15 (1, 2, 1, 2)
```

[] 연산 - Inedxing을 재정의

빈 Dctionary에 디폴트 값 저장

02

Python Programming - Object-Oriented

__getitem__ __setitem__

```python
class DoubleMapper(object):
    def __init__(self):
        self.mapping = {}

    def __getitem__(self, index):                # Indexing get
        return self.mapping.get(index, index * 2)

    def __setitem__(self, index, item):          # Indexing set
        self.mapping[index] = item

mapper = DoubleMapper()
print(mapper[10], mapper[1, 2])
mapper[10] = 15
print(mapper[10], mapper[1, 2])
```

Magic Method

```
(base) PS C:\Users\GyuHyeon\Documents\랩 /
on programming'; & 'python' 'c:\Users\GyuH
ython\debugpy\launcher' '14363' '--' 'c:\U
20 (1, 2, 1, 2)
15 (1, 2, 1, 2)
```

[] 연산 - Inedxing을 재정의

빈 Dctionary에 디폴트 값 저장

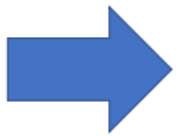
02

Python Programming - Object-Oriented for

실제 for 문에서 일어나는 일은 ?

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

for elem in seq:
    print(elem)
```



```
seq = list([1, 2, 3, 4, 5])

iterable = iter(seq)
while True:
    try:
        elem = next(iterable)
    except StopIteration:
        break
    print(elem)
```

iter - 내장 함수

해당 객체의 순환자 반환

__iter__ 호출

next - 내장 함수

해당 순환자를 진행

__next__ 호출

감사합니다!

20190431 박규현