고급 PYTHON PROGRAMMING - 2

SSL SEMINAR

20190431 박규현

목차

1 지난 세미나 문제점

2 Python Programming

01

Python - 1

.get()은 왜 get인가?

__getitem__

```
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])
```

dict에서 참조하는 함수이므로
get이라는 이름을 사용 하는 것 !

< .get()의 순서>

1. index 값 가지고 오기

2-1. 있으면 해당 값 return

2-2 없으면 index * 2 reutrn

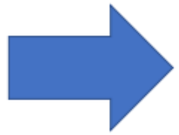
01

Python - 1

순환자 사용 할때의 list 선언

```
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)
```

버전의 문제 !!

3.5에서는 list()없이는 사용 불가

3.6에서는 없이 사용 가능 !

01

Python - 1

range() 실행 순서

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

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

my_range가 다시 호출되면 yield 부터 다시 시작

my_range가 다음 숫자 요청 -> number return

- my_range 함수에서 만들어진 iterator에 의해 실행

02

Python Programming - Property

Getter & Setter 명시적 설정 가능

```python
class Circle(object):
    PI = 3.141592
    def __init__(self, raidus=3.):
        self.radius = raidus

    def get_area(self):
        return Circle.PI * self.radius ** 2

    def set_area(self, value):
        self.radius = (value / Circle.PI) ** .5

circle = Circle(5.)
print(circle.get_area())
circle.set_area(10)

print(circle.radius)
```

```python
class Circle(object):
    PI = 3.141592
    def __init__(self, raidus=3.):
        self.radius = raidus

    @property
    def area(self):
        return Circle.PI * self.radius ** 2

    @area.setter
    def area(self, value):
        self.radius = (value / Circle.PI) ** .5

circle = Circle(5.)
print(circle.area)

circle.area = 10.
print(circle.radius)
```

Encapsulation 등에 활용

02

Python Programming - Static & Class Method

```
class Number:
    Constant = 10

    @staticmethod
    def static_factory():
        obj = Number()
        obj.value = Number.Constant
        return obj

    @classmethod
    def class_factory(cls):
        obj = cls()
        obj.value = cls.Constant
        return obj

number_static = Number.static_factory()
number_class = Number.class_factory()
print(number_static.value, number_class.value)
```

파이썬에는 2가지 정적 함수 존재

객체에서는 접근 불가능

일반적으로 Class.method 형태로 사용

Static Method

Class Method

02

Python Programming - Static & Class Method

```python
class Number:
    Constant = 10

    @staticmethod
    def static_factory():
        obj = Number()
        obj.value = Number.Constant
        return obj

    @classmethod
    def class_factory(cls):
        obj = cls()
        obj.value = cls.Constant
        return obj

number_static = Number.static_factory()
number_class = Number.class_factory()
print(number_static.value, number_class.value)
```

Static Method

- staticmethod 꾸밈자 사용
- 특별한 argument를 받지 않음
- 일반적으로 class 내 유틸함수로 사용
- Class를 일종의 Namespace로 활용

02

Python Programming - Static & Class Method

```python
class Number:
    Constant = 10

    @staticmethod
    def static_factory():
        obj = Number()
        obj.value = Number.Constant
        return obj

    @classmethod
    def class_factory(cls):
        obj = cls()
        obj.value = cls.Constant
        return obj

number_static = Number.static_factory()
number_class = Number.class_factory()
print(number_static.value, number_class.value)
```

class Method

- Classmethod 꾸밈자 사용
- 호출된 class인 cls를 받음 (self와 비슷)
- factory 패턴에서 사용

상속하면서 차이가 발생

02

Python Programming - Static & Class Method

```python
class Number:
    Constant = 10

    @staticmethod
    def static_factory():
        obj = Number()
        obj.value = Number.Constant
        return obj

    @classmethod
    def class_factory(cls):
        obj = cls()
        obj.value = cls.Constant
        return obj

class Number2(Number):
    Constant = 20
```

```python
number_static = Number2.static_factory()
number_class = Number2.class_factory()

print(number_static.value, number_class.value)
```

```
개인 및 시스템 프로필을 로드하는 데 1690ms가 걸렸습니다.
(base) PS C:\Users\GyuHyeon\Documents\랩 세미나\python programming>  & 'python' 'c:\Users\GyuHyeon\.
vscode\extensions\ms-python.python-2021.12.1559732655\pythonFiles\lib\python\debugpy\launcher' '1430
7' '--' 'c:\Users\GyuHyeon\Documents\랩 세미나\python programming\4.py'
10 20
```

02

Python Programming - Defaultdict

```
characters = {}

for char in text:
    count = characters.get(char, None)

    if count is None:
        characters[char] = 0

    characters[char] += 1
```

Dictionary의 기본 값을 지정 가능

```
from collections import defaultdict

characters = defaultdict(int) # 기본값 int()
# characters = defaultdict(lambda: 0)
for char in text:
    characters[char] += 1
```

파이썬 기본 dictionary는 키가 없을 경우 에러

에러 발생을 막기위해선 .get 메소드의 default 값 지정 필요

02

Python Programming - Named Tuple

```
class Coords3D:
    def __init__(self, x, y, z):
        self._x = x
        self._y = y
        self._z = z

    @property
    def x(self):
        return self._x

    @property
    def y(self):
        return self._y

    @property
    def z(self):
        return self._z
```

데이터만을 담기 위한 클래스 사용

- 클래스 선언 및 getter 작성 필요
- 숫자로 inedxing 불가능
    - _getitem_ 작성 필요

귀찮아 .....

02

Python Programming - Named Tuple

```
point = (10, 20, 30)

print(point[0]) # 뭐가 x였지…?
print(point[1])
```

- 그냥 튜플로도 관리는 가능 !
  - 쉽게 자료구조 생성 가능

- 관리하기 불편 ..
  - Attribute 명으로 접근이 불가능

이름이 없기 때문 !!

02

Python Programming - Named Tuple

```python
from collections import namedtuple

# 새로운 타입 생성
Coords3D = namedtuple("Coords3D", ['x', 'y', 'z'])

point = Coords3D(10, 20, z=30)
print(point.x)          # Attribute 이름으로 참조
Print(point[1])         # Index로 참조
print(*point)           # Tuple Unpacking

point[1] += 1           # Error 발생
```

```python
from collections import namedtuple
# 새로운 타입 생성
Coords3D = namedtuple("Coords3D", ['x', 'y', 'z'])

point = Coords3D(10, [20,15], z=30)

print(point.x)
print(point[1][1])
```

```
(base) PS C:\Users\GyuHyeon\Documents\랩 세미나\python programming>  c:; cd 'c:\Users\G
eon\Documents\랩 세미나\python programming'; & 'python' 'c:\Users\GyuHyeon\.vscode\exte
ns\ms-python.python-2022.0.1814523869\pythonFiles\lib\python\debugpy\launcher' '9163'
'c:\Users\GyuHyeon\Documents\랩 세미나\python programming\5.py'
10
15
```

- 각 튜플 원소에 이름 붙이기 가능
- 클래스가 아니라 튜플이다 !!
  - Unpacking 가능
- 튜플이기 때문에 수정이 불가능

02

Python Programming - Dataclass

```python
from dataclasses import dataclass

@dataclass
class Coords3D:
    x: float
    y: float
    z: float = 0

    def norm(self) -> float:
        return (self.x ** 2 + self.y ** 2 + self.z ** 2) ** .5

point = Coords3D(10, 20, z=30)
print(point)
print(point.x)
print(point[1])
print(point.norm())
```

Pythonic한 데이터 클래스를 위해선
dataclasses의 dataclass 데코레이터
활용

감사합니다!

20190431 박규현