Day17 Object Oriented Programming in Python

Recap

- Comprehensions
  - List
  - Dictionary
  - Set
  - Tuple
- Lambda
- pass
- map()
- filter()

Today Objectives

- Iterators
- Generators
- Object Oriented Programming

Iterators

list, string, tuple, set, dictionary

100 lines of code

string with 50 charac

50 lines I need 25 character

100 I need 25 char

In [1]:

```
s1 = """Python is an interpreted high-level general-purpose programming language. Pytho
Developer: Python Software Foundation
Stable release: 3.9.5 / 3 May 2021; 29 days ago
Preview release: 3.10.0b1 / 3 May 2021; 29 days ago
Typing discipline: Duck, dynamic, strong typing; gradual (since 3.5, but ignored in CPy
First appeared: February 1991; 30 years ago
Paradigm: Multi-paradigm: object-oriented, procedural (imperative), functional, structu
```

In [2]:

```
it = iter(s1)


print(it)
```

```
<str_iterator object at 0x000001EF79D16970>
```

In [3]:

```
next(it)
```

Out[3]:

```
'P'
```

In [4]:

```
print(next(it))
```

```
y
```

In [5]:

```
for i in range(10):
    print(next(it))
```

```
t
h
o
n

i
s

a
n
```

In [6]:

```
1 next(it)
```

Out[6]:

```
' '
```

In [7]:

```
1 next(it)
```

Out[7]:

```
'i'
```

In [9]:

```
1 it1 = iter('Python')
2 it2 = 'python'
```

In [10]:

```
1 for i in it1:
2     print(i)
```

```
P
y
t
h
o
n
```

In [12]:

```
1 for j in it2:
2     print(j)
```

```
p
y
t
h
o
n
```

In [14]:

```
1 print(next(it1))
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-14-a1664e0d0ec1> in <module>
----> 1 print(next(it1))

StopIteration:
```

In [15]:

```
for i in it1:
    print(i)
```

In [16]:

```
for i in it2:
    print(i)
```

```
p
y
t
h
o
n
```

In [17]:

```
li = [1, 2, 3, 4, 5, 6]

li2 = iter([1, 2, 3, 4, 5, 6])
```

In [18]:

```
print(li, li2)
```

```
[1, 2, 3, 4, 5, 6] <list_iterator object at 0x000001EF79D2FA30>
```

In [19]:

```
next(li2)
```

Out[19]:

```
1
```

In [20]:

```
print(next(li2))
```

```
2
```

In [21]:

```
next(li2)
```

Out[21]:

```
3
```

In [24]:

```
1 next(li2)
```

Out[24]:

```
6
```

In [25]:

```
1 next(li2)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-25-a51a8cea1203> in <module>
----> 1 next(li2)

StopIteration:
```

In [26]:

```
1 print(li)
```

```
[1, 2, 3, 4, 5, 6]
```

In [27]:

```
1 li2 = iter([1, 2, 3, 4, 5, 6])
```

In [28]:

```
1 for i in range(3):
2     print(next(li2))
```

```
1
2
3
```

In [29]:

```
1 for i in range(3):
2     print(next(li2))
```

```
4
5
6
```

In [30]:

```
for i in range(3):
    print(next(li2))
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-30-9c42caeb07b7> in <module>
      1 for i in range(3):
----> 2     print(next(li2))

StopIteration:
```

Generator

In [36]:

```
def gen():
    a = 10
    yield a

    a **= 5
    yield a

    a **= 10
    yield a
```

In [37]:

```
g = gen()
```

In [38]:

```
next(g)
```

Out[38]:

```
10
```

In [39]:

```
next(g)
```

Out[39]:

```
100000
```

In [40]:

```
1 next(g)
```

Out[40]:

```
100000000000000000000000000000000000000000000000000
```

In [41]:

```
1 next(g)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-41-e734f8aca5ac> in <module>
----> 1 next(g)

StopIteration:
```

In [42]:

```
1 t1 = (i for i in range(100))
```

In [43]:

```
1 print(t1)
```

```
<generator object <genexpr> at 0x000001EF79E14350>
```

In [44]:

```
1 next(t1)
```

Out[44]:

```
0
```

In [46]:

```
1 li = range(0, 100)
```

In [47]:

```
for i in li:
    print(i)
    if i == 10:
        break
```

```
0
1
2
3
4
5
6
7
8
9
10
```

In [48]:

```
for i in li:
    print(i)
    if i == 10:
        break
```

```
0
1
2
3
4
5
6
7
8
9
10
```

In [49]:

```
for i in t1:
    print(i)
    if i == 10:
        break
```

```
1
2
3
4
5
6
7
8
9
10
```

In [51]:

```
for i in t1:
    print(i)
    if i == 20:
        break
```

```
11
12
13
14
15
16
17
18
19
20
```

Object Oriented Programming In Python

- Attributes -> Variables created inside class
- Method -> Function created inside the class
- Class -> Blueprint for the object -> Design of a thing -> No memory
- Object -> Instance of the class -> It occupies memory

Structure

```
class class_name:
    """Doc String"""
    Attributes
    Methods
```

In [54]:

```
class Student:
    """This is Blueprint created for student Id card"""
```

In [56]:

```
class Student2:

```

```
  File "<ipython-input-56-29f97fd7958c>", line 2

    ^
SyntaxError: unexpected EOF while parsing
```

In [57]:

```
class Student2:
    pass
```

In [58]:

```
apssdc = Student()
```

In [59]:

```
print(apssdc)
```

```
<__main__.Student object at 0x000001EF79E04E20>
```

In [60]:

```
apssdc.__doc__
```

Out[60]:

```
'This is Blueprint created for student Id card'
```

In [61]:

```
Student.__doc__
```

Out[61]:

```
'This is Blueprint created for student Id card'
```

In [62]:

```
class Student:
    """This is Blueprint created for student Id card"""
    college = 'APSSDC'
    branch = 'Data Science'
    collegeAddress = 'Tadepalli, Guntur'
```

In [65]:

```
std1 = Student()
std2 = Student()
```

In [66]:

```
std1.college
```

Out[66]:

```
'APSSDC'
```

In [67]:

```
std2.college
```

Out[67]:

'APSSDC'

In [68]:

```
std1.collegeAddress
```

Out[68]:

'Tadepalli, Guntur'

In [74]:

```
class Student:
    """This is Blueprint created for student Id card"""
    college = 'APSSDC'
    branch = 'Data Science'
    collegeAddress = 'Tadepalli, Guntur'
    def __init__(self): # dendur init, constructor, instance method, initializer
        print("Student object is created")
```

In [75]:

```
std1 = Student()
std2 = Student()
```

```
Student object is created
Student object is created
```

In [76]:

```
std1.college
```

Out[76]:

'APSSDC'

In [77]:

```
class Student:
    """This is Blueprint created for student Id card"""
    college = 'APSSDC'
    branch = 'Data Science'
    collegeAddress = 'Tadepalli, Guntur'
    def __init__(self, name, rollNo, mobileNo): # dendur init, constructor, instance me
        self.name = name
        self.rollNo = rollNo
        self.mobileNo = mobileNo
        print("{} object is created".format(self.name))
```

In [78]:

```
std1 = Student('Python', '123456', '9876543210')
std2 = Student('Data', '789987', '1234567890')
```

```
Python object is created
Data object is created
```

In [79]:

```
std1.name
```

Out[79]:

```
'Python'
```

In [80]:

```
std2.name
```

Out[80]:

```
'Data'
```

In [81]:

```
std1.college, std2.college
```

Out[81]:

```
('APSSDC', 'APSSDC')
```

In [82]:

```
print(std1, std2)
```

```
<__main__.Student object at 0x000001EF7BCC1730> <__main__.Student object at
0x000001EF7BCC1760>
```

In [87]:

```
class Student:
    """This is Blueprint created for student Data"""
    college = 'APSSDC'
    branch = 'Data Science'
    collegeAddress = 'Tadepalli, Guntur'
    def __init__(self, name, rollNo, mobileNo, python, java): # dendur init, constructo
        self.name = name
        self.rollNo = rollNo
        self.mobileNo = mobileNo
        self.python = python
        self.java = java
        print("{} object is created".format(self.name))

    def average(self):
        avg = (self.python + self.java) / 2
        return avg

    def py_status(self):
        if self.python >= 40:
            print("Hurry you have passed python subject")
        else:
            print("Hurry you have failed python subject")

    def java_status(self):
        if self.java >= 40:
            print("Hurry you have passed Java subject")
        else:
            print("Hurry you have failed Java subject")
```

In [88]:

```
std1 = Student('Python', '123456', '9876543210', 85, 65)
std2 = Student('Data', '789987', '1234567890', 65, 85)
```

```
Python object is created
Data object is created
```

In [89]:

```
std1.average()
```

Out[89]:

```
75.0
```

In [90]:

```
std2.average()
```

Out[90]:

```
75.0
```

In [93]:

```
print(std1.python, std2.python)
```

```
85 65
```

In [94]:

```
std1.java_status()
```

```
Hurry you have passed Java subject
```

In [95]:

```
class Student:
    """This is Blueprint created for student Data"""
    college = 'APSSDC'
    branch = 'Data Science'
    collegeAddress = 'Tadepalli, Guntur'
    def __init__(self, name, rollNo, mobileNo, python, java): # dendur init, constructo
        self.name = name
        self.rollNo = rollNo
        self.mobileNo = mobileNo
        self.python = python
        self.java = java
        print("{} object is created".format(self.name))

    def average(self):
        avg = (self.python + self.java) / 2
        return avg

    def py_status(self, PyaddOn):
        self.python += PyaddOn
        if self.python >= 40:
            print("Hurry you have passed python subject")
        else:
            print("Hurry you have failed python subject")

    def java_status(self, JaddOn):
        self.java += JaddOn
        if self.java >= 40:
            print("Hurry you have passed Java subject")
        else:
            print("Hurry you have failed Java subject")
```

In [96]:

```
std1 = Student('Python', '123456', '9876543210', 35, 65)
std2 = Student('Data', '789987', '1234567890', 65, 25)
```

```
Python object is created
Data object is created
```

In [97]:

```
std1.py_status(5)
```

```
Hurry you have passed python subject
```

In [98]:

```
std1.python
```

Out[98]:

```
40
```

In [99]:

```
std1.pyaddOn
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-99-d1447455e913> in <module>
----> 1 std1.pyaddOn

AttributeError: 'Student' object has no attribute 'pyaddOn'
```

In [100]:

```
std2.py_status(0)
```

```
Hurry you have passed python subject
```

In [101]:

```
std1 = Student('Python', '123456', '9876543210', 35, 65)
std2 = Student('Data', '789987', '1234567890', 65, 25)
```

```
Python object is created
Data object is created
```

In [102]:

```
std2.name = 'Data Science'
```

In [103]:

```
std2.name
```

Out[103]:

```
'Data Science'
```

In [104]:

```
print(std1, std2)
```

```
<__main__.Student object at 0x000001EF7BCBC4F0> <__main__.Student object at
0x000001EF7BCBCFD0>
```

In [108]:

```
class Student:
    """This is Blueprint created for student Data"""
    college = 'APSSDC'
    branch = 'Data Science'
    collegeAddress = 'Tadepalli, Guntur'
    def __init__(self, name, rollNo, mobileNo, python, java): # dendur init, constructo
        self.name = name
        self.rollNo = rollNo
        self.mobileNo = mobileNo
        self.python = python
        self.java = java
        print("{} object is created".format(self.name))

    def average(self):
        avg = (self.python + self.java) / 2
        return avg

    def py_status(self, PyaddOn):
        self.python += PyaddOn
        if self.python >= 40:
            print("Hurry you have passed python subject")
        else:
            print("Hurry you have failed python subject")

    def java_status(self, JaddOn):
        self.java += JaddOn
        if self.java >= 40:
            print("Hurry you have passed Java subject")
        else:
            print("Hurry you have failed Java subject")
    def __str__(self):
        return "This class belongs to {}".format(self.name)
```

In [109]:

```
std1 = Student('Python', '123456', '9876543210', 35, 65)
std2 = Student('Data', '789987', '1234567890', 65, 25)
```

```
Python object is created
Data object is created
```

In [110]:

```
print(std1, std2)
```

```
This class belongs to Python This class belongs to Data
```

In [119]:

```
class Student:
    """This is Blueprint created for student Data"""
    college = 'APSSDC'
    branch = 'Data Science'
    collegeAddress = 'Tadepalli, Guntur'
    def __init__(self, name, rollNo, mobileNo, python, java): # dendur init, constructo
        self.name = name
        self.rollNo = rollNo
        self.mobileNo = mobileNo
        self.python = python
        self.java = java
        print("{} object is created".format(self.name))

    def average(self):
        avg = (self.python + self.java) / 2
        return avg

    def py_status(self, PyaddOn):
        self.python += PyaddOn
        if self.python >= 40:
            print("Hurry you have passed python subject")
        else:
            print("Hurry you have failed python subject")

    def java_status(self, JaddOn):
        self.java += JaddOn
        if self.java >= 40:
            print("Hurry you have passed Java subject")
        else:
            print("Hurry you have failed Java subject")
    def __str__(self):
        return "This class belongs to {}".format(self.name)
    def __del__(self): # destructor
        print("{} data has deleted".format(self.name))
```

In [121]:

```
std1 = Student('data', '123456', '9876543210', 35, 65)
std2 = Student('python', '789987', '1234567890', 65, 25)
```

```
data object is created
Python data has deleted
python object is created
Data data has deleted
```

In [114]:

```
std1
```

Out[114]:

```
5
```

Types of methods

- Instance Method

- Class method
- Static Method

In [122]:

```
class Student:
    """This is Blueprint created for student Data"""
    college = 'APSSDC'
    branch = 'Data Science'
    collegeAddress = 'Tadepalli, Guntur'
    def __init__(self, name, rollNo, mobileNo, python, java): # dendur init, constructo
        self.name = name
        self.rollNo = rollNo
        self.mobileNo = mobileNo
        self.python = python
        self.java = java
        print("{} object is created".format(self.name))

    def average(self):
        avg = (self.python + self.java) / 2
        return avg

    def py_status(self, PyaddOn):
        self.python += PyaddOn
        if self.python >= 40:
            print("Hurry you have passed python subject")
        else:
            print("Hurry you have failed python subject")

    def java_status(self, JaddOn):
        self.java += JaddOn
        if self.java >= 40:
            print("Hurry you have passed Java subject")
        else:
            print("Hurry you have failed Java subject")
    def __str__(self):
        return "This class belongs to {}".format(self.name)
    def __del__(self): # destructor
        print("{} data has deleted".format(self.name))

    @staticmethod # Decorator
    def clgNo():
        return 1234567890
```

In [123]:

```
Student.clgNo()
```

Out[123]:

```
1234567890
```

In [124]:

```
class Student:
    """This is Blueprint created for student Data"""
    college = 'APSSDC'
    branch = 'Data Science'
    collegeAddress = 'Tadepalli, Guntur'
    def __init__(self, name, rollNo, mobileNo, python, java): # dendur init, constructo
        self.name = name
        self.rollNo = rollNo
        self.mobileNo = mobileNo
        self.python = python
        self.java = java
        print("{} object is created".format(self.name))

    def average(self):
        avg = (self.python + self.java) / 2
        return avg

    def py_status(self, PyaddOn):
        self.python += PyaddOn
        if self.python >= 40:
            print("Hurry you have passed python subject")
        else:
            print("Hurry you have failed python subject")

    def java_status(self, JaddOn):
        self.java += JaddOn
        if self.java >= 40:
            print("Hurry you have passed Java subject")
        else:
            print("Hurry you have failed Java subject")
    def __str__(self):
        return "This class belongs to {}".format(self.name)
    def __del__(self): # destructor
        print("{} data has deleted".format(self.name))

    @staticmethod # Decorator
    def clgNo():
        return 1234567890

    @classmethod
    def ClgName(cls):
        return cls.college
```

In [126]:

```
Student.ClgName()
```

Out[126]:

```
'APSSDC'
```

Oops Concepts

- Encapsulation
- Polymorphism
- Inheritance

    - Types
        - Multi Level
        - Hybrid
        - Multiple
        - Hierarchy
    - Operator Overloading
    - Method overwriting
- Abstraction

Today Outcomes

- Iterator
- Generator
- OOP
    - Class
    - Object
    - Method
        - Instance
        - Static
        - Class
    - Attribute
        - Instance
        - Class