5.2 - Classes

Data Science Practicum 2021/22, Lesson 5.2

Marko Tkalčič

Univerza na Primorskem

Table of Contents

Classes

Classes Exercises

Assignment

Classes

- You can create your own classes in Python

```
class MyClass:
  x = 5
```

Classes

- You can create your own classes in Python

```
class MyClass:
  x = 5
```

```
p1 = MyClass()
print(p1.x)
```

The __init__() Function

- All classes have a function called `__init__()`, which is always executed when the class is being initiated.

```
class Person:

    def __init__(self, name, age):
        self.name = name
        self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

```
John
36
```

- what is self?

The __init__() Function

- the self variable represents the instance of the object itself. Most object-oriented languages pass this as a hidden parameter to the methods defined on an object; Python does not. You have to declare it explicitly.

```python
class A(object):
    def __init__(self):
        self.x = 'Hello'

    def method_a(self, foo):
        print(self.x + ' ' + foo)
```

- When you create an instance of the A class and call its methods, it will be passed automatically,

```python
a = A()               # We do not pass any argument to the __init__ method
a.method_a('Sailor!') # We only pass a single argument
```

Methods

```python
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

Inheritance

- a class that inherits properties of another class
    - a Student is a kind of Person

```
public class Student extends Person
```

```
class Student(Person):
  ...
```

Inheritance

- a class that inherits properties of another class
  - a Student is a kind of Person

```
public class Student extends Person
```

```
class Student(Person):
  ...
```

- a method can be overridden

```
class Parent:       # define parent class
   def myMethod(self):
      print('Calling parent method')

class Child(Parent): # define child class
   def myMethod(self):
      print('Calling child method')

c = Child()          # instance of child
c.myMethod()         # child calls overridden method
```

Overriding `__init__()`

- The child's `__init__()` function overrides the inheritance of the parent's `__init__()` function.

```
class Student(Person):
  def __init__(self, fname, lname):
    Person.__init__(self, fname, lname)
```

Naming Conventions

- Class names should be in Upper Case (Camel Case)
- Built-in classes, on the other hand, are usually in lowercase
- Method names should be in lowercase
- Multi-word method names should be separated by underscore
- Example:
  - Classes: MyNewClass, Account, Person, NewCar
  - Methods: get_data(), print_price(), add_name(), turn_on_engine()

Table of Contents

Classes

Classes Exercises

Assignment

Exercise

- write a class
  - two variables : x,y
  - init method that instantiates x and y using parameters
- instantiate
- print the values of the x and y variables

Exercise

- write a class
  - two variables : x,y
  - init method that instantiates x and y using parameters
- instantiate
- print the values of the x and y variables

```
class FourthClass:
    x=1
    y=1
    def __init__(self,x,y):
        self.x = x
        self.y = y
a = FourthClass(5,6)
print(a.x, a.y)
```

Exercise

- Write a Python class named Rectangle constructed by a length and width and a method which will compute the area of a rectangle.

Exercise

- Write a Python class named Rectangle constructed by a length and width and a method which will compute the area of a rectangle.

```python
class Rectangle():
    def __init__(self, l, w):
        self.length = l
        self.width  = w

    def rectangle_area(self):
        return self.length*self.width

newRectangle = Rectangle(12, 10)
print(newRectangle.rectangle_area())
```

Exercise

- write a class that instantiates a 2D list with the init method

```
a = TwoDList(5,6)
```

- populate the list with random integer values from 1 to 20

```
import random
random.randint(1,20)

list_2d = [[foo for i in range(m)] for j in range(n)]
```

- write a method that, given the parameters a and b counts and returns:
    - less than a
    - between a and b
    - more than b
- print the return values

Exercise

```python
import random

class TwoDList:
    rows = 0
    cols = 0
    list_2d = []

    def __init__(self,c,r):
        self.list_2d = [[random.randint(1,20) for i in range(r)] for j in range(c)]
        self.rows = r
        self.cols = c

    def count(self,a,b):
        hi=0
        mid=0
        lo=0

        for i in range(self.rows):
            for j in range(self.cols):
                if self.list_2d[j][i] <a:
                    lo+=1
                elif self.list_2d[j][i] <b:
                    mid+=1
                else:
                    hi+=1
        return [lo,mid,hi]

a = TwoDList(5,6)
print(a.count(5,10))
```

Table of Contents

Classes

Classes Exercises

Assignment

Assignment

Create a Bus child class that inherits from the Vehicle class. The default fare charge of any vehicle is seating capacity * 100. If Vehicle is Bus instance, we need to add an extra 10% on full fare as a maintenance charge. So total fare for bus instance will become the final amount = total fare + 10% of the total fare.

Note: The bus seating capacity is 50. so the final fare amount should be 5550. You need to override the fare() method of a Vehicle class in Bus class.

Use the following code for your parent Vehicle class. We need to access the parent class from inside a method of a child class.

```python
class Vehicle:
    def __init__(self, name, mileage, capacity):
        self.name = name
        self.mileage = mileage
        self.capacity = capacity

    def fare(self):
        return self.capacity * 100

class Bus(Vehicle):
    pass

School_bus = Bus("School Volvo", 12, 50)
print("Total Bus fare is:", School_bus.fare())
```

References

Part of the material has been taken from the following sources. The usage of the referenced copyrighted work is in line with fair use since it is for nonprofit educational purposes.

- https://realpython.com/python-namespaces-scope/
- https://www.tutorialspoint.com/python/python_functions.htm
- https://www.w3schools.com/python/python_functions.asp
- https://pynative.com/python-functions-exercise-with-solutions/
- https://www.w3schools.com/python/python_classes.asp
- https://stackoverflow.com/questions/625083/what-init-and-self-do-on-python