ÖZYEĞİN UNIVERSITY

EE393 Python for Engineers

Dr. Orhan Gökçöl

orhan.gokcol@ozyegin.edu.tr

online

16.11.2020

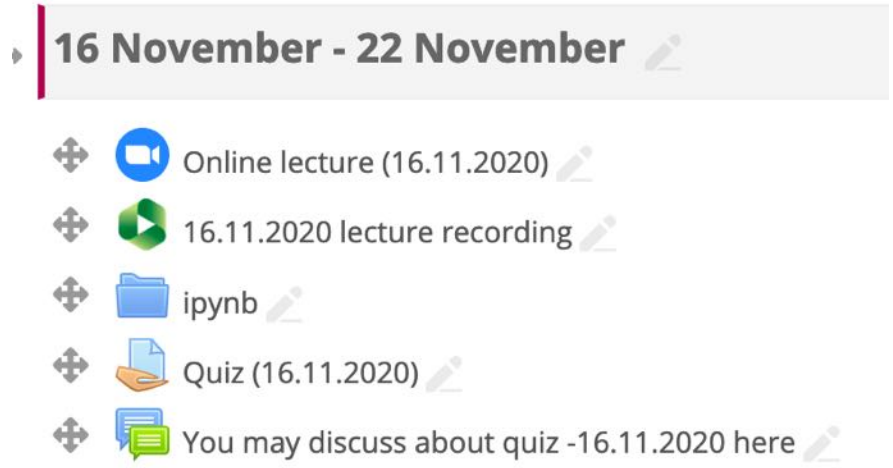
2020-2021 Fall Semester

Agenda

- Short review
- Working with Excel files
  - openpyxl
- Objects in Python
- OOP basics
- OOP examples

(Download to your local)

16.11.2020 | LMS resources



Learning objectives for 16.11.2020

- Working with Excel files using **openpyxl**
- Understand “object” concept in python
- Learn OOP basics to create own object prototypes – classes
- Working with our own libraries

MIDTERM EXAM

SCHEDULED ON DECEMBER 21, 2020; 08:40

Online
(Details will be announced later)

5

Schocking news:

Guido, who is the founder of Python, started working for Microsoft!!!!!!

6

Working with Excel in Python!!!!!!!!!

We'll use **openpyxl** module

(Good tutorial!)

SEE
excel.ipynb
&
demo.xlsx

Terminology :

A column is a vertical line. It is represented by uppercase letters

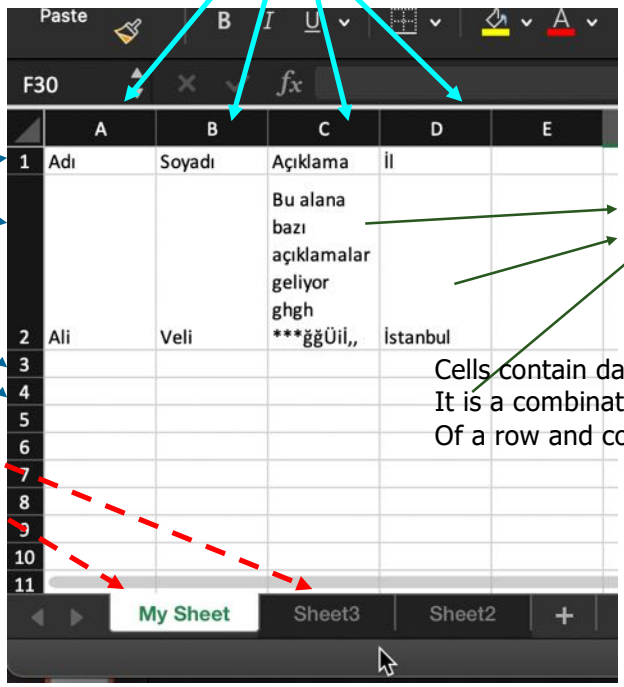
Column

A row is a horizontal line. It is represented by a number

Row:

Sheet or Worksheet:

Cell:



Cells contain data. It is a combination Of a row and column

Book or Workbook : Contains one or more sheets

```
import openpyxl;
wb = openpyxl.load_workbook(filename = 'sample.xlsx', data_only = True)

mySheets = wb.sheetnames #get sheet names
print (mySheets) #print sheet names
```

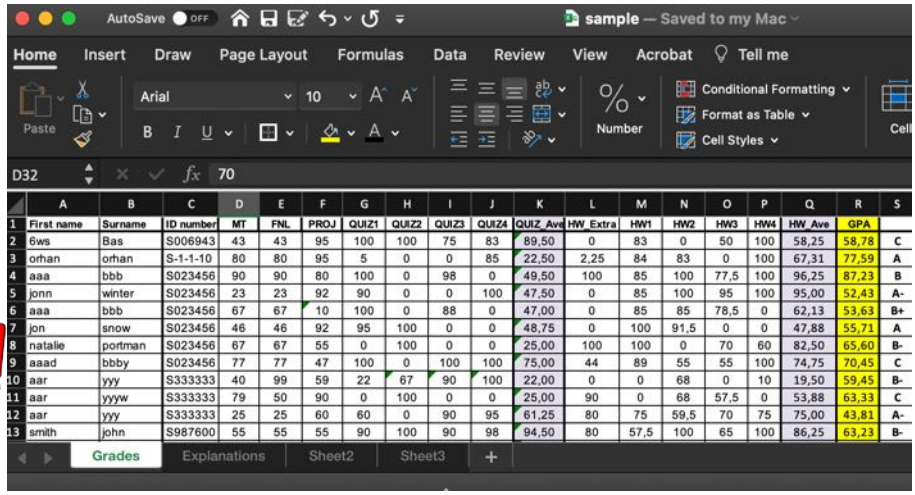
(if you only need values)

```
['Grades', 'Explanations', 'Sheet2', 'Sheet3']
```

Sheet names are stored in a list!!!

excel.ipynb

sample.xlsx



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	First name	Surname	ID number	MT	FNL	PROJ	QUIZ1	QUIZ2	QUIZ3	QUIZ4	QUIZ_Ave	HW_Extra	HW1	HW2	HW3	HW4	HW_Ave	GPA	
2	6ws	Bas	S006943	43	43	95	100	100	75	83	89,50	0	83	0	50	100	58,25	58,78	C
3	orhan	orhan	S-1-1-10	80	80	95	5	0	0	85	22,50	2,25	84	83	0	100	67,31	77,59	A
4	aaa	bbb	S023456	90	90	80	100	0	98	0	49,50	100	85	100	77,5	100	96,25	87,23	B
5	jonn	winter	S023456	23	23	92	90	0	0	100	47,50	0	85	100	95	100	95,00	52,43	A-
6	aaa	bbb	S023456	67	67	10	100	0	88	0	47,00	0	85	85	78,5	0	62,13	53,63	B+
7	jon	snow	S023456	46	46	92	95	100	0	0	48,75	0	100	91,5	0	0	47,88	55,71	A
8	natalie	portman	S023456	67	67	55	0	100	0	0	25,00	100	100	0	70	60	82,50	65,60	B-
9	aaad	bbby	S023456	77	77	47	100	0	100	100	75,00	44	89	55	55	100	74,75	70,45	C
10	aar	yyy	S333333	40	99	59	22	67	90	100	22,00	0	0	68	0	10	19,50	59,45	B-
11	aar	yyyw	S333333	79	50	90	0	100	0	0	25,00	90	0	68	57,5	0	53,88	63,33	C
12	aar	yyy	S333333	25	25	60	60	0	90	95	61,25	80	75	59,5	70	75	75,00	43,81	A-
13	smith	john	S987600	55	55	55	90	100	90	98	94,50	80	57,5	100	65	100	86,25	63,23	B-

(Data shown in sample.xlsx is for demonstration only. It does not contain any real grade!!!!!!)

P.S. For windows, use the following path syntax: "**C:\\Users\\Admin\\Desktop\\demo.xlsx**"

Last week

- Iterables, Referencing
- Generators
- Global variables
- Important utility libs : random, sys, os, datetime
- Special methods in python (starts with double underscore; __)
  - __name__, __main__
- **eval** and **exec** functions
- Python libs for science and engineering : numpy, scipy, matplotlib etc. – an introduction

Our experience with the objects so far…

- Till now, we have been using objects all along the past two months. Python provides us with many built-in objects. Here is some simple code where the first few lines should feel very simple and natural to all of us.

```
[5]: mylist = list()
     mylist.append('python')
     mylist.append('OzU')
     mylist.append('EE393')
     mylist.sort()
     print (mylist[0])
     print (mylist.__getitem__(0))
     print (list.__getitem__(mylist,0))
```

```
EE393
EE393
EE393
```

oop.ipynb

- Every list we create has data
- There are methods we can use on the instances of lists that we created : append, sort, count, ….
- We use methods using "dot notation"

OOP –Object oriented programming

- In all the programs we developed so far, we have designed our program around **functions** i.e. blocks of statements which manipulate data. This is called the *procedural programming*. It helps us better organizing and maintaining programs
- There is another way of organizing our program which is **to combine data and functionality and wrap it inside** something called an **object**.
- This is called the ***object oriented*** programming paradigm. Most of the time we can use procedural programming, but when writing large programs or have a problem that is better suited to this method, you can use object oriented programming techniques.

oop.ipynb

Object Oriented

A program is made up of many **cooperating objects**

Instead of being the "whole program"

- each object is a little "island" within the program and

cooperatively working with other objects

A program is made up of one or more objects working together
- objects make use of each other's capabilities

In **Python**, almost everything is an object. That is, everything we created as a "type" will have data attached to it and will have functions attached to it, aswell.

Typical example is "list". It is similar for tuple, dict, set, int, float, str …
All of these types can be used to create instances of then. Then we use many built-in functions (methods) of these objects to do different tasks.

The Object of Objects

- Basic idea – view a complex system as the interaction of simpler *objects*. An object is a sort of active data type that combines data and operations.
- Objects *know stuff* (contain data) and they can *do stuff* (have operations).
- Objects interact by sending each other *messages*.

Example: Suppose we want to develop a data processing system for OzU.

- We must keep records on students who attend the school. Each student will be represented as an object.

The Object of Objects

- The **student** object would contain data like:
  - Name
  - Surname
  - Department
  - ID number
  - Courses taken
  - E-mail address
  - Home Address
  - GPA
  - etc.

The Object of Objects

- The student object should also respond to requests.
- We may want to send out a campus-wide mailing, so we'd need a campus address for each student.
- We could send the **`printEmailAddress`** to each student object. When the student object receives the message, it prints its own address.

- Objects may refer to other objects.
- Each course might be represented by an object:
  - Instructor
  - Course syllabus
  - Prerequisite courses
  - When and where the class meets

- Sample Operation
  - `addStudent`
  - `delStudent`
  - `changeRoom`
  - …

Object

An Object is a bit of self-contained Code (i.e. methods or functions) and Data

A key aspect of the Object approach is to break the problem into smaller understandable parts (divide and conquer)

Objects have boundaries that allow us to ignore un-needed detail

As we discussed previously, we have been using objects all along: String Objects, Integer Objects, Dictionary Objects, List Objects...

See example built-in objects in **oop.ipynb**

Definitions

- Class - a template to create objects
- Method or Message - A defined capability of a class
- Field or attribute- A bit of data in a class
- Object or Instance – A working copy of a template – something which is created from a template

Terminology: Class

Defines the abstract characteristics of a thing (object), including the thing's characteristics (its attributes, fields or properties) and the thing's behaviors (the things it can do, or methods, operations or features). One might say that a class is a blueprint or factory that describes the nature of something.
For example, the class Dog would consist of traits shared by all dogs, such as breed and fur color (**characteristics**), and the ability to bark and sit (**behaviors**).

http://en.wikipedia.org/wiki/Object-oriented_programming

Terminology: Instance

One can have an instance of a class or a particular object. The instance is the actual object created at runtime. In programmer jargon, the Lassie object is an instance of the Dog class. The set of values of the attributes of a particular object is called its state.

The object consists of state and the behavior that's defined in the object's class.

Object and Instance are often used interchangeably.

http://en.wikipedia.org/wiki/Object-oriented_programming

Terminology: Method

An object's abilities. In language, methods are verbs. Lassie, being a Dog, has the ability to bark. So bark() is one of Lassie's methods. She may have other methods as well, for example sit() or eat() or walk() or save_timmy().

Within the program, using a method usually affects only one particular object; all Dogs can bark, but you need only one particular dog to do the barking

Method and Message are often used interchangeably.

http://en.wikipedia.org/wiki/Object-oriented_programming

Classes and objects are the two main aspects of object oriented programming:

A **class** creates a new *type* where **objects** are **instances** of the class.
The simplest class possible is shown in the following example.

```
class Person:
    def say_hi(self):
        print('Hello, how are you?')

p = Person()
p.say_hi()
```

An instance of the object is CREATED!

oop.ipynb

self keyword is used for the class method. Class methods have only one specific difference from ordinary functions - they must have an extra first name that has to be added to the beginning of the parameter list, but you **do not** give a value for this parameter when you call the method, Python will provide it. This particular variable refers to the object *itself*, and by convention, it is given the name **self**
It is like "this" pointer in C++ or "this" reference in Java.

__init__ method

This method is run as soon as an object of a class is instantiated (i.e. created). The method is useful to do any *initialization (it is like C++* ***constructor****)*

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

    def say_hi(self):
        print('Hello, my name is', self.name)
p = Person('Ali Veli')
p.say_hi()
```


self

It is like the pointer "this" in C++
In Python, functions in a class access data via **self**

```
#simple class v2
class Person:
    def __init__(self, tmp=""):
        self.name = tmp

    def say_hi(self):
        print('Hello {0:s}, how are you?'.format(self.name))

p = Person()
p.say_hi()
p2 = Person ("Jon Snow")
p2.say_hi()
```

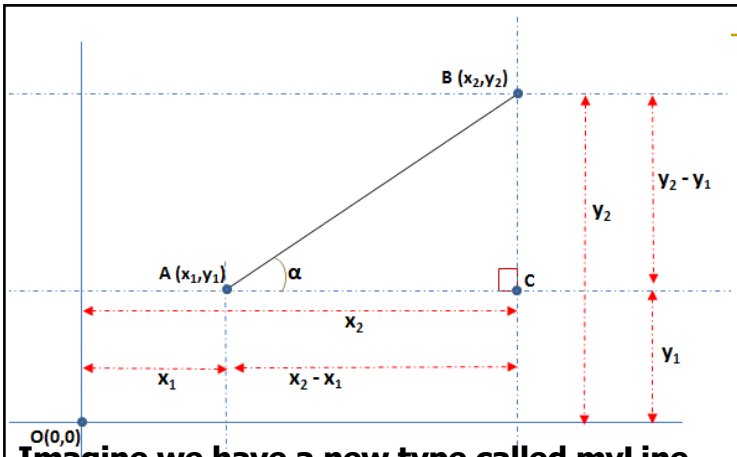
```
Hello , how are you?
Hello Jon Snow, how are you?
```

- What's the name of the class?
- What are the data stored in each instances of the object?
- What are the meythods?

oop.ipynb

self acts like a variable, but it won't call any data
If we print "self" in a class function, it will print the reference of the object created!!!

Case study: line class



A line can be defined by giving two coordinates, namely (x1,y1) and (x2,y2)

DATA: x1,y1,x2,y2

If we have a line, we could want to know:

- its length
- İts new position when we move it by a certain amount
- its slope
- <etc.>

These are class methods

Imagine we have a new type called myLine

```
line1 = myLine() #default line (0,0)-(1,1)
line2 = myLine(2.7,-0.5,6.25,3) #(1.7,-0.5)-(6.25,3)
print ("length of line1 :", line1.length())
print ("length of line2 :", line2.length())
line1.moveX(1,3) #(x1=x1+1 and y1=y1+3)
print ("length of line1 :", line1.length())
print ("slope of line 2 :", line2.slope())
```

```
length of line1 : 1.4142135623730951
length of line2 : 4.985228179331413
length of line1 : 3.1622776601683795
slope of line 2 0.9859154929577465
```

oop.ipynb


```
class myLine:
#constructor
    def __init__(self,a=0,b=0,c=1,d=1):
        self.x1=a; self.y1=b;
        self.x2=c; self.y2=d

    def length(self): #returns length
        import math
        dx = self.x1-self.x2
        dy = self.y1-self.y2
        ll = math.sqrt(dx*dx + dy*dy)
        return ll

    def moveX(self, dx1=0,dx2=0):
        self.x1 += dx1
        self.x2 += dx2

    def moveY(self, dy1=0,dy2=0):
        self.y1 += dy1
        self.y2 += dy2

    def slope(self):
        return (self.y2-self.y1)/(self.x2-self.x1)
```

Constructor method is called whenever we create a new **myLine** object. X1,x2,y1 and y2 are our data which defines "line"

Length is an ordinary Python function. It is used as a method from an object (i.e. outside of the class) such as **x.length().** It returns a value to the caller

These are other methods we define in the class. They are used in the same way as «length». Note that, we put «self» all the time. A method can invoke (call) several values as normal functions in Python do

oop.ipynb

Converting classes to library

line.py

```
class myLine:
#constructor
    def __init__(self,a=0,b=0,c=1,d=1):
        self.x1=a; self.y1=b
        self.x2=c; self.y2=d

    def length(self): #returns length
        import math
        dx = self.x1-self.x2
        dy = self.y1-self.y2
        ll = math.sqrt(dx*dx + dy*dy)
        return ll

    def moveX(self, dx1=0,dx2=0):
        self.x1 += dx1
        self.x2 += dx2

    def moveY(self, dy1=0,dy2=0):
        self.y1 += dy1
        self.y2 += dy2

    def slope(self):
        return (self.y2-self.y1)/(self.x2-self.x1)
```

```
import line
l1 = line.myLine(1,2,-2,4)
print ("length of line :", l1.length())
l1.moveX(1,3) #(x1=x1+1 and x2=x2+3)
l1.moveY(0,-2) #(y2=y2-2)
print ("length of line :", l1.length())
```

```
length of line : 3.605551275463989
length of line : 1.0
```

```
import line
print (dir(line.myLine))
```

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format
__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subcl
ass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__re
duce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__
', '__weakref__', 'length', 'moveX', 'moveY', 'slope']
```


How can we improve this method?

- (Quiz)
- Take **myLine** as a starting point and extend it by adding new methods (at least five more) and new data (such as line color, thickness etc. –at least one data must be added) and use them in a driver program.
  - Some ideas: slope angle in degrees, rotate by certain amount around (0,0), make mirror of it, add two lines to each other to form a new line, scale the line by a factor, get the equation of the line in the form of y=mx+n, determine the intersection point with the x axis, determine intersection point of two lines,… and possibly more!!!!

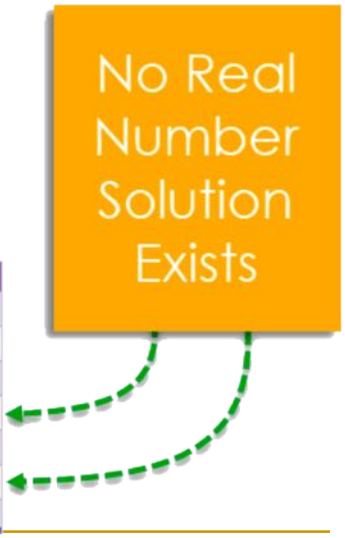
Case study: quadEqn class (simple)

- 2nd order quad eqn and its solution is given as follows:

$$ax^2 + bx + c = 0$$

$$\Rightarrow \quad x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

A	B	C	Quadratic Equation	Result 1	Result 2
1	-3	2	$x^2 + -3x + 2 = 0$	2	1
1	5	6	$x^2 + 5x + 6 = 0$	-2	-3
3	-6	3	$3x^2 + -6x + 3 = 0$	1	1
2	-4	7	$2x^2 + -4x + 7 = 0$	#NUM!	#NUM!
-3	-24	-48	$-3x^2 + -24x + -48 = 0$	-4	-4
1	2	3	$x^2 + 2x + 3 = 0$	#NUM!	#NUM!
1	2	0	$x^2 + 2x + 0 = 0$	0	-2



Class simpleEqn

simpleEqn

DATA: **a,b,c**

Methods

solve

```
myEquation = simpleEqn(1,4,2)
print ("Coeffs:", myEquation.a, myEquation.b, myEquation.c)
print ("Solution :",myEquation.solve())

myEquation2 = simpleEqn(-1,4,-6)
print ("Coeffs:", myEquation2.a, myEquation2.b, myEquation2.c)
print ("Solution :",myEquation2.solve())
```

```
Coeffs: 1 4 2
Solution : [-0.5857864376269049, -3.414213562373095]
Coeffs: -1 4 -6
Solution : ['no real root']
```

```
class simpleEqn:
#solver
    def solve(self):
        import math
        for x in [self.a, self.b, self.c]:
            if (type(x)!=int) and (type(x)!=float):
                return ["invalid input"]
        if self.a==0:
            return ["it's not a 2nd order eqn"]
        roots = [0,0]
        d = self.b**2 - 4*self.a*self.c
        if (d<0):
            roots = ["no real root"]
        else:
            roots[0] = (-self.b + math.sqrt(d))/(2*self.a)
            roots[1] = (-self.b - math.sqrt(d))/(2*self.a)
        return roots

#default constructor
    def __init__(self, aa=1, bb=1, cc=1):
        self.a = aa
        self.b = bb
        self.c = cc
```

31

Example : QuadEquation class (Advanced)

Object : QuadEqn
It holds a second order equation

Data:
- Coefficients of the equation
- Name of the equation

class QuadEqn contains methods suitable for working with 2nd order equations given as ax^2+bx+c=0
The methods are:

- delta() : returns b^2-4ac
- sumOfRoots() : returns -b/a
- multOfRoots() : returns c/a
- solve () : solves the equation and returns a list
- printEquation () : prints equation in a fancy format
- _ *add* _ : adds two equation to each other by overloading "+"
- change() : changes the coefficients of the equation

Usage :
eqn1 = QuadEqn(-3,-9,5, "my equation1") is used to define -3x^2-9x+5=0
Then, eqn1.solve() returns a list which contains solution!

32

```
eqn1 = QuadEqn(-3,-9,5, name="Equation 1")
eqn2 = QuadEqn(2,2,3, "Eqn 2")
eqn1.printEquation()
solution=eqn1.solve()
print (solution)
print ("delta is", eqn1.delta())
print (eqn1.sumOfRoots())
print (eqn1.multOfRoots())
eqn2.printEquation()
eqn3 = eqn1 + eqn2 # + is overloaded
eqn3.printEquation()
print ("Number of Equations:",QuadEqn.count)
eqn3.change(2,2,2,"new")
eqn3.printEquation()
```

Object Lifecycle

Objects are created, used, and discarded

We have special blocks of code (methods) that get called

- At the moment of creation (constructor)
- At the moment of destruction (destructor)

Constructors are used a lot

Destructors are seldom used

Access Modifiers

- In Python, there is no keywords like ‘public’, ‘protected’ and ‘private’ to define the accessibility.
- In other words, In Python, it acquiesce that all attributes are public.

- But there is a method in Python to define Private:
- Add “__” in front of the variable and methodn name can hide them when accessing them from out of class.

Access Modifiers


```
class Person:
    def __init__(self):

        self.A = 'Ali Veli'
        self.__B = 'Natalie Portman'

    def PrintName(self):
        print (self.A)
        print self.__B

P = Person()

>>> P.A
'Ali Veli'
>>> P.__B
Traceback (most recent call last):
  File "<pyshell#61>", line 1, in <module>
    P.__B
AttributeError: Person instance has no attribute '__B'
>>> P.PrintName()
Ali Veli
Natalie Portman
```

Public variable

Private variable

Invoke private variable in class

Access public variable out of class, succeed

Access private variable our of class, fail

Access public function but this function access Private variable __B successfully since they are in the same class.

Accessibility

- Actually, the private accessibility method is just a rule, not the limitation of compiler.
- Its fact is to change name of private name like __variable or __function() to **_ClassName__**variable or **_ClassName__function()**.
  So we can't Access them because of wrong names.
- We even can use the special syntax to access the private data or methods. The syntax is actually its changed name.

Accessibility


```
#Accessibility
class C:
    def accessible(self):
        print ("you can see me")

    def __inaccessible(self):
        print ("you can NOT see me")

var1 = C()
var1.accessible()
var1.__inaccessible()
```

Define public function

Define private function

Access public function

Can't access private function

```
you can see me
---------------------------------------------------------
AttributeError                            Traceback (m
<ipython-input-166-16cec1890434> in <module>
      9 var1 = C()
     10 var1.accessible()
---> 11 var1.__inaccessible()

AttributeError: 'C' object has no attribute '__inacces
```

```
var1._C__inaccessible()
```

Access private function via changed name

```
you can NOT see me
```

Static Methods

To declare a static method, you have to specify the `@staticmethod` descriptor before the name of the method as shown below:

```
class Car:

    @staticmethod
    def get_class_details():
        print ("This is a car class")

Car.get_class_details()
```

In the above script, we create a class `Car` with one static method `get_class_details()`. Let's call this method using the class name.

```
Car.get_class_details()
```

You can see that we did not need to create an instance of the `Car` class in order to call the `get_class_details()` method, rather we simply used the class name. It is important to mention that static methods can only access *class* attributes in Python.

In summary:

- Class - a template
- Attribute – A variable within a class
- Method - A function within a class
- Object - A particular instance of a class
- Constructor – Code that runs when an object is created

Object Oriented programming is a very structured approach to code reuse

We can group data and functionality together and create many independent instances of a class

SEE YOU NEXT WEEK!!!

DR. ORHAN GÖKÇÖL
gokcol@gmail.com
orhan.gokcol@ozyegin.edu.tr