Python Programming
Lesson 8
Classes and Objects

Presented by Advaspire Team

What is Classes and Objects?

Class and object is powerful tool in Python that can make you code more organized and powerful.
In Python, we are dealing with different types of data and not all data can be represent using the essential data that we normally use like numbers, strings and Boolean.
With class and object, we can kind of create our own data type.

With class, you can define your own data type and it is very useful and its used in most major programming language.

For example, I want to create a data type for my phone, I can use the class to do so.

Create a Class

In this lesson we are going to use multiple file so make sure to have a main file and another for the class

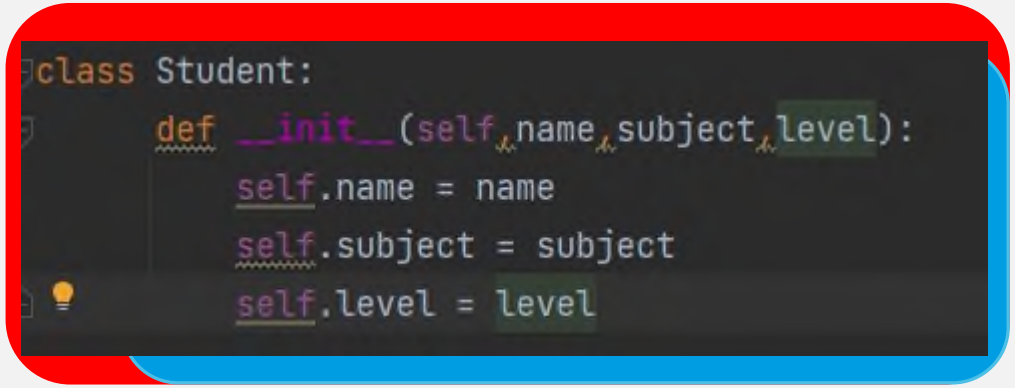
So this time, I want to create an Advaspire student data type. First just type in class Student: . Once you press enter, it should be indented. Starting from there, it will be part of the class. We can define a bunch of attributes about the student. You can use strings,integer and Boolean on what should a student be and have.

Next, we need to define a initialize function. Type it like in the example here. The init function helps us mapping out what attributes a student should have. After the self in the bracket, we need to insert what value should be a part of a student.

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

Insert a Value

A student should have some these traits like name, class or subject that they take, and also what level they are at. So all the stuff mentioned will be what defines a student.
The next lines of code will be a bit out of place right now, but it will be used with the object tool later.

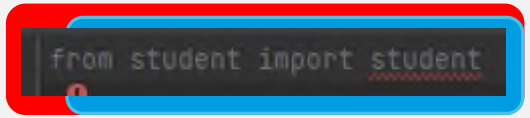


```
class Student:
    def __init__(self,name,subject,level):
        self.name = name
        self.subject = subject
        self.level = level
```

Creating a Student

Now open back the main file. We are going to create an actual student using the object. An object here will be the student who have a name, subject and level. The class is just an overview of what attributes should a student have.

We need to import the student data first. Based on the code here, from the student file, we need to import the student class. Both student is referring to different thing.
From = file
Import = function



To create an object, it is almost the same like variable, You need to have a name for it. Then the student class with the bracket. Inside the bracket will be the attributes, so type it in order.

```
from student import Student

student1 = Student("Zack",'Python','Level 11')

print (student1.name)
```

```
from student import student

student1 = student("Zack",'Python','Level 11')
student2 = student("William",'Python','Level 2')

print (student1.name)
```

Once you create a student, you can print the attributes inside by adding the dot after the object name. So if you want to print the level, just type level, name and so on. Now you can create multiple object using the same class.

How does this work?

When we were creating the student class, remember that we have some attributes. Those attributes are what storing the data that we used in object.

For the self, whatever that is put for the object, will be the same as name we passed the info in.

```
class Student:
    def __init__(self,name,subject,level):
        self.name = name
        self.subject = subject
        self.level = level
```

Object/Class Functions

This is just to show what functions you can use in a class or object. It will mostly be the same as our previous coding style.

So in the same file for our student class, lets define a new function to tell us whether the student is part of the dean list or not. So first you need to add the self.gpa attribute. Else this will not work.

Next, define the function just like in the example here. Make sure its indented or the def aligned with the def on top.
Now you have the dean_list function. Lets try it out.

```
def dean_list(self):
    if self.gpa >= 3.5:
        return True
    else:
        return False
```

The function that created will tell us whether the object/students here are on the dean list or not. Just print it like you would with a variable and function. It will return the value in the terminal.

This is just some small part of the object/class function. Be sure to try it out on different thing.

```
from student import student

student1 = student("Zack",'Scratch','Level 11',3.6)
student2 = student("William",'Python','Level 2',3.3)

print (student2.dean_list())
```

Inheritance

Like the name suggest, inheritance will let you inherit the attributes from a class to another class.
By doing so, you don't need to write another lines of codes again.

Its recommended to open another project as base for this task.

Create 3 different file. Main, Chef and Mamak(this can be changed according to you) .

First we are going to use the Chef file to create a class and define what the chef can do.

Inside Chef file

Create the chef class first. Then define 3 functions with each of them printing what they are cooking. Follow the example.

```
class Chef:
    def make_chicken(self):
        print ("The chef cooks a chicken")

    def make_salad(self):
        print ('The chef makes a salad')

    def make_special_dish(self):
        print ("The chef makes a bbq beef")
```

Inside the Main file

Now we need to import the stuff. Start with from Chef import Chef. Then create the object with the name mychef = Chef(). Try calling it using the function. It should print out the strings from the function before.

```
from Chef import Chef

mychef = Chef()
mychef.make_special_dish()
```

```
C:\Users\haziq\PycharmProj
The chef makes a bbq beef
```

Inherit from Chef

Let say the Mamak chef can do most of what the Chef can do but it have different dishes. Do we need to copy paste the code from the Chef to Mamak?

No you don't. This is where you use the inherit feature. Now go to the Mamak file. You will still need to import the file and functions from the Chef so make sure to do it at the start

```
from Chef import Chef

class mamak (Chef):
```

When classing the mamak, create a bracket after it with the Chef inside. Now it have inherited the functions from Chef.

Define more in mamak:

Like I said previously, the mamak chef know some different dishes. So we need to add those dishes. Don't worry, it will not overwrite the functions in Chef. Type in the code in example.

An exception for the overwrite rule is if you print the same function but different strings, make sure define in the file that is inheriting it.

```
class mamak (Chef):

    def make_fried_chicken(self):
        print ("The chef makes fried chicken")

    def make_special_dish(self):
        print ("The chef makes nasi kandar")
```

Test it out

Now go back to the main file. Copy the code in the example. If you have changed special dish in the mamak, it should appear differently than the one from the chef. Also, try running other function in the mamak to see if it inherited the functions from chef or not.

```
from Chef import Chef
from mamak import mamak

mychef = Chef()
mychef.make_special_dish()

myMamak = mamak()
myMamak.make_special_dish()
```

You can direct message your teacher and ask your question through Slack Robotene Community or arrange a One-to-One Consultation with your teacher.

Any Questions?

Thank you :)