

ICS-104

# Chapter 9

# Notes

By: Naif Alqahtani

 Twitter: @NaifAlqahtani

 Youtube.com/MWNLLT

## • Classes:

- Classes are basically a package of functions
- When functions are used constantly, it may be better to put them in **classes** together.

## • Object Oriented Programming (OOP):

- OOP is a concept based on programming objects containing **data** and **code**.
- We have used objects in python before. For example: strings.
- Strings are objects because they contain **data** & **code** (methods)

contains Data  
↓  
"hello, world!".upper()  
└── string object ─┘ └── contains (method) code ─┘

- Here we have a **class** **str** which has multiple methods (code)
- We can create many objects of **class** **str**. Each object has access to all **methods** (code) in the class **str**.
- In the example: "hello, world!" is an **object** of **class** **str** meaning it has access to all **methods** (code) in class **str** like:
  - .upper()
  - .isdigit()
  - .rstrip(), etc...
- There are other built-in classes in python like the class of Lists etc..
- You **cannot** use methods of one class on other classes.

## • Public Interfaces:

- The set of all methods provided by a class, together with a description is called:

The public interface of the class.

- As a programmer, you do not need to know how letters are converted to unicode or how variables are stored in memory.
- All a programmer needs to know is the public interface like what methods one can use & what they do.
- This process of providing a public interface and hiding its implementations is called encapsulation.
- It is common that the implementations change like making the language more efficient. These changes do not affect the programmers.

## • Class Constructors:

- A class constructor is a class method that initializes important variables for the object.
- Class constructors are automatically called when an object is created.
- In python, the special name `__init__()` is used to define a constructor method.
- Each class cannot have more than constructor.
- You can create constructor argument that asks for variables need to create an object of the class.
- You can specify default variables inside `__init__`.
- Example below.

• most of these lines are copied from Dr. Wasti's slides

## • Self:

- Every method in every class must contain `self` as an argument. Including constructor methods.
- `Self` is a place holder name for the name of the object that has been created with that class.
- Variables that use `self` are called `instance variable`. Those that don't are simply Local Variables.
- You must also use `self` to refer to variables as `instance variables`.
- The name given to the object when creating the class object is called `object reference`.
- Object reference "takes" the place of `self` inside the class. This allows class methods to be called on that object.

## • Creating a Class:

- Suppose we want to create a class called `point`.
- This class needs to take two arguments to be created: `x`, `y`.
- This class has 3 methods: `.getX()`, `.getY()`, `.Translate()`
- First we set our class header with its name:

```
class point:
```

- Now we create our constructor method like this:

```
def __init__(self, x=0, y=0):
```

↑ constructor    ↑ must include    ↑ take X Default to zero    ↑ take Y Default to zero

- Inside our constructor method, we need to define our instance variables:

```
self.x = x  
self.y = y
```

must include    set their values to the numbers passed when object was created

- Now let's create our `.translate()` method that updates our instance variable:

```
def translate(self, dx, dy):
```

↑ must    ↑ values to increment

```
self.x += dx  
self.y += dy
```

update instance variables    add:    values passed into argument

• Finally, the `.getX()` `.getY()`

- Example of encapsulation
- A class user should not directly access instance variables which is why we return them.

```
def getX(self):
    return self.x

def getY(self):
    return self.y
```

*Annotations:*  
 - `self` in `getX` and `getY` is labeled "must".  
 - `self.x` and `self.y` are labeled "return instance variable".

## • Creating Objects of a Class:

• Now that we have made our class, let's make objects that use our class:

• We can create as many object of our class as we want.

*Annotations:*  
 - `point` is labeled "use class name".  
 - `A = point(3, 4)` is labeled "creates a class with instance variables self.x = 3 and self.y = 4".  
 - `B = point()` is labeled "since we set a default value, pointB will have self.x = 0 and self.y = 0".  
 - `C = point(-3, 4)` is labeled "creates a class with instance variables self.x = -3 and self.y = 4".

• Now each of our objects (A,B,C) will have separate instance variables and methods!

• We can use any of our 3 methods with any of the objects.

`B.getX()` ← returns 0  
*(No arguments passed!)*

`C.getX()` ← returns -3

`B.Translate(2, 0)` ← returns nothing, but will update instance variables of object B ONLY

`B.getX()` ← returns 2

• Notice: even though the method `getX()` has `self` inside the class, it has no other arguments, so passing any arguments to `.getX()` will cause error.

• Basically if a method is created with `n` arguments, you have to provide `n-1` argument when using it, because `self` is not an argument to be passed!



*# create a class*

```
class point:
```

```
    # constructor method
```

```
    def __init__(self, x=0, y=0): # set default values
```

```
        # set instance variables to arguments passed
```

```
        self.x = x
```

```
        self.y = y
```

```
    # class methods:
```

```
    def Translate(self, dx, dy):
```

```
        self.x += dx
```

```
        self.y += dy
```

```
    def getX(self):
```

```
        return self.x # return instance variables
```

```
    def getY(self):
```

```
        return self.y
```

*# create objects*

```
A = point(3,4)
```

```
B = point() # defaults to x=0 and y=0
```

```
C = point(-3, 4)
```

```
print( B.getX() )
```

```
print( C.getX() )
```

```
B.Translate(2, 0) # returns nothing
```

```
print( B.getX() )
```

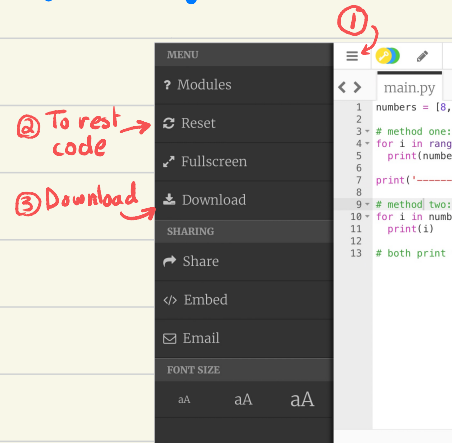
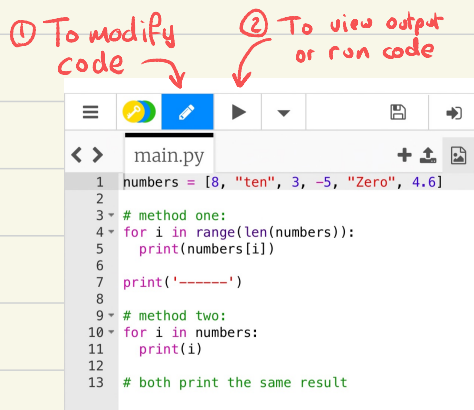
Scan to try code



How to use: Last Page

## • How to use barcodes:

- I added barcodes to programs I mentioned in these notes to help make the notes more interactive & so you can try yourselves.
- I do not know if they will work as intended. (hopefully, they do!)
- I also don't know for how long will they work for (hopefully, for many years)



If you're reading this, I ask Allah to help you with your studies and grant you the marks you need. Don't forget to make dua'a for me and also follow me on my socials.

Best of luck to you all!