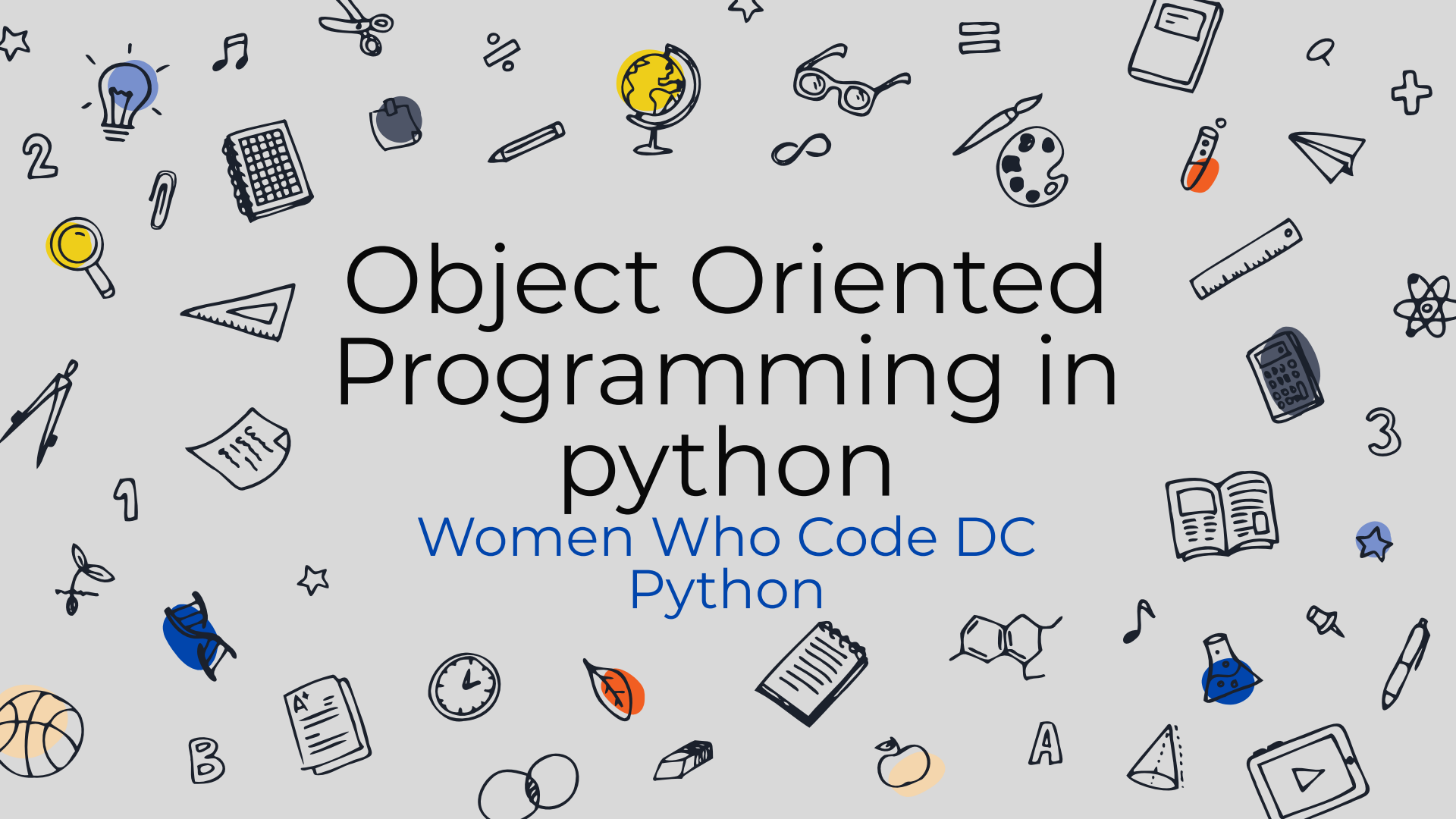


Object Oriented Programming in python

Women Who Code DC Python

Women Who Code

Women Who Code (WWC) is a global non-profit dedicated to inspiring women to excel in technology careers. We work to support this generation in being and becoming leaders and role models in the tech industry.

- We are the DC Chapter!
- Support us! Volunteer or donate.
- Visit our Meetup site: https://www.meetup.com/Women-Who-Code-DC/
- Request to join our Slack chat website: http://bit.ly/wwcdcslack
- Python Beginners: 1st Wednesday of every other month
- Python Lab: 3rd Wednesday of the month

Hello!

I am Bri McGowan

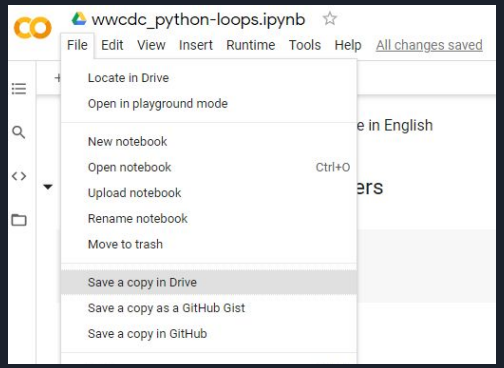
Women Who Code DC Director of Python - Volunteer

Women Who Code DC Slack: @BriannaMcGowan

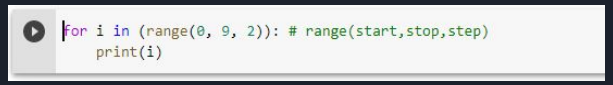
Twitter: @CodeToMove

HOUSEKEEPING

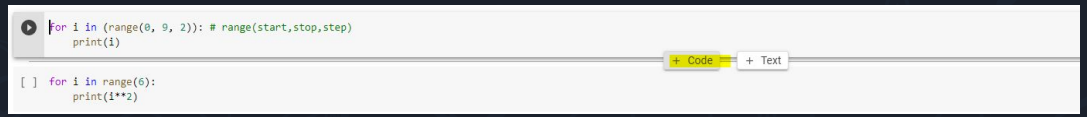
- Use Zoom chat for questions.
- To get started, save a copy of the Colab notebook to your Google drive:



- In Colab:
  - Use Run button to execute code snippet



  - Use “+ Code” button to create your own code cell, copy, edit and run your version of the code snippet



A Quick Overview

Object Oriented Programming (OOP)

Can be used for many different things!

- Data Science
- App Development
- Video Games
- What else?

Advantages of OOP

- Reusability
- Maintainability
- Easier Testing & Debugging
- Extensibility
- Improved Design
- Modularity

Roadmap

Classes, Objects, & Instances	Methods	Encapsulation	Abstraction	Inheritance
Classes are blueprints that are used to represent real-world objects in the program. Objects are instances of a class.	Methods are a body of actions that access the data of the instance via the self parameter.	Encapsulation is an Object Oriented Programming concept that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse.	In Python, data abstraction can be defined as ,hiding all the irrelevant data/process of an application in order to reduce complexity and increase the efficiency of the program.	Inheritance allows us to define a child class that inherits all the methods and properties from parent class.

Classes, Objects, & Instances

Class >> Object >> Instance

- The class = the ‘blue print’ is the abstract description of the object.
- The Object is built based on the ‘blue print’. It is the ‘physical’ existence of the instance.
- An instance is a theoretical concept of object. You create an instance by instantiating an object.

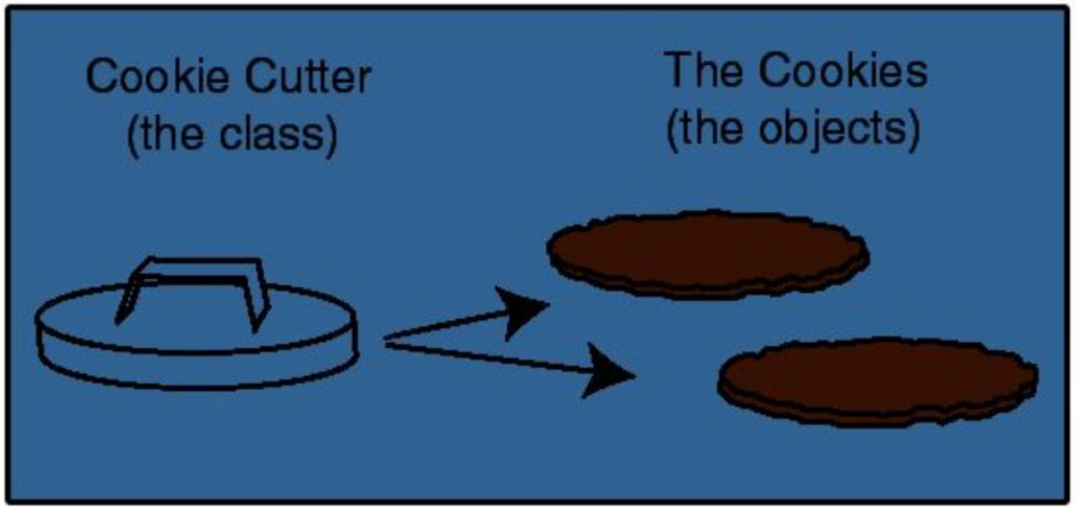
**Colab

But first we have to define Object v Class

Class is the mold that provides the structure for creating objects.

From Quora:

Example below :



Let's Make cookies

Class :

```
1  class Cookie:
2      def __init__(self, price, color):
3          self.price = price
4          self.color = color
```

Object :

```
1  my_cookie1 = Cookie(100, 'Red')
2  my_cookie2 = Cookie(5, 'Yellow')
```

Class vs. Instance ATTRIBUTES

Class	Instance
Belongs to the class	Belongs to the instance
Shared by all the instances	Each instance has a separate copy of each instance attribute
Changing them affects all instances	Changing their value only affects that instance and the other instances remain unchanged
Usually located at the top of the class	

INSTANCE ATTRIBUTES

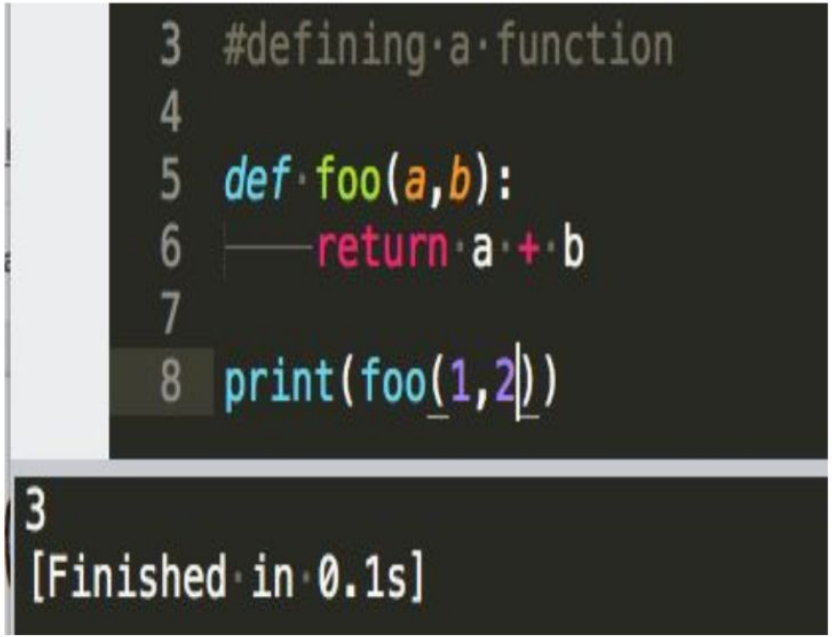
- Instance attributes are independent for each instance

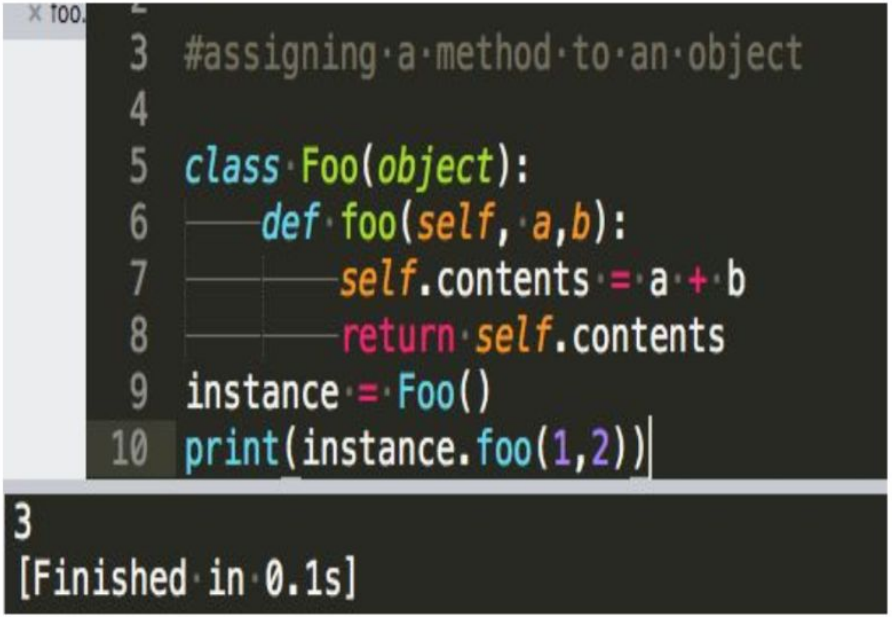
Attribute: Value

- If a value of an instance changes, the other instances are not affected

Methods

Methods are associated with Objects/classes; Functions are not





def __init__(self, parameters):

- Reserved method
- Kow as the class constructor
- Executed when an instance is created

Why use “self”?

- Refers to the instance
- By using the “self” keyword we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.

  i.e. self.price = price

Methods

- Access the data of the instance via the self parameter
- Body of actions

Example Time!

We will be working through the example posted in the chat

Encapsulation & Abstraction

Encapsulation & Abstraction

Encapsulation

- Bundling of data & methods into a single class
- Restrict Access & Information Hiding
- Public vs. Private Attributes

Abstraction

- Interface with the functionality of code that the user doesn't see

Public vs Private

Public

- Can be accessed anywhere in the code
- No access restriction

Self.model = model

Private

- Can’t or shouldn’t be accessed outside the class
- Two levels of access restriction

Self._id_num = id_num

Getters & Setters

- Members of a class (methods)
- Purpose is to “get” and “set” the value of an instance attribute
- Protect data by providing an indirect way to access & modify it

Getters

- Let us access the attribute indirectly

Setters

- Modify an attribute indirectly

Properties

A built-in function that lets us access the instance attribute by using the properties of the getter and setter

@Property Decorator:

- More compact
- More readable
- Avoid calling property directly
- Avoid namespace pollution
  - No get_<attr>
  - No set_<attr>

@property Decorator cont.

- You can define properties with the @property syntax, which is more compact and readable.
- @property can be considered the "pythonic" way of defining getters, setters, and deleters.
- By defining properties, you can change the internal implementation of a class without affecting the program, so you can add getters, setters, and deleters that act as intermediaries "behind the scenes" to avoid accessing or modifying the data directly.

Example Time!

We will be working through the example posted in the chat

Inheritance

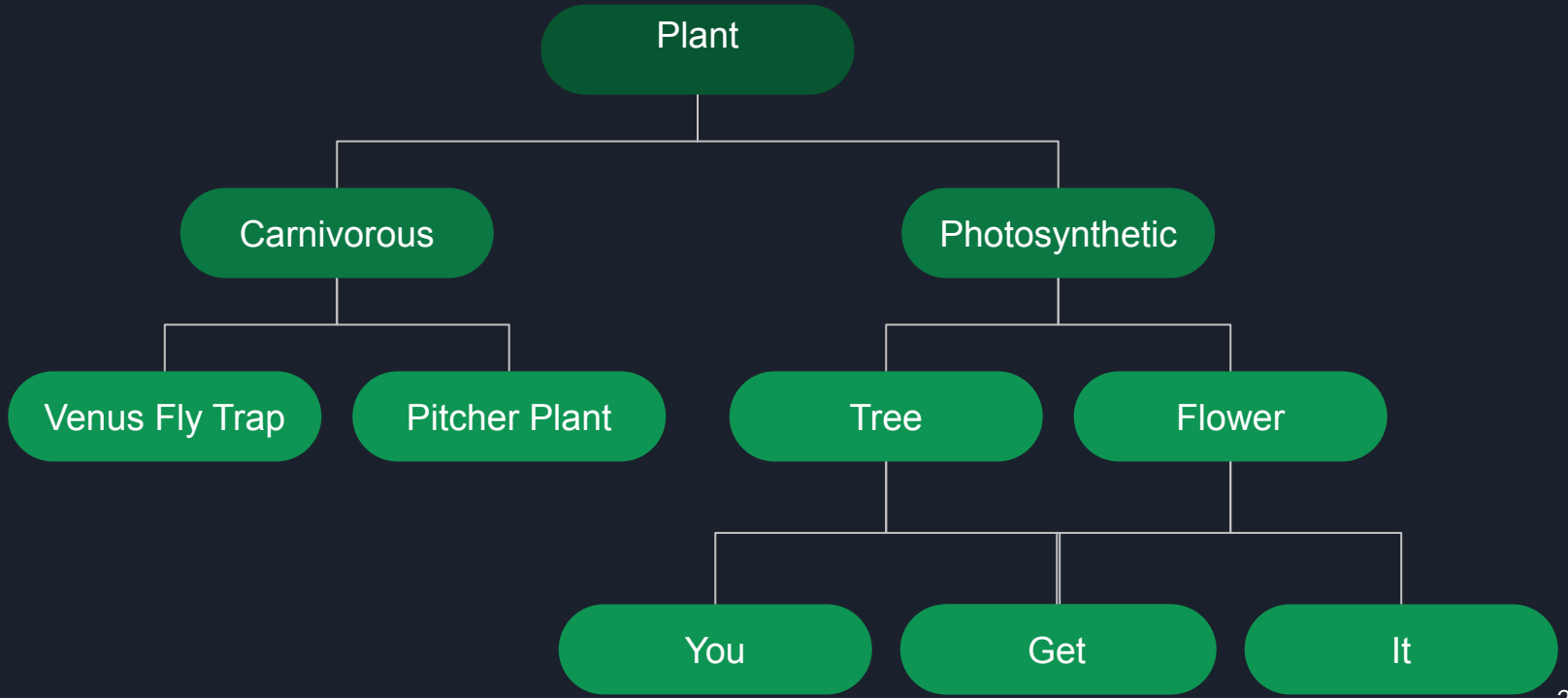
Inheritance

Principle: Don’t Repeat Yourself!!! Aka avoiding repetition

Format: (inherited child class) is a type of (parent class)

- Child class inherits attributes and methods from parent
- Parent class is more abstract and reusable where other classes inherit its attributes and methods

Example


Plant
Carnivorous
Photosynthetic
Venus Fly Trap
Pitcher Plant
Tree
Flower
You
Get
It

Syntax

```
Class ChildClass(ParentClass):
    def __init__(self, arguments):
        ParentClass.__init__(parent arguments)
```

Example Time!

We will be working through the example posted in the chat

Resources <3

- Our WWCodeDC Python Github:

  https://github.com/womenwhocodedc/python-community

- Inspiration for this lab: The Come Up

  https://www.youtube.com/watch?v=7J_qcttfnJA

- Slack: Sign up! womenwhocodedc.slack.com

Python Training/ Resources

In Person:
WWCDC (1st & 3rd wed)
Hear Me Code (Once a Month on Saturdays)

ONLINE
- Coursera
- HackerRank
- StackOverflow
- Slack
- CodeCombat
- GitHub
- Kaggle
- EDX by microsoft
- How to Think Like a Computer Scientist - PDF
- Learn Python the Hard Way
- Data Camp

THANKS!

Any questions?

You can find me on Women Who Code DC Slack
@BriannaMcGowan

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash