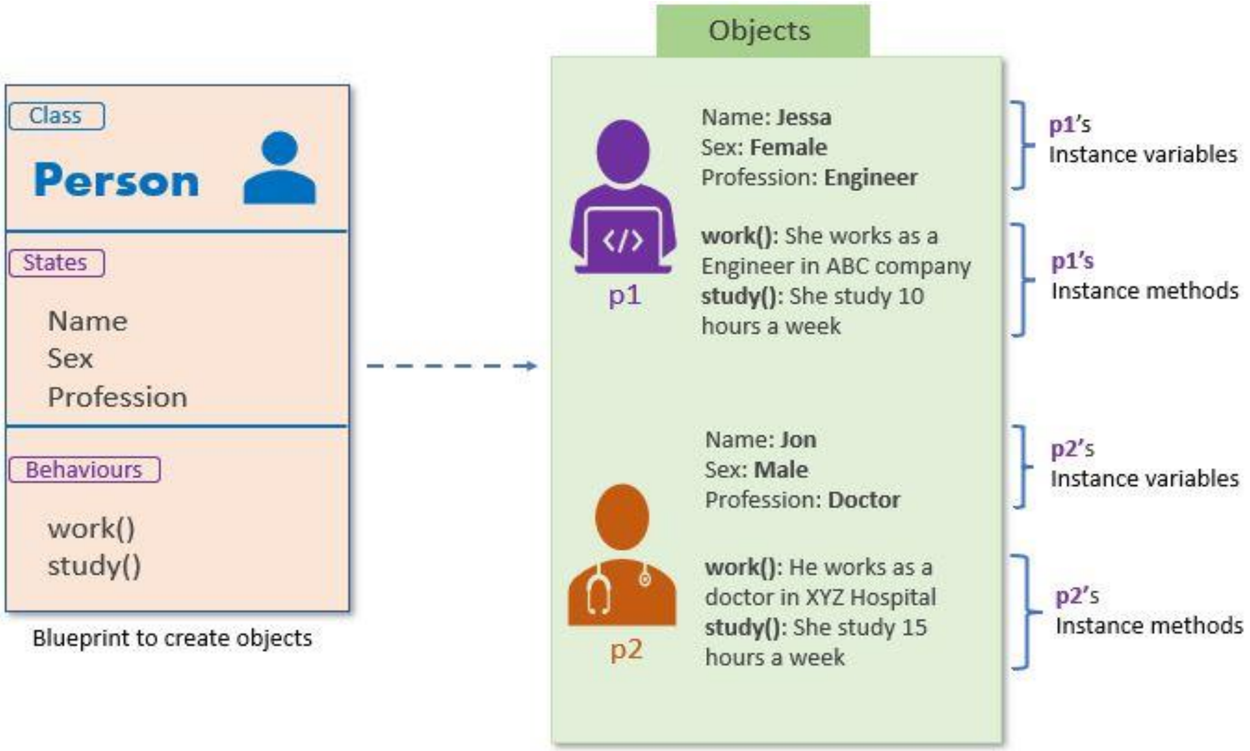
Artificial Intelligence (Machine Learning & Deep Learning) [Course]

Week 2 - Day 2 & Day 3

[See examples / code in GitHub code repository]

It is not about Theory, it is 20% Theory and 80% Practical – Technical/Development/Programming [Mostly Python based]

Class and Object

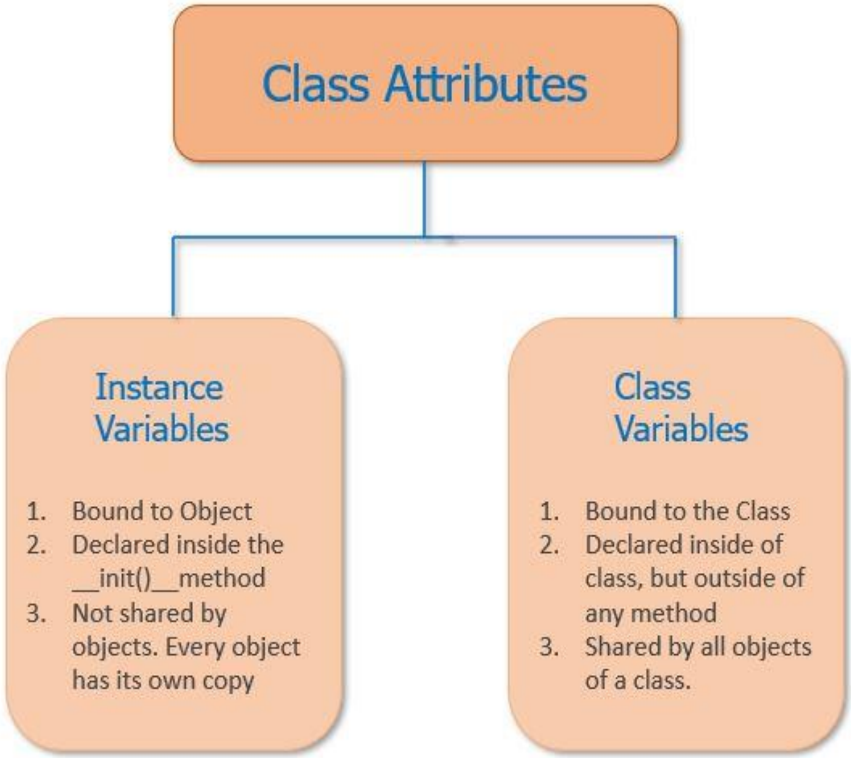


References:

https://www.w3schools.com/python/python_classes.asp
https://www.programiz.com/python-programming/class
https://www.tutorialspoint.com/python/python_classes_objects.htm
https://pynative.com/python-classes-and-objects/

Exercises

Class and Object



Class Attributes

Instance Variables

1. Bound to Object
2. Declared inside the __init()__method
3. Not shared by objects. Every object has its own copy

Class Variables

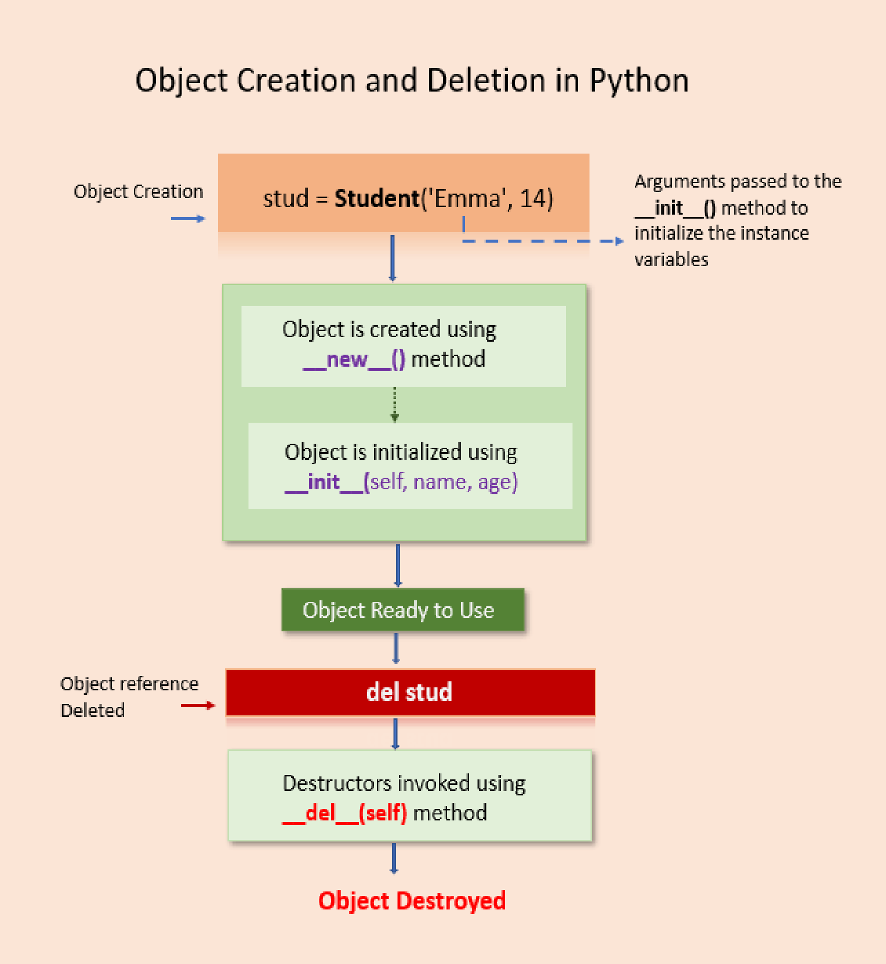
1. Bound to the Class
2. Declared inside of class, but outside of any method
3. Shared by all objects of a class.

References:

https://www.w3schools.com/python/python_classes.asp
https://www.programiz.com/python-programming/class
https://www.tutorialspoint.com/python/python_classes_objects.htm
https://pynative.com/python-classes-and-objects/

Exercises

Constructors and Destructors

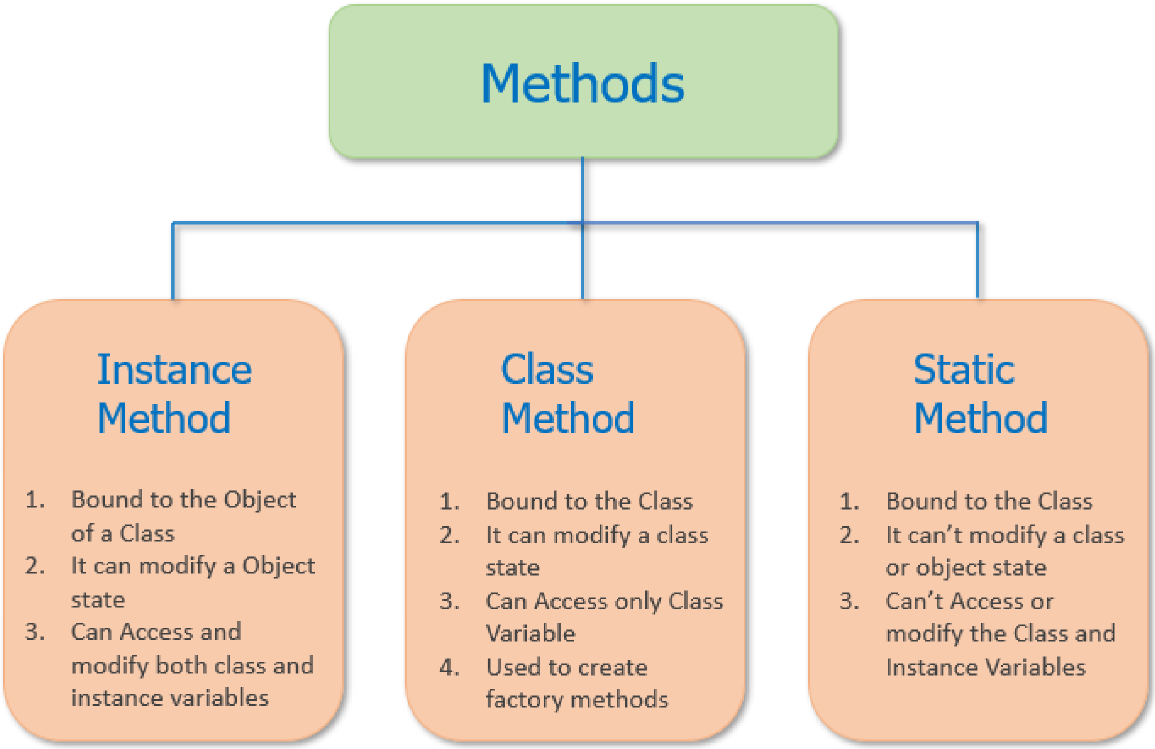

Object Creation and Deletion in Python
Object Creation
stud = Student('Emma', 14)
Arguments passed to the __init__() method to initialize the instance variables
Object is created using __new__() method
Object is initialized using __init__(self, name, age)
Object Ready to Use
Object reference Deleted
del stud
Destructors invoked using __del__(self) method
Object Destroyed

References:

https://pynative.com/python-destructor/
https://www.analyticsvidhya.com/blog/2024/02/destructor-in-python/
https://codedamn.com/news/python/destructors-in-python
https://medium.com/@abhishekjainindore24/demystifying-constructors-and-destructors-in-python-a-beg
b5bd6988f4bb

Exercises

Types of Method



Methods

Instance Method

1. Bound to the Object of a Class
2. It can modify a Object state
3. Can Access and modify both class and instance variables

Class Method

1. Bound to the Class
2. It can modify a class state
3. Can Access only Class Variable
4. Used to create factory methods

Static Method

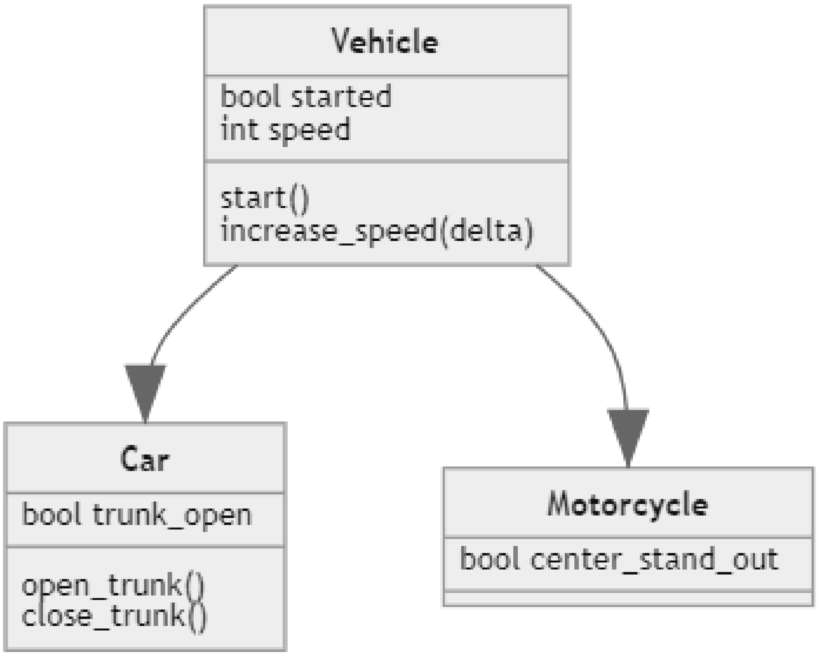
1. Bound to the Class
2. It can't modify a class or object state
3. Can't Access or modify the Class and Instance Variables

References:

https://pynative.com/python-class-method-vs-static-method-vs-instance-method/
https://www.linkedin.com/pulse/static-method-vs-class-instance-python-3-ryan-parsa-kvgdc/
https://medium.com/codex/python-class-methods-class-vs-instance-vs-static-methods-96d075d27c68
https://realpython.com/instance-class-and-static-methods-demystified/

Exercises

Inheritance



References:

https://www.programiz.com/python-programming/polymorphism
https://www.toppr.com/guides/python-guide/tutorials/python-oops/polymorphism-in-python-with-examples/
https://www.w3schools.com/python/python_polymorphism.asp
https://www.almabetter.com/bytes/tutorials/python/python-inheritance-and-polymorphism

Exercises

Types of Inheritance

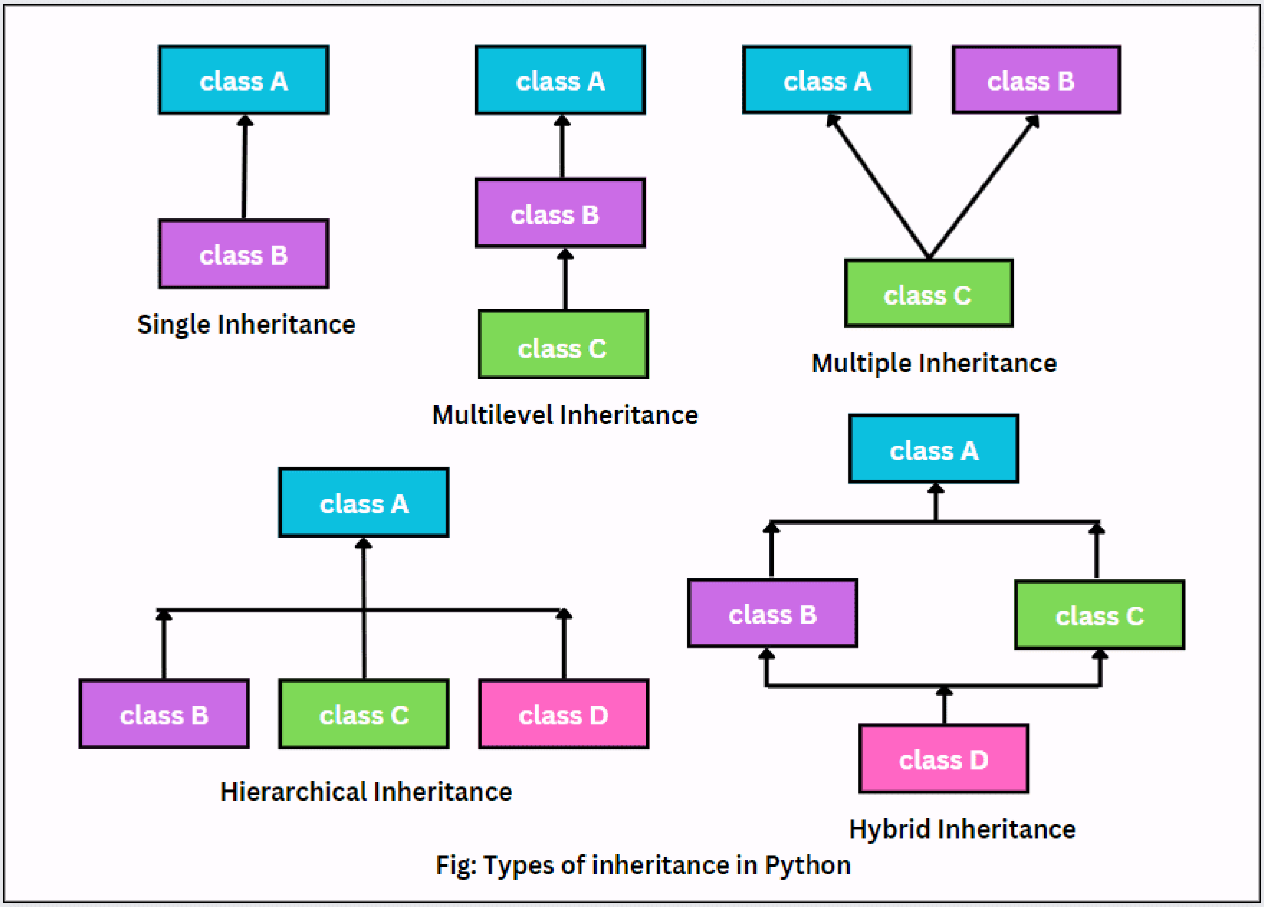


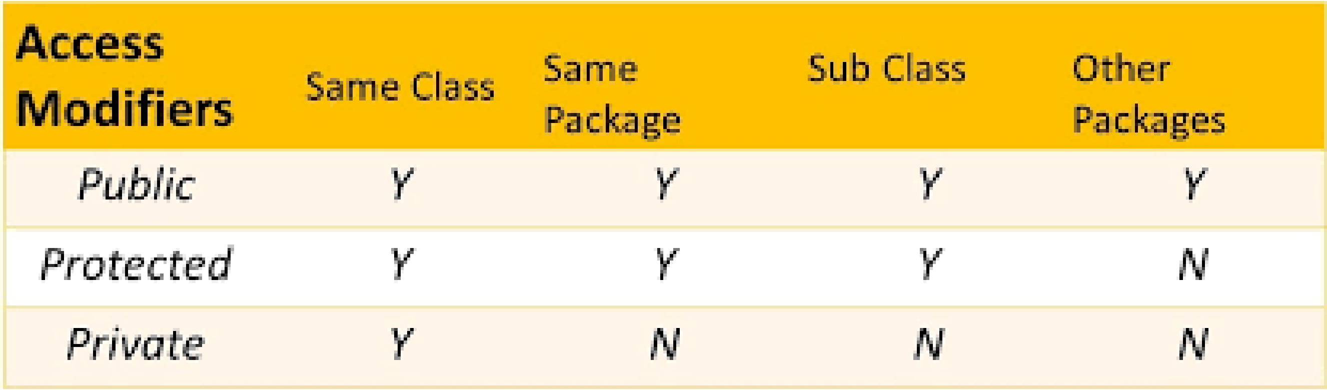
Fig: Types of inheritance in Python

References:

https://www.scientecheasy.com/2023/09/types-of-inheritance-in-python.html/
http://www.btechsmartclass.com/python/Python_Tutorial_Python_Inheritance.html
http://www.btechsmartclass.com/python/Python_Tutorial_Python_Inheritance.html
https://innovationyourself.com/types-of-inheritance-in-python/

Exercises

Access Specifiers: Private, Public, Protected



Access Modifiers	Same Class	Same Package	Sub Class	Other Packages
Public	*Y*	*Y*	*Y*	*Y*
Protected	*Y*	*Y*	*Y*	*N*
Private	*Y*	*N*	*N*	*N*

References:

https://www.scaler.com/topics/access-modifiers-in-python/
https://www.tutorialspoint.com/access-modifiers-in-python-public-private-and-protected
https://www.studytonight.com/python/access-modifier-python
https://www.tutorialsteacher.com/python/public-private-protected-modifiers

Exercises

Polymorphism : Compile Time Polymorphism/Overloading

Compile-Time Polymorphism (Method Overloading)
Method overloading occurs when a class contains many methods with the same name. The types and amount of arguments passed by these overloaded methods vary. Python does not support method overloading or compile-time polymorphism. If there are multiple methods with the same name in a class or Python script, the method specified in the latter one will override the earlier one.

Python does not use function arguments in method signatures, hence method overloading is not supported.

References:
https://www.toppr.com/guides/python-guide/tutorials/python-oops/polymorphism-in-python-with-examples/

Exercises

Polymorphism : Run Time Polymorphism/Overriding

Like in other programming languages, the child classes in Python also inherit methods and attributes from the parent class. We can redefine certain methods and attributes specifically to fit the child class, which is known as **Method Overriding**.

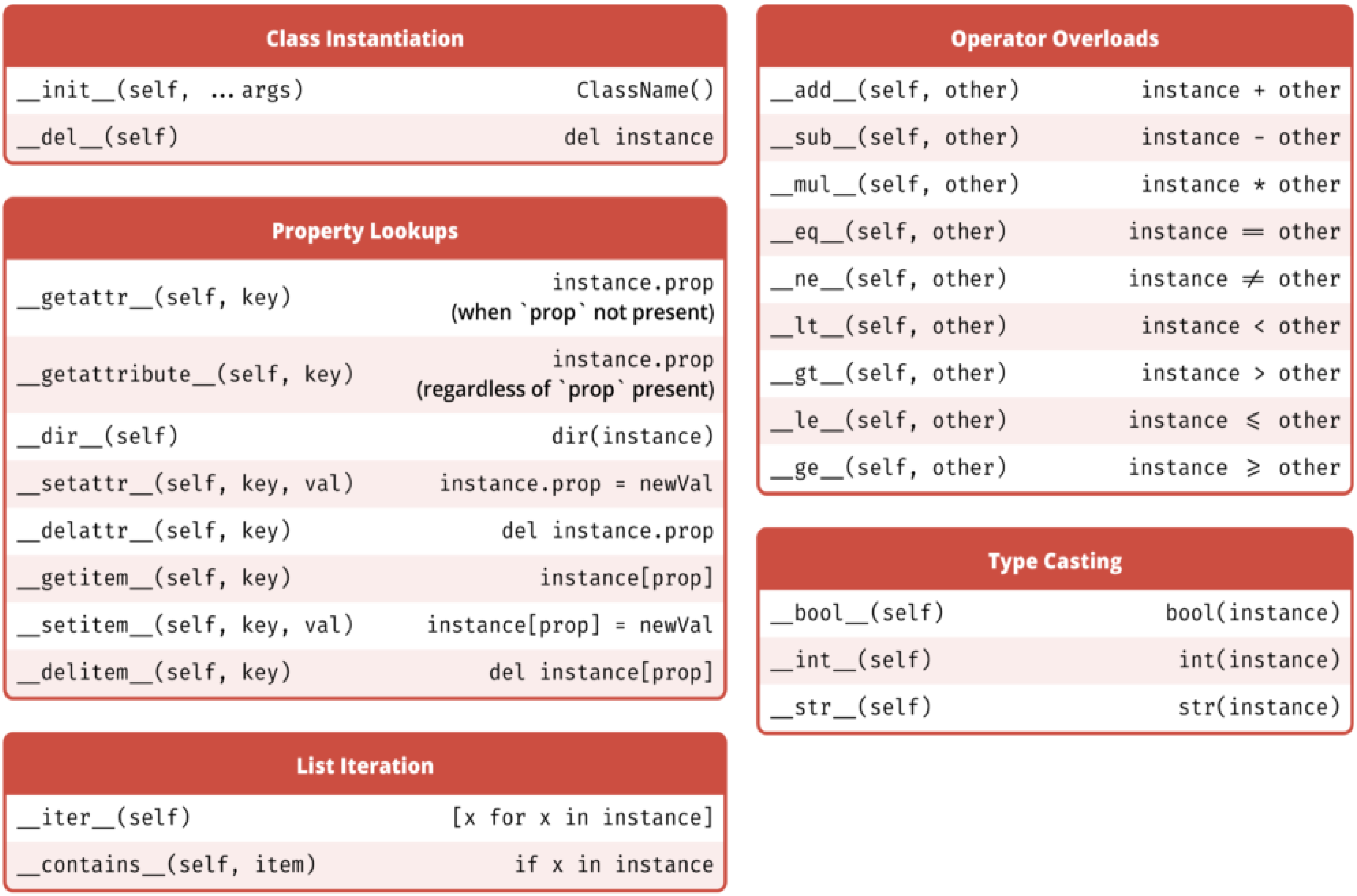
Polymorphism is supported in Python via method overriding and operator overloading. However, Python does not support method overloading in the classic sense.

References:

https://algodaily.com/lessons/association-aggregation-composition-casting/python
https://faun.pub/association-aggregation-composition-python-ec9947832cbd
https://www.geeksforgeeks.org/python-oops-aggregation-and-composition/

Exercises

Magic Functions/Dunder Functions



Class Instantiation

`__init__(self, ...args)`	`ClassName()`
`__del__(self)`	`del instance`

Property Lookups

`__getattr__(self, key)`	`instance.prop` (when `prop` not present)
`__getattribute__(self, key)`	`instance.prop` (regardless of `prop` present)
`__dir__(self)`	`dir(instance)`
`__setattr__(self, key, val)`	`instance.prop = newVal`
`__delattr__(self, key)`	`del instance.prop`
`__getitem__(self, key)`	`instance[prop]`
`__setitem__(self, key, val)`	`instance[prop] = newVal`
`__delitem__(self, key)`	`del instance[prop]`

List Iteration

`__iter__(self)`	`[x for x in instance]`
`__contains__(self, item)`	`if x in instance`

Operator Overloads

`__add__(self, other)`	`instance + other`
`__sub__(self, other)`	`instance - other`
`__mul__(self, other)`	`instance * other`
`__eq__(self, other)`	`instance == other`
`__ne__(self, other)`	instance ≠ other
`__lt__(self, other)`	`instance < other`
`__gt__(self, other)`	`instance > other`
`__le__(self, other)`	instance ⩽ other
`__ge__(self, other)`	instance ⩾ other

Type Casting

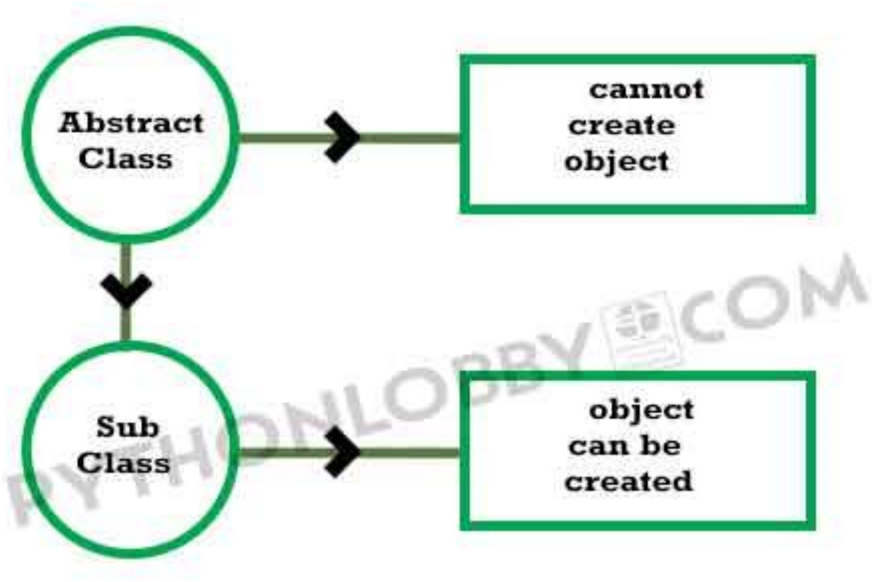
`__bool__(self)`	`bool(instance)`
`__int__(self)`	`int(instance)`
`__str__(self)`	`str(instance)`

References:

https://realpython.com/python-magic-methods/
https://www.tutorialsteacher.com/python/magic-methods-in-python
https://builtin.com/data-science/dunder-methods-python

Exercises

Abstract Method and Class, Empty Class, Data Class

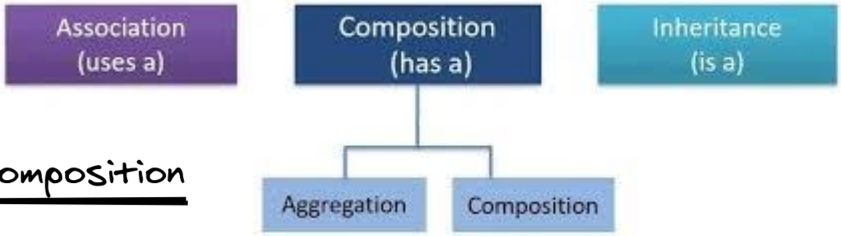


References:

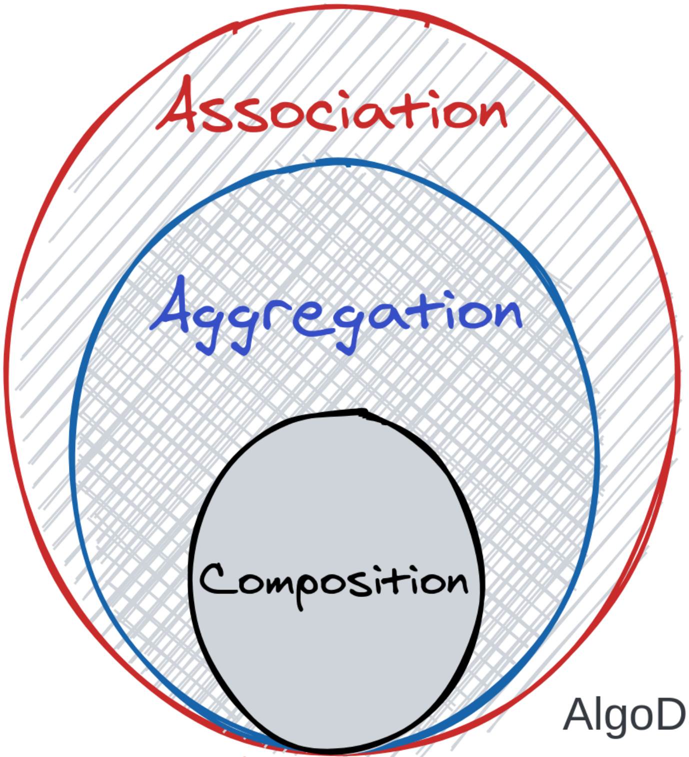
https://www.scaler.com/topics/abstract-class-in-python/
https://pythonlobby.com/abstract-class-in-object-oriented-programming-oops-in-python-programming/#goo
https://www.datacamp.com/tutorial/python-abstract-classes
https://www.datacamp.com/tutorial/python-data-classes
https://realpython.com/python-data-classes/
https://www.dataquest.io/blog/how-to-use-python-data-classes/

Exercises

Inner/Nested Class Association, Aggregation, Composition



Association, Aggregation, Composition



AlgoDaily

References:

https://algodaily.com/lessons/association-aggregation-composition-casting/python
https://faun.pub/association-aggregation-composition-python-ec9947832cbd
https://www.geeksforgeeks.org/python-oops-aggregation-and-composition/

Exercises

Thank you - for listening and participating

- □ **Questions / Queries**
- □ **Suggestions/Recommendation**
- □ **Ideas…..?**

Shahzad Sarwar
Cognitive Convergence
https://cognitiveconvergence.com
shahzad@cognitiveconvergence.com
voice: +1 4242530744 (USA) +92-3004762901 (Pak)