1. A module is a file that contains functions and values that you can reference from your program. The import statement lets you import a module into our code. We would want to use the import statement when we don't want to write an entire class definition (in example) that is already written, and it improves the readability of the code.
2. When we import a file, we are usually using an absolute path, because it includes the full path to locate the desired resources and is based on the project’s root directory.

Sometimes we use the relative imports because it allows including code from sub-modules within the same package, that in comparison to the absolute path, doesn’t include, and this approach saves you time and simplifies imports in larger projects by using the relationship between files to locate resources. For our example, we would use an absolute path, because we don’t want a specific class or function, we want the whole file, so it will look like this:

import subdir.my\_file

1. Python interpreter analyses source file and defines a few special variables/global variables before running code.

The Python interpreter sets the special \_\_name\_\_ variable to have the value "\_\_main\_\_" if that module (the source file) is being run as the main application. \_\_name\_\_ will be set to the name of the module if this file is being imported from another module. The \_\_name\_\_ global variable accepts the module name as a value, so in conclusion, we can use an if \_\_name\_\_ == "\_\_main\_\_" block to allow or prevent parts of code from being run when the modules are imported.

1. A class in Python is a user-defined data type that includes both the actual data and the methods that can be used to alter it. Classes act as a kind of template for building objects. They describe the traits and functions that the objects will use. Consider a class as a model for a structure.
2. Functions are great to use when data is central to the work being done. Classes are great when you need to represent a collection of attributes and methods that will be used over and over again in other places. In comparison to instance variable, if i would like to change the value of a instance variable, it doesn’t affect all the instances of that variable, but for classes,, the changes affects all the instances.

- You can create numerous base classes using the inheritance approach.

- After you create a class, you can modify its attributes and methods during program runtime, and use it as many times as you’d wish.

- The class structure is an organised structure that helps with maintaining the code.

1. A class variable characterises the properties of a whole course of objects, and applies to all occasions of it.

For example,

class Shark:

animal\_type = "fish" # class variable

location = "ocean" # class variable

followers = 5 # class variable

1. Object of a class is making a variable of the class with personalised data into the class in the main program (and not in the class itself).

For example,

Fish1 = Shark() # class object

1. A constructor is a special function that gets called consequently when an object is made of a class. The main reason of a constructor is to initialise or relegate values to the information individuals of that class. It cannot return any value other than none.

Destructors are called when an object gets destroyed.

1. Self speaks to the occurrence of class. This keyword permits you to get to variables, properties, and methods of a characterised class.
2. Inheritance permits us to define a class that acquires all the methods and properties from another class. Parent class is the class being inherited from, too called base class. Child class is the class that inherits from another class. Inheritance gives code reusability to the program since we will use an existing class to form a new class rather than making it from scratch.
3. Polymorphism means “many forms” , and allows to many methods or operators to be with the same name that executes on many classes or objects.
4. A static class is bound to a specific class rather then the object for that class, which means that you can call a static method without an object for that class.so we would want to use static functions when we would want a function that doesn’t get to any of the properties of a class but makes sense that it belongs to that class. On the other hand, in a regular method, you have to create an object, because it uses dynamic binding.

For example,

Static method:

@staticmethod

def addNumbers(x, y):

return x + y

Static property:

class MyClass:

i = 3

Enums:

1. Enum is a special “ class “ that represents a group of consts. It can be iterated over to return its canonical members in definition order.
2. - Enums help prevent the use of incorrect or inconsistent values in your code, reducing the risk of bugs and errors.

- Enums provide meaningful names to values, making the code more human-readable and self-explanatory.