**Question # 1**

**O**bject-**o**riented **p**rogramming (**OOP**) refers to a type of computer programming (software design) in which [programmers](https://www.webopedia.com/TERM/P/programmer.html) define the [data type](https://www.webopedia.com/TERM/D/data_type.html) of a [data structure](https://www.webopedia.com/TERM/D/data_structure.html), and also the types of operations ([functions](https://www.webopedia.com/TERM/F/function.html)) that can be applied to the data structure.

In this way, the data structure becomes an [object](https://www.webopedia.com/TERM/O/object.html) that includes both [data](https://www.webopedia.com/TERM/D/data.html) and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects.

**Question # 2**

* It provides a clear ***modular structure*** for programs which makes it good for defining abstract datatypes in which implementation details are hidden
* Objects can also be ***reused*** within an across applications. The reuse of software also lowers the cost of development. More effort is put into the object-oriented analysis and design, which lowers the overall cost of development.
* It makes software***easier to maintain.*** Since the design is modular, part of the system can be updated in case of issues without a need to make large-scale changes.
* Reuse also enables ***faster development***. Object-oriented programming languages come with rich libraries of objects, and code developed during projects is also reusable in future projects.
* It provides a good framework for code libraries where the supplied software components can be ***easily adapted and modified by the programmer***. This is particularly useful for developing graphical user interfaces.
* ***Better Productivity*as** OOP techniques enforce rules on a programmer that, in the long run, help her get more work done; finished programs work better, have more features and are easier to read and maintain. OOP programmers take new and existing software objects and "stitch" them together to make new programs. Because object libraries contain many useful functions, software developers don't have to reinvent the wheel as often more of their time goes into making the new program.

**Question # 3**

A **Function** is a piece of code that is called by name. It can be passed data to operate on (i.e. the parameters) and can optionally return data (the return value). All data that is passed to a function is explicitly passed.

A **Method** is a piece of code that is called by a name that is associated with an object. In most respects it is identical to a function except for two key differences:

1. A method is implicitly passed the object on which it was called.
2. A method is able to operate on data that is contained within the class (remembering that an object is an instance of a class - the class is the definition, the object is an instance of that data).

**Question # 4**

* **Object** − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.
* **Class** − A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.
* **Attributes** are the characteristics of the class that help to distinguish it from other classes.
* **Behaviors** are the tasks that an object performs. A person's attributes, for example, include their age, name, and height, while their behaviors include the fact that a person can speak, run, walk, and eat.

**Question # 5**

class car:

name = ""

kind = "car"

color = ""

value = 100.00

def description(self):

desc\_str = "%s is a %s %s worth $%.2f." % (self.name, self.color, self.kind, self.value)

return desc\_str

# your code goes here

car1 = car()

car1.name = "Fer"

car1.color = "red"

car1.kind = "convertible"

car1.value = 60000.00

car2 = car()

car2.name = "Jump"

car2.color = "blue"

car2.kind = "van"

car2.value = 10000.00

# test code

print(car1.description())

print(car2.description())

obj = car('Ferrari','Red','truck','100000')

obj. description ()