Before going through all of this, in General all the object oriented programming (OOP) concepts are derived directly from real life so that we can almost assure we can program anything we can see.

For example, Human is a class (or a category) where as “Ram” is an object. Class contains the blue print like human have 2 legs, 2 hands, 2 eyes and may have one or more names, age (There are all properties mentioned in using of variables) and actions like human can walk, talk etc.

Ram is an object created from the blueprint called class human.

Let’s see another example, here “Bangaru” is a class having certain properties (look, name, wealth etc) and actions (mannerisms etc). Ram is a child class which was inherited from parent class “Bangaru”. Child class may have full or partial inheritance. Means Ram may have all the qualities and properties Bangaru had or only few with Ram’s own additional properties and actions.

Note: Where ever “properties” or “attributes” are used, it means variables in a class and where ever actions, behaviors are used, it means functions/methods in a class.

Multiple inheritance is simple just like normal inheritance but here there are 2 base class. Bangaru and Susila. Child class Ram can have some or all of Bangaru’s properties and functions and some or all of Susila’s properties and functions and Ram can have its own additional properties and functions.

Finally ram can overwrite some of its parent class functions/properties too.

In the same order we can pretty much explain everything in OOP with real time example and it supposed to make programming easy and fun.

But these are all for your understanding, if we talk like this most of the interviewers may not get (because they might have got questions and answers from internet) so let’s get into technical words.

**Inheritance**

Inheritance is used to indicate that one class will get most or all of its features from a parent class. This happens implicitly whenever you write ``**class ram(bangaru)**`` and it means ram is the class created and inherited from another class called Bangaru. Bangaru is the base class and ram is child class.

Example 1: Using parent class functions.

class Parent(object):

def walk(self):

print "walking"

class Child(Parent):

pass

dad = Parent()

son = Child()

dad.walk()

son.walk()

In the above example, even though child class has no walk function, it still call’s parent’s function because it’s inherited.

And output to the above code would be

walking

walking

Example 2: Override Parent function

class Parent(object):

def walk(self):

print "walking stright"

class Child(Parent):

def walk(self):

print "wakling like drunken :)"

dad = Parent()

son = Child()

dad.walk()

son.walk()

In the above example, child inherited parent’s function but its override with its own function. and output would be,

walking stright

wakling like drunken :)

Example 3: Altered parent function

class Parent(object):

def walk(self):

print "walking"

class Child(Parent):

def walk(self):

super(Child, self).walk()

print "and also running :)"

dad = Parent()

son = Child()

dad.walk()

son.walk()

The above example just like previous one but here it used super keyword to call parent’s function as well. So it’s kind of having parent function + additional functionality and output would be like,

walking

walking

and also running :)

**super():**

Super is like parent’s object where you can call parents function even though child has a function which overrides.

For more read : <https://learnpythonthehardway.org/book/ex44.html>

**Mutable and immutable objects**

This one you may or may not get it from my writing. But actually it’s simple. If you not getting from my writing, please let me know, we can talk.

In simple words, mutable means liable to change and immutable is something unchanging over the time.

In python world, let’s take example of List.

List is mutable and string is immutable list. Let me explain more.

Generally as we studied everywhere, string is a sequence of character. In programming string is an array of character(C, C++, and Java etc) and Python has no array and it has only list. So in python, string is a list of characters.

In the below example,

listvar = [1,2,3,4]

stringvar = "Ramkumar"

print listvar[2]

print stringvar[2]

There is a list variable and string variable and am trying to read both of them like treating them as list and it works. Output would be,

3

m

Lets try to change the value,

listvar = [1,2,3,4]

stringvar = "Ramkumar"

listvar[2] = 10

print listvar

stringvar[2] = "n"

print stringvar

In the above example, we try to change 3rd element in both of them and list get changed whereas string will throw error. Output would be like,

[1, 2, 10, 4]

Traceback (most recent call last):

File "c:\Users\rbangaru\Documents\BitBucketRepos\Merge\TestAutomation\temp-hrbkjoryvv.python", line 5, in <module>

stringvar[2] = "n"

TypeError: 'str' object does not support item assignment

So string is immutable list whereas list is mutable.

You may get a doubt like you can assign completely different string to stringvar and it works like below,

stringvar = "Ramkumar"

print stringvar

stringvar = "Bangaru"

print stringvar

Output would be like,

Ramkumar

Bangaru

It works because you are changing the value of a variable. Not changing the object itself (string)

**Pass by value and reference**

In Python, everything is an object. So right way to call this is, Pass by object and Pass by object reference.

In python we won’t explicitly say pass by value and pass by reference. It’s implicit.

Ok. Before going there, what is pass by value and pass by reference in general.

**Pass by value:**  When you parameter a value to a function, it takes copy of the value and do changes there and do not modify the original. Like below,

def print\_with\_hello(welcomestring):

welcomestring = "Hello" + welcomestring

print welcomestring

inputstring = "Ramkumar"

print inputstring

print\_with\_hello(inputstring)

print inputstring

In the above example welcomestring is a copy of inputstring and changes happen to welcomestring did not impact the inputstring. This is called “Pass by value”

**Pass by reference:** Here instead of taking the copy of the value from an object/variable, it takes its reference and use it (like taking alias)

def print\_list(welcomelist):

welcomelist.append(6)

print welcomelist

inputlist = [1,2,3,4,5]

print inputlist

print\_list(inputlist)

print inputlist

Output will be,

[1, 2, 3, 4, 5]

[1, 2, 3, 4, 5, 6]

[1, 2, 3, 4, 5, 6]

The list has been passed and there is an operation on a list which also impacts original list. This is called “Pass by reference”

There is a one more example,

def print\_list(welcomelist):

welcomelist = welcomelist + [6]

print welcomelist

inputlist = [1,2,3,4,5]

print inputlist

print\_list(inputlist)

print inputlist

And output will be,

[1, 2, 3, 4, 5]

[1, 2, 3, 4, 5, 6]

[1, 2, 3, 4, 5]

If you see here original list did not get impacted. Function still received it as reference and in previous example, append function also treated it as reference but in this one assignment operator made it as value hence reference dropped and copy of the value had been taken which is why original value did not get impacted.

If you have noticed, String and list reacted differently, so in general rule in python is,

1. Immutable object goes by value
2. Mutable object goes by reference.

**Reverse index**

Let’s take example of list again,

inputlist = [1, 2, 3, 4, 5]

#indexing is

print inputlist[2]

#Reverse indexing is

print inputlist[-1]

and output would be,

3

5

Reverse indexing is generally used in sequence like list, string etc. indexing starts with 0 and navigate from left to right where as reverse indexing starts from -1 and comes from right to left. So when we use -1 in list indexing, it returns last element of the list.

Advantages of python

It’s a relative question, you can say any 3 or 4 of them from below link,

<https://en.wikiversity.org/wiki/Python_Concepts/Why_learn_Python>

My choice would be,

* Cross-platform
* Widely Supported
* It's Safe because doesn’t have pointers like concepts
* Object oriented programming
* Easy Syntax

Regular expression

A little big one, the below website covers well. Let me also try to write something simple in the meantime.

<https://docs.python.org/2/library/re.html>