1. **What are python modules? Name some commonly used built-in modules in Python?**

*Ans:* Python modules are files containing Python code. This code can either be functions classes or variables. A Python module is a .py file containing executable code.

Some of the commonly used built-in modules are:

* os
* sys
* math
* random
* data time
* JSON

1. **How to install Python on Windows and set path variable?**

*Ans:* To install Python on Windows, follow the below steps:

* Install python from this link: <https://www.python.org/downloads/>
* After this, install it on your PC. Look for the location where PYTHON has been installed on your PC using the following command on your command prompt: cmd python.
* Then go to advanced system settings and add a new variable and name it as PYTHON\_NAME and paste the copied path.
* Look for the path variable, select its value and select ‘edit’.
* Add a semicolon towards the end of the value if it’s not present and then type %PYTHON\_HOME%

3. **What is \_\_init\_\_?**

*Ans:* \_\_init\_\_ is a method or constructor in [Python](https://www.edureka.co/blog/python-programming-language). This method is automatically called to allocate memory when a new object/ instance of a class is created. All classes have the \_\_init\_\_ method.

Here is an example of how to use it.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | class Employee:  def \_\_init\_\_(self, name, age,salary):  self.name = name  self.age = age  self.salary = 20000  E1 = Employee("XYZ", 23, 20000)  # E1 is the instance of class Employee.  #\_\_init\_\_ allocates memory for E1.  print(E1.name)  print(E1.age)  print(E1.salary) |

**Output:**

XYZ

**What is a lambda function?**

*Ans:* An anonymous function is known as a lambda function. This function can have any number of parameters but, can have just one statement.

**Example:**

|  |  |
| --- | --- |
| 1  2 | a = lambda x,y : x+y  print(a(5, 6)) |

**Output:**11

**What does [::-1} do?**

*Ans:* [::-1] is used to reverse the order of an array or a sequence.

*For example:*

|  |  |
| --- | --- |
| 1  2  3 | import array as arr  My\_Array=arr.array('i',[1,2,3,4,5])  My\_Array[::-1] |

**Output**: array(‘i’, [5, 4, 3, 2, 1])

[::-1] reprints a reversed copy of ordered data structures such as an array or a list. the original array or list remains unchanged.

**What is the difference between range & xrange?**

**Ans:** For the most part, xrange and range are the exact same in terms of functionality. They both provide a way to generate a list of integers for you to use, however you please. The only difference is that range returns a Python list object and x range returns an xrange object.

This means that xrange doesn’t actually generate a static list at run-time like range does. It creates the values as you need them with a special technique called yielding. This technique is used with a type of object known as generators. That means that if you have a really gigantic range you’d like to generate a list for, say one billion, xrange is the function to use.

This is especially true if you have a really memory sensitive system such as a cell phone that you are working with, as range will use as much memory as it can to create your array of integers, which can result in a Memory Error and crash your program. It’s a memory hungry beast.

**What is pickling and unpickling?**

**Ans:** Pickle module accepts any Python object and converts it into a string representation and dumps it into a file by using dump function, this process is called pickling. While the process of retrieving original Python objects from the stored string representation is called unpickling.

**What are the generators in python?**

*Ans:* Functions that return an iterable set of items are called generators.

**How will you convert a string to all lowercase?**

*Ans:* To convert a string to lowercase, lower() function can be used.

**Example:**

|  |  |
| --- | --- |
| 1  2 | stg='ABCD'  print(stg.lower()) |

**Output:** abcd

**What does this mean: \*args, \*\*kwargs? And why would we use it?**

**Ans:** We use \*args when we aren’t sure how many arguments are going to be passed to a function, or if we want to pass a stored list or tuple of arguments to a function. \*\*kwargs is used when we don’t know how many keyword arguments will be passed to a function, or it can be used to pass the values of a dictionary as keyword arguments. The identifiers args and kwargs are a convention, you could also use \*bob and \*\*billy but that would not be wise.

**What is the difference between deep and shallow copy?**

*Ans: Shallow copy* is used when a new instance type gets created and it keeps the values that are copied in the new instance. Shallow copy is used to copy the reference pointers just like it copies the values. These references point to the original objects and the changes made in any member of the class will also affect the original copy of it. Shallow copy allows faster execution of the program and it depends on the size of the data that is used.

*Deep copy* is used to store the values that are already copied. Deep copy doesn’t copy the reference pointers to the objects. It makes the reference to an object and the new object that is pointed by some other object gets stored. The changes made in the original copy won’t affect any other copy that uses the object. Deep copy makes execution of the program slower due to making certain copies for each object that is been called.

**What are Python libraries? Name a few of them.**

Python libraries are a collection of Python packages. Some of the majorly used python libraries are – [Numpy](https://www.edureka.co/blog/python-numpy-tutorial/), [Pandas](https://www.edureka.co/blog/python-pandas-tutorial/), [Matplotlib](https://www.edureka.co/blog/python-matplotlib-tutorial/), [Scikit-learn](https://www.edureka.co/blog/scikit-learn-machine-learning/) and many more.

**How to import modules in python?**

Modules can be imported using the **import**keyword.  You can import modules in three ways-

**Example:**

|  |  |
| --- | --- |
| 1  2  3 | import array           #importing using the original module name  import array as arr    # importing using an alias name  from array import \*    #imports everything present in the array module |

**What is monkey patching in Python?**

**Ans:** In Python, the term monkey patch only refers to dynamic modifications of a class or module at run-time.

Consider the below example:

|  |  |
| --- | --- |
| 1  2  3  4 | # m.py  class MyClass:  def f(self):  print "f()" |

We can then run the monkey-patch testing like this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | import m  def monkey\_f(self):  print "monkey\_f()"    m.MyClass.f = monkey\_f  obj = m.MyClass()  obj.f() |

The output will be as below:

monkey\_f()

As we can see, we did make some changes in the behavior of*f()* in *MyClass*using the function we defined, *monkey\_f()*, outside of the module *m*.

**What is map function in Python?**

**Ans:** *map* function executes the function given as the first argument on all the elements of the iterable given as the second argument. If the function given takes in more than 1 arguments, then many iterables are given. #Follow the link to know more similar functions.

**Differentiate between lists and tuples.**

The major difference is that a list is mutable, but a tuple is immutable.

**Explain help() and dir() functions in Python.**

The help() function displays the documentation string and help for its argument.

**What is slicing?**

Slicing is a technique that allows us to retrieve only a part of a list, tuple, or string. For this, we use the slicing operator [].

**How do you reverse a list?** Using reverse() method

**What is list comprehension? Example**

**When is the else part of a try-except block executed?**

In an if-else block, the else part is executed when the condition in the if-statement is False. But with a try-except block, the else part executes only if no exception is raised in the try part.

**Explain try, raise, and finally.**

These are the keywords we use with [*exception-handling*](https://data-flair.training/blogs/python-exception-handling/). We put risky code under a try block, use the raise statement to explicitly raise an error, and use the finally block to put code that we want to execute anyway.

**What is a generator?**

Python generator produces a sequence of values to iterate on. This way, it is kind of an iterable. We define a function that ‘yields’ values one by one, and then use a for loop to iterate on it.

def squares(n):

i=1

while(i<=n):

yield i\*\*2

i+=1

for i in squares(7):

print(i)

Output:

**1** **4** **9** **16** **25** **36** **49**

**Which one of these is floor division?**

**a) / b) // c) % d) None of the mentioned**

**Answer:** b) //

**Suppose list1 is [2, 33, 222, 14, 25], What is list1[-1] ?**

**a) Error b) None c) 25 d) 2**

**Answer:** c) 25

**Write a program in Python to check if a number is prime.**

**Write a program in Python to check if a sequence is a Palindrome.**

**Write a program to produce Fibonacci series in Python.**