Department of Computer Engineering

Important

22616 PWP MCQ (Programming With Python)

6th Sem all subject MCQs: click here
Download pdfs: click here

22616 PWP MCQ (Important MCQ for Summer 2021 Exam) Download pdf of 22616 PWP MCQ, The following 101,102 pdf are important for the Exam so keep that in your head. Please go through all the MCQ. Below pdf is released by cwipedia and the credits go to a different entity.

https://www.pythonprogramming.in/object-oriented-programming.html

Python Functions

1

Question 1

What will be the output of the following code :

```
print type(type(int))
```

A type 'int'

B type 'type'

C Error

D 0

Ans: B

Question 2

What is the output of the following code :

```
L = ['a','b','c','d']
print "".join(L)
```

A Error

B None

C abcd

D [‘a’,’b’,’c’,’d’]

Ans: **(C)**

Explanation: “” depicts a null string and the join function combines the elements of the list into a string.

Question 3

What is the output of the following segment :

```
chr(ord('A'))
```

A A

B B

C a

D Error

Ans: **(A)**

Explanation: ord() function converts a character into its ASCII notation and chr() converts the ASCII to character.

Question 4

What is the output of the following program :

```
y = 8
z = lambda x : x * y
print z(6)
```

A 48

B 14

C 64

D None of the above

Ans: **(A)**

Explanation: lambdas are concise functions and thus, result = 6 * 8

Question 5

What is called when a function is defined inside a class?

A Module

B Class

C Another Function

D Method

Ans: **(D)**

Question 6

Which of the following is the use of id() function in python?

A Id returns the identity of the object

B Every object doesn't have a unique id

C All of the mentioned

D None of the mentioned

Ans: **(A)**

Explanation: Each object in Python has a unique id. The id() function returns the object's id.

Question 7

What is the output of the following program :

```
import re
sentence = 'horses are fast'
regex = re.compile('(?P<animal>w+) (?P<verb>w+) (?P<adjective>w+)')
matched = re.search(regex, sentence)
print(matched.groupdict())
```

A {'animal': 'horses', 'verb': 'are', 'adjective': 'fast'}

B ('horses', 'are', 'fast')

C 'horses are fast'

D 'are'

Ans: **(A)**

Explanation: This function returns a dictionary that contains all the matches.

Question 8

Suppose list1 is [3, 4, 5, 20, 5, 25, 1, 3], what is list1 after list1.pop(1)?

A [3, 4, 5, 20, 5, 25, 1, 3]

B [1, 3, 3, 4, 5, 5, 20, 25]

C [3, 5, 20, 5, 25, 1, 3]

D [1, 3, 4, 5, 20, 5, 25]
Ans: **(C)**

Explanation: pop(i) removes the ith index element from the list

Question 9
time.time() returns ________
A the current time
B the current time in milliseconds
C the current time in milliseconds since midnight
D the current time in milliseconds since midnight, January 1, 1970
E the current time in milliseconds since midnight, January 1, 1970 GMT (the Unix time)
Ans: (E)

Question 10
Consider the results of a medical experiment that aims to predict whether someone is going to develop myopia based on some physical measurements and heredity. In this case, the input dataset consists of the person's medical characteristics and the target variable is binary: 1 for those who are likely to develop myopia and 0 for those who aren't. This can be best classified as

A	Regression
B	Decision Tree
C	Clustering
D	Association Rules

Ans: (B)

Explanation: Regression: It is a statistical analysis which is used to establish relation between a response and a predictor variable. It is mainly used in finance related applications.
Decision Tree: Decision tree is a computational method which works on descriptive data and records the observations of each object to reach to a result.
Clustering: It is a method of grouping more similar objects in a group and the non-similar objects to other groups.
Association Rules: It uses if-then reasoning method using the support-confidence technique to give a result.
According to the question Decision Tree is the most suitable technique that can be used to get best result of the experiment.

Question 1
What is the output of the following code :

```
print 9//2
```

A 4.5
B 4.0
C 4
D Error

Ans: C

Question 2

Which function overloads the >> operator?

A	more()
B	gt()
C	ge()
D	None of the above

(D)

Explanation: rshift() overloads the >> operatorQuestion 3

Which operator is overloaded by the or() function?

A ||

B |

C //

D /

Ans: (B)

Explanation: or() function overloads the bitwise OR operator

Question 4

What is the output of the following program :

```
i = 0
while i < 3:
        print i
        i++
        print i+1
```

A 0 2 1 3 2 4

B 0 1 2 3 4 5

C Error

D 1 0 2 4 3 5

Ans: (C)

Explanation: There is no operator ++ in Python

Question 1

What is the output of the following program :

```
print "Hello World"[::-1]
```

A dlroW olleH

B Hello Worl

C d

D Error

Ans: (A)

Explanation: [::] depicts extended slicing in Python and [::-1] returns the reverse of the string.

Question 2

Given a function that does not return any value, what value is shown when executed at the shell?

A int

B bool

C void

D None

Ans: (D)

Explanation: Python explicitly defines the None object that is returned if no value is specified.

Question 3

Which module in Python supports regular expressions?

A re

B regex

C pyregex

D None of the above

Ans: (A)

Explanation: re is a part of the standard library and can be imported using: import re.

Question 4

What is the output of the following program :

```
print 0.1 + 0.2 == 0.3
```

A True

B False

C Machine dependent

D Error

Ans: Answer: (B)

Explanation: Neither of 0.1, 0.2 and 0.3 can be represented accurately in binary. The round off errors from 0.1 and 0.2 accumulate and hence there is a difference of 5.5511e-17 between (0.1 + 0.2) and 0.3.

Question 5

Which of the following is not a complex number?

A k = 2 + 3j

B k = complex(2, 3)

C k = 2 + 3l

D k = 2 + 3J

Ans: (C)

Explanation: l (or L) stands for long.

Question 6
What does ~~~~~~5 evaluate to?
A +5
B -11
C +11
D -5
Ans: (A)

Explanation: ~x is equivalent to -(x+1).

Question 7
Given a string s = “Welcome”, which of the following code is incorrect?
A print s[0]
B print s.lower()
C s[1] = ‘r’
D print s.strip()
Ans: (C)

Explanation: strings are immutable in Python

Question 8
________ is a simple but incomplete version of a function.
A Stub
B Function
C A function developed using bottom-up approach
D A function developed using top-down approach
Ans: (A)

Question 9
To start Python from the command prompt, use the command ______
A execute python
B go python
C python
D run python
Ans: (C)

Question 10
Which of the following is correct about Python?
A It supports automatic garbage collection.
B It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java
C Both of the above

D None of the above
Ans: **(C)**

Question 1
Which of these is not a core data type?
A Lists
B Dictionary
C Tuples
D Class
Ans: (D)

Explanation: Class is a user defined data type

Question 2
What data type is the object below ? L = [1, 23, ‘hello’, 1]
A List
B Dictionary
C Tuple
D Array
Ans: (A)

Explanation: [ ] defines a list

Question 3
Which of the following function convert a string to a float in python?
A int(x [,base])
B long(x [,base])
C float(x)
D str(x)
Ans: (C)

Explanation: float(x) – Converts x to a floating-point number

Question 4
Which of the following statement(s) is TRUE?

1. A hash function takes a message of arbitrary length and generates a fixed length code.
2. A hash function takes a message of fixed length and generates a code of variable length.
3. A hash function may give the same hash value for distinct messages.

A I only
B II and III only
C I and III only

D II only

Ans: (C)

Explanation:

Hash function is defined as any function that can be used to map data of arbitrary size of data to a fixed size data.. The values returned by a hash function are called hash values, hash codes, digests, or simply hashes : Statement 1 is correct
Yes, it is possible that a Hash Function maps a value to a same location in the memmory that’s why collision occurs and we have different technique to handle this problem : Statement 3 is coorect.
eg : we have hash function, h(x) = x mod 3

Acc to Statement 1, no matter what the value of ‘x’ is h(x) results in a fixed mapping location.
Acc. to Statement 3, h(x) can result in same mapping mapping location for different value of ‘x’ e.g. if x = 4 or x = 7 , h(x) = 1 in both the cases, although collision occurs.

What is the output of the following program :

```
def myfunc(a):
    a = a + 2
        a = a * 2
    return a

print myfunc(2)
```

A 8

B 16

C Indentation Error

D Runtime Error

Ans: (C)

Explanation: Python creates blocks of code based on the indentation of the code. Thus, new indent defines a new scope.

Question 2
What is the output of the expression : 3*1**3

A 27

B 9

C 3

D 1

Ans: (C)

Explanation: Precedence of ** is higher than that of 3, thus first 1**3 will be executed and the result will be multiplied by 3.

Question 3

What is the output of the following program :

```
print '{0:.2}'.format(1.0 / 3)
```

A 0.333333

B 0.33

C 0.333333:-2

D Error

Ans: (B)

Explanation: .2 defines the precision of the floating point number.

Question 4

What is the output of the following program :

```
print '{0:-2%}'.format(1.0 / 3)
```

A 0.33

B 0.33%

C 33.33%

D 33%

Ans: (C)

Explanation: The % converts the 0.33 to percentage with respect to 1.0

Question 5

What is the output of the following program :

```
i = 0
while i < 3:
    print i
    i += 1
else:
    print 0
```

A 0 1 2 3 0

B 0 1 2 0

C 0 1 2

D Error

Ans: (B)

Explanation: The else part is executed when the condition in the while statement is false.

Question 6

What is the output of the following program :

```
i = 0
while i < 5:
    print(i)
    i += 1
    if i == 3:
        break
else:
    print(0)
```

A 0 1 2 0

B 0 1 2

C Error

D None of the above

Ans: (B)

Explanation: The else part is not executed if control breaks out of the loop.

Question 7

What is the output of the following program :

```
print 'cd'.partition('cd')
```

A (‘cd’)

B (”)

C (‘cd’, ”, ”)

D (”, ‘cd’, ”)

Ans: (D)

Explanation: The entire string has been passed as the separator hence the first and the last item of the tuple returned are null strings.

Question 8

What is the output of the following program :

```
print 'abef'.partition('cd')
```

A (‘abef’)

B (‘abef’, ‘cd’, ”)

C (‘abef’, ”, ”)

D Error

Ans: (C)

Explanation: The separator is not present in the string hence the second and the third elements of the tuple are null strings.

Question 9

What is the output of the following program :

```
print 'abcefd'.replace('cd', '12')
```

A ab1ef2

B abcefd

C ab1efd

D ab12ed2

Ans: (B)

Explanation: The first substring is not present in the given string and hence nothing is replaced.

Question 10

What will be displayed by the following code?

```
def f(value, values):
     v = 1
     values[0] = 44
```

```
t = 3
v = [1, 2, 3]
f(t, v)
print(t, v[0])
```

A 1 1

B 1 44

C 3 1

D 3 44

(D)

Explanation: The value of t=3 is passed in funcion f(value,values) , v [list] is passed as values in the same function. The v is stored in values and values[0]=44 , changes the value at index['0'] in the list hence v=[44,2,3].

What is the output of the following code? Consider Python 2.7.

```
print tuple[1:3] if tuple == ( 'abcd', 786 , 2.23, 'john', 70.2 ) else
tuple()
```

A ('abcd', 786 , 2.23, 'john', 70.2)

B abcd

C (786, 2.23)

D None of the above

(D)

Predict the output of following python programs:

Program 1:

```
r = lambda q: q * 2
s = lambda q: q * 3
x = 2
x = r(x)
x = s(x)
x = r(x)
print x
```

Output:

```
24
```

Explanation : In the above program r and s are lambda functions or anonymous functions and q is the argument to both of the functions. In first step we have initialized x to 2. In second step we have passed x as argument to the lambda function r, this will return x*2 which is stored in x. That is, x = 4 now. Similarly in third step we have passed x to lambda function s, So x = 4*3. i.e, x = 12 now. Again in the last step, x is multiplied by 2 by passing it to function r. Therefore, x = 24.

Program 2:

```
a = 4.5
b = 2
print a//b
```

Output:

```
2.0
```

Explanation : This type of division is called truncating division where the remainder is truncated or dropped.

Program 3:

```
a = True
b = False
c = False

if a or b and c:
    print "GEEKSFORGEEKS"
else:
    print "geeksforgeeks"
```

Output:

```
GEEKSFORGEEKS
```

Explanation : In Python, AND operator has higher precedence than OR operator. So, it is evaluated first. i.e, (b and c) evaluates to false.Now OR operator is evaluated. Here, (True or False) evaluates to True. So the if condition becomes True and GEEKSFORGEEKS is printed as output.

Program 4:

```
a = True
b = False
c = False

if not a or b:
    print 1
elif not a or not b and c:
    print 2
elif not a or b or not b and a:
    print 3
else:
    print 4
```

Output:

```
3
```

Explanation: In Python the precedence order is first NOT then AND and in last OR. So the if condition and second elif condition evaluates to False while third elif condition is evaluated to be True resulting in 3 as output.

Program 5:

```
count = 1

def doThis():

    global count

    for i in (1, 2, 3):
        count += 1

doThis()

print count
```

Output:

```
 4
```

Explanation: The variable count declared outside the function is global variable and also the count variable being referenced in the function is the same global variable defined outside of the function. So, the changes made to variable in the function is reflected to the original variable. So, the output of the program is 4.

Predict the output of following Python Programs.

Program 1:

```
class Acc:
    def __init__(self, id):
        self.id = id
        id = 555

acc = Acc(111)
print acc.id
```

Output:

```
111
```

Explanation: Instantiation of the class "Acc" automatically calls the method __init__ and passes the object as the self parameter. 111 is assigned to data attribute of the object called id. The value "555" is not retained in the object as it is not assigned to a data attribute of the class/object. So, the output of the program is "111"

Program 2:

```
for i in range(2):
    print i
```

```
for i in range(4,6):
    print i
```

Output:

```
0
1
4
5
```

Explanation: If only single argument is passed to the range method, Python considers this argument as the end of the range and the default start value of range is 0. So, it will print all the numbers starting from 0 and before the supplied argument.
For the second for loop the starting value is explicitly supplied as 4 and ending is 5.

Program 3:

```
values = [1, 2, 3, 4]
numbers = set(values)

def checknums(num):
    if num in numbers:
        return True
    else:
        return False

for i in filter(checknums, values):
    print i
```

Output:

```
1
2
3
4
```

Explanation: The function “filter” will return all items from list values which return True when passed to the function “checkit”. “checkit” will check if the value is in the set. Since all the numbers in the set come from the values list, all of the original values in the list will return True.

Program 4:

```
counter = {}

def addToCounter(country):
    if country in counter:
        counter[country] += 1
    else:
        counter[country] = 1

addToCounter('China')
addToCounter('Japan')
addToCounter('china')

print len(counter)
```

Output:

```
3
```

Explanation: The task of "len" function is to return number of keys in a dictionary. Here 3 keys are added to the dictionary "country" using the "addToCounter" function.
Please note carefully – The keys to a dictionary are **case sensitive.**
Try yourself: What happens If same key is passed twice??

Predict the output of following Python Programs.

Program 1:

```
class Geeks:
    def __init__(self, id):
        self.id = id

manager = Geeks(100)

manager.__dict__['life'] = 49

print manager.life + len(manager.__dict__)
```

Output:

```
51
```

Explanation : In the above program we are creating a member variable having name 'life' by adding it directly to the dictionary of the object 'manager' of class 'Geeks'. Total numbers of items in the dictionary is 2, the variables 'life' and 'id'. Therefore the size or the length of the dictionary is 2 and the variable 'life' is assigned a value '49'. So the sum of the variable 'life' and the size of the dictionary is 49 + 2 = 51.

Program 2:

```
a = "GeeksforGeeks "

b = 13

print a + b
```

Output:

```
An error is shown.
```

Explanation : As you can see the variable ‘b’ is of type integer and the variable ‘a’ is of type string. Also as Python is a strongly typed language we cannot simply concatenate an integer with a string. We have to first convert the integer variable to the type string to concatenate it with a string variable. So, trying to concatenate an integer variable to a string variable, an exception of type “TypeError” is occurred.

Program 3:

```
dictionary = {}
dictionary[1] = 1
dictionary['1'] = 2
dictionary[1] += 1

sum = 0
for k in dictionary:
    sum += dictionary[k]

print sum
```

Output:

```
4
```

Explanation : In the above dictionary, the key 1 enclosed between single quotes and only 1 represents two different keys as one of them is integer and other is string. So, the output of the program is 4.

Predict the output of following Python Programs.

Program 1:

```
nameList = ['Harsh', 'Pratik', 'Bob', 'Dhruv']

print nameList[1][-1]
```

Output:`k`

Explanation:
The index position -1 represents either the last element in a list or the last character in a String. In the above given list of names “nameList”, the index 1 represents the second element i.e, the second string “Pratik” and the index -1 represents the last character in the string “Pratik”. So, the output is “k”.

Program 2:

```
nameList = ['Harsh', 'Pratik', 'Bob', 'Dhruv']

pos = nameList.index("GeeksforGeeks")

print pos * 5
```

Output:

```
An Exception is thrown, ValueError: 'GeeksforGeeks' is not in list
```

Explanation:
The task of index is to find the position of a supplied value in a given list. In the above program the supplied value is “GeeksforGeeks” and list is nameList. As GeeksforGeeks is not present in the list, an exception is thrown.

Program 3:

```
geekCodes = [1, 2, 3, 4]

geekCodes.append([5,6,7,8])

print len(geekCodes)
```

Output:

```
5
```

Explanation:
The task of append() method is to append a passed *obj* into an existing list. But instead of passing a list to the append method will not merge the two lists, the entire list which is passed is added as an element of the list. So the output is 5.

Program 4:

```
def addToList(listcontainer):
    listcontainer += [10]

mylistContainer = [10, 20, 30, 40]
addToList(mylistContainer)
print len(mylistContainer)
```

Output:

```
5
```

Explanation:
In Python, everything is a reference and references are passed by value. Parameter passing in Python is same as reference passing in Java. As a consequence, the function can modify the value referred by passed argument, i.e. the value of the variable in the caller’s scope can be changed. Here the task of the function “addToList” is to add an element 10 in the list, So this will increase the length of list by 1. So the output of program is 5.

Program 1:

```
def gfgFunction():
    "Geeksforgeeks is cool website for boosting up technical skills"
    return 1

print gfgFunction.__doc__[17:21]
```

Output:`cool`

Explanation:
There is a docstring defined for this method, by putting a string on the first line after the start of the function definition. The docstring can be referenced using the __doc__ attribute of the function.
And hence it prints the indexed string.

Program 2:

```
class A(object):
    val = 1

class B(A):
    pass

class C(A):
    pass

print A.val, B.val, C.val
B.val = 2
print A.val, B.val, C.val
A.val = 3
print A.val, B.val, C.val
```

Output:

```
1 1 1
1 2 1
3 2 3
```

Explanation:
In Python, class variables are internally handled as dictionaries. If a variable name is not found in the dictionary of the current class, the class hierarchy (i.e., its parent classes) are searched until the referenced variable name is found, if the variable is not found error is being thrown.
So, in the above program the first call to print() prints the initialized value i.e, 1.
In the second call since B. val is set to 2, the output is 1 2 1.
The last output 3 2 3 may be surprising. Instead of 3 3 3, here B.val reflects 2 instead of 3 since it is overridden earlier.

Program 3:

```
check1 = ['Learn', 'Quiz', 'Practice', 'Contribute']
check2 = check1
check3 = check1[:]
```

```
check2[0] = 'Code'
check3[1] = 'Mcq'

count = 0
for c in (check1, check2, check3):
    if c[0] == 'Code':
        count += 1
    if c[1] == 'Mcq':
        count += 10

print count
```

Output:

```
12
```

Explanation:
When assigning check1 to check2, we create a second reference to the same list. Changes to check2 affects check1. When assigning the slice of all elements in check1 to check3, we are creating a full copy of check1 which can be modified independently (i.e, any change in check3 will not affect check1).
So, while checking check1 'Code' gets matched and count increases to 1, but Mcq doest gets matched since its available only in check3.
Now checking check2 here also 'Code' gets matched resulting in count value to 2.
Finally while checking check3 which is separate than both check1 and check2 here only Mcq gets matched and count becomes 12.

Program 4:

```
def gfg(x,l=[]):
    for i in range(x):
        l.append(i*i)
    print(l)

gfg(2)
gfg(3,[3,2,1])
gfg(3)
```

Output:

```
[0, 1]
[3, 2, 1, 0, 1, 4]
[0, 1, 0, 1, 4]
```

Explanation:
The first function call should be fairly obvious, the loop appends 0 and then 1 to the empty list, l. l is a name for a variable that points to a list stored in memory. The second call starts off by creating a new list in a new block of memory. l then refers to this new list. It then appends 0, 1 and 4 to this new list. So that's great. The third function call is the weird one. It uses the original list stored in the original memory block. That is why it starts off with 0 and 1.

Predict the output of the following Python programs. These question set will make you conversant with List Concepts in Python programming language.

- **Program 1**

```
list1 = ['physics', 'chemistry', 1997, 2000]

list2 = [1, 2, 3, 4, 5, 6, 7 ]

print "list1[0]: ", list1[0]        #statement 1
print "list1[0]: ", list1[-2]       #statement 2
print "list1[-2]: ", list1[1:]      #statement 3
print "list2[1:5]: ", list2[1:5]    #statement 4
```

Output:

```
list1[0]:  physics
list1[0]:  1997
list1[-2]:  ['chemistry', 1997, 2000]
list2[1:5]:  [2, 3, 4, 5]
```

Explanation:
To access values in lists, we use the square brackets for slicing along with the index or indices to obtain required value available at that index.For N items in a List MAX value of index will be N-1.
Statement 1 : This will print item located at index 0 in Output.
Statement 2 : This will print item located at index -2 i.e.second last element in Output.
Statement 3 : This will print items located from index 1 to end of the list.
Statement 4 : This will print items located from index 1 to 4 of the list.

- **Program 2**

```
list1 = ['physics', 'chemistry', 1997, 2000]

print "list1[1][1]: ", list1[1][1] #statement 1

print "list1[1][-1]: ", list1[1][-1] #statement 2
```

Output:

```
list1[1][1]:  h
list1[1][-1]:  y
```

Explanation:
In python we can slice a list but we can also slice a element within list if it is a string. The declaration list[x][y] will mean that 'x' is the index of element within a list and 'y' is the index of entity within that string.

- **Program 3**

```
list1 = [1998, 2002, 1997, 2000]
list2 = [2014, 2016, 1996, 2009]
```

```
print "list1 + list 2 = : ", list1 + list2   #statement 1

print "list1 * 2 = : ", list1 * 2 #statement 2
```

Output:

```
list1 + list 2 = :  [1998, 2002, 1997, 2000, 2014, 2016, 1996, 2009]
list1 * 2 = :  [1998, 2002, 1997, 2000, 1998, 2002, 1997, 2000]
```

Explanation:
When addition(+) operator uses list as its operands then the two lists will get concatenated. And when a list id multiplied with a constant k>=0 then the same list is appended k times in the original list.

- **Program 4**

```
list1 = range(100, 110) #statement 1
print "index of element 105 is : ", list1.index(105)  #statement 2
```

Output:

```
index of element 105 is :  5
```

Explanation:
Statement 1 : will genetrate numbers from 100 to 110 and appent all these numbers in the list.
Statement 2 : will give the index value of 105 in the list list1.

- **Program 5**

```
list1 = [1, 2, 3, 4, 5]
list2 = list1

list2[0] = 0;

print "list1= : ", list1 #statement 2
```

Output:

```
list1= :  [0, 2, 3, 4, 5]
```

Explanation:
In this problem, we have provided a reference to the list1 with another name list2 but these two lists are same which have two references(list1 and list2). So any alteration with list2 will affect the original list.

Predict the output of the following Python programs. These question set will make you conversant with String Concepts in Python programming language.

- **Program 1**

```
var1 = 'Hello Geeks!'
var2 = "GeeksforGeeks"

print "var1[0]: ", var1[0]        # statement 1
print "var2[1:5]: ", var2[1:5]    # statement 2
```

Output:

```
var1[0]:  H
var2[1:5]:  eeks
```

Explanation:
Strings are among the most popular types in Python. We can create a string by enclosing characters within quotes. **Python treats single quotes same as double quotes.** It is notable that unlike C or C++ python does not support a character type; in fact single characters are treated as strings of length one, thus also considered a substring. To access substrings, use the square brackets for slicing along with the index or indices to obtain your substring.
Statement 1: will simply put character at 0 index on the output screen.
Statement 2: will put the character starting from 0 index to index 4.

- **Program 2**

```
var1 = 'Geeks'

print "Original String :-", var1

print "Updated String :- ", var1[:5] + 'for' + 'Geeks' # statement 1
```

Output:

```
Original String :- Geeks
Updated String :-  GeeksforGeeks
```

Explanation:
Python provides a flexible way to update string in your code. Use square brackets and specify the index from where string has to be updated and use + operator to append the string. [x:y] operator is called Range Slice and gives the characters from the given range.

Statement 1: In the given code tells the interpreter that from 5th index of string present in var1 append ‘for’ and ‘Geeks’ to it.

- **Program 3**

```
para_str = """this is a long string that is made up of
```

```
several lines and non-printable characters such as
TAB ( \t ) and they will show up that way when displayed.
NEWLINEs within the string, whether explicitly given like
this within the brackets [ \n ], or just a NEWLINE within
the variable assignment will also show up.
"""
print para_str
```

Output:

```
this is a long string that is made up of
several lines and non-printable characters such as
TAB (      ) and they will show up that way when displayed.
NEWLINEs within the string, whether explicitly given like
this within the brackets [
 ], or just a NEWLINE within
the variable assignment will also show up.
```

Explanation:
Python's triple quotes comes to the rescue by allowing strings to span multiple lines, including NEWLINEs, TABs, and any other special characters.The syntax for triple quotes consists of three consecutive single or double quotes.

- **Program 4**

```
print 'C:\\inside C directory' # statement1

print r'C:\\inside C directory' # statement2
```

Output:

```
C:\inside C directory
C:\\inside C directory
```

Explanation:
Raw strings do not treat the backslash as special characters at all.
Statement 1 : will print the message while considering backslash as a special character.
Statement 2 : is a raw string that will treat backslash as a normal character.

- **Program 5**

```
print '\x25\x26'
```

Output:

```
%&
```

Explanation:
In the above code \x is an escape sequence that means the following 2 digits are a hexadecimal number encoding a character. Hence the corresponding symbols will be on the output screen.

Predict the output of the following Python programs.

- **Program 1**

  ```
  list = [1, 2, 3, None, (1, 2, 3, 4, 5), ['Geeks', 'for', 'Geeks']]
  print len(list)
  ```

  Output:

  ```
  6
  ```

  Explanation:
  The beauty of python list datatype is that within a list, a programmer can nest another list, a dictionary or a tuple. Since in the code there are 6 items present in the list the length of the list is 6.

- **Program 2**

  ```
  list = ['python', 'learning', '@', 'Geeks', 'for', 'Geeks']

  print list[::]
  print list[0:6:2]
  print list[ :6: ]
  print list[ :6:2]
  print list[ ::3]
  print list[ ::-2]
  ```

  Output:

  ```
  ['python', 'learning', '@', 'Geeks', 'for', 'Geeks']
  ['python', '@', 'for']
  ['python', 'learning', '@', 'Geeks', 'for', 'Geeks']
  ['python', '@', 'for']
  ['python', 'Geeks']
  ['Geeks', 'Geeks', 'learning']
  ```

  Explanation:
  In python list slicing can also be done by using the syntax listName[x:y:z] where x means the initial index, y-1 defines the final index value and z specifies the step size. If anyone of the values among x, y and z is missing the interpreter takes default value.

  Note:
  1. For x default value is 0 i.e. start of the list.
  2. For y default value is length of the list.
  3. For z default value is 1 i.e. every element of the list.

- **Program 3**

  ```
  d1 = [10, 20, 30, 40, 50]
  d2 = [1, 2, 3, 4, 5]
  print d1 - d1
  ```

  Output:

```
No Output
```

Explanation:
Unlike additon or relational operators not all the artihmatic operaters can use lists as their operands. Since – minus operator can't take lists as its operand no output will be produced. Program will produce following error.

```
TypeError: unsupported operand type(s) for -: 'list' and 'list'
```

- **Program 4**

```
list = ['a', 'b', 'c', 'd', 'e']
print list[10:]
```

Output:

```
[]
```

Explanation:
As one would expect, attempting to access a member of a list using an index that exceeds the number of members (e.g., attempting to access list[10] in the list above) results in an IndexError. However, attempting to access a slice of a list at a starting index that exceeds the number of members in the list will not result in an IndexError and will simply return an empty list.

- **Program 5**

```
list = ['a', 'b', 'c']*-3
print list
```

Output:

```
[]
```

Explanation:
A expression list[listelements]*N where N is a integer appends N copies of list elements in the original list. If N is a negetive integer or 0 output will be a empty list else if N is positive list elements will be added N times to the original list.

1) What is the output of the following program?

filter_none

edit

play_arrow

brightness_4

```
dictionary = {'GFG' : 'geeksforgeeks.org',
              'google' : 'google.com',
              'facebook' : 'facebook.com'
```

```
               }
del dictionary['google'];
for key, values in dictionary.items():
    print(key)
dictionary.clear();
for key, values in dictionary.items():
    print(key)
del dictionary;
for key, values in dictionary.items():
    print(key)
```

a) Both b and d
b) Runtime error
c) GFG
facebook
d) facebook
GFG

Ans. (a)
Output:

```
facebook
GFG
```

Explanation: The statement: **del dictionary;** removes the entire dictionary, so iterating over a deleted dictionary throws a runtime error as follows:

```
Traceback (most recent call last):
  File "cbeac2f0e35485f19ae7c07f6b416e84.py", line 12, in
    for key, values in dictionary.items():
NameError: name 'dictionary' is not defined
```

2) What is the output of the following program?

filter_none

edit

play_arrow

brightness_4

```
dictionary1 = {'Google' : 1,
               'Facebook' : 2,
               'Microsoft' : 3
               }
dictionary2 = {'GFG' : 1,
               'Microsoft' : 2,
               'Youtube' : 3
               }
dictionary1.update(dictionary2);
for key, values in dictionary1.items():
    print(key, values)
```

a) Compilation error
b) Runtime error
c) ('Google', 1)
('Facebook', 2)
('Youtube', 3)
('Microsoft', 2)
('GFG', 1)
d) None of these

Ans. (c)
Explanation: dictionary1.update(dictionary2) is used to update the entries of dictionary1 with entries of dictionary2. If there are same keys in two dictionaries, then the value in second dictionary is used.

3) What is the output of the following program?

filter_none

edit

play_arrow

brightness_4

```
dictionary1 = {'GFG' : 1,
               'Google' : 2,
               'GFG' : 3
               }
print(dictionary1['GFG']);
```

a) Compilation error due to duplicate keys
b) Runtime time error due to duplicate keys
c) 3
d) 1

Ans. (c)
Explanation: Here, GFG is the duplicate key. Duplicate keys are not allowed in python. If there are same keys in a dictionay, then the **value assigned mostly recently** is assigned to the that key.

4) What is the output of the following program?

filter_none

edit

play_arrow

brightness_4

```
temp = dict()
```

```
temp['key1'] ={'key1' : 44, 'key2' : 566}
temp['key2'] = [1, 2, 3, 4]
for (key, values) in temp.items():
    print(values, end = "")
```

a) Compilation error
b) {‘key1’: 44, ‘key2’: 566}[1, 2, 3, 4]
c) Runtime error
d) None of the above

Ans. (b)
Explanation: A dictionary can hold any value such as an integer, string, list or even another dictionary holding key value pairs.
Note: This code can be executed only in python versions above 3

5) What is the output of the following program?

filter_none

edit

play_arrow

brightness_4

```
temp = {'GFG' : 1,
        'Facebook' : 2,
        'Google' : 3
        }
for (key, values) in temp.items():
    print(key, values, end = " ")
```

a) Google 3 GFG 1 Facebook 2
b) Facebook 2 GFG 1 Google 3
c) Facebook 2 Google 3 GFG 1
d) Any of the above

Ans. (d)
Explanation: Dictionaries are unordered. So any key value pairs can be added at any location within a dictionary. Any of the output may come.
Note: This code can be executed only in python versions above 3.

Question: Describe how multithreading is achieved in Python.

Answer: Even though Python comes with a multi-threading package, if the motive behind multithreading is to speed the code then using the package is not the go-to option.

The package has something called the GIL or Global Interpreter Lock, which is a construct. It ensures that one and only one of the threads execute at any given time. A thread acquires the GIL and then do some work before passing it to the next thread.

This happens so fast that to a user it seems that threads are executing in parallel. Obviously, this is not the case as they are just taking turns while using the same CPU core. Moreover, GIL passing adds to the overall overhead to the execution.

Hence, if you intend to use the threading package for speeding up the execution, using the package is not recommended.

Question: Draw a comparison between the range and xrange in Python.

Answer: In terms of functionality, both range and xrange are identical. Both allow for generating a list of integers. The main difference between the two is that while range returns a Python list object, xrange returns an xrange object.

Xrange is not able to generate a static list at runtime the way range does. On the contrary, it creates values along with the requirements via a special technique called yielding. It is used with a type of object known as generators.

If you have a very enormous range for which you need to generate a list, then xrange is the function to opt for. This is especially relevant for scenarios dealing with a memory-sensitive system, such as a smartphone.

The range is a memory beast. Using it requires much more memory, especially if the requirement is gigantic. Hence, in creating an array of integers to suit the needs, it can result in a Memory Error and ultimately lead to crashing the program.

Question: Explain Inheritance and its various types in Python?

Answer: Inheritance enables a class to acquire all the members of another class. These members can be attributes, methods, or both. By providing reusability, inheritance makes it easier to create as well as maintain an application.

The class which acquires is known as the child class or the derived class. The one that it acquires from is known as the superclass or base class or the parent class. There are 4 forms of inheritance supported by Python:

Interested in Python? Enquire for the best course suited for you.

- **Single Inheritance –** A single derived class acquires from on single superclass.
- **Multi-Level Inheritance –** At least 2 different derived classes acquire from two distinct base classes.
- **Hierarchical Inheritance –** A number of child classes acquire from one superclass
- **Multiple Inheritance –** A derived class acquires from several superclasses.

Question: Explain how is it possible to Get the Google cache age of any URL or webpage using Python.

Answer: In order to Get the Google cache age of any URL or webpage using Python, the following URL format is used:

http://webcache.googleusercontent.com/search?q=cache:URLGOESHERE

Simply replace URLGOESHERE with the web address of the website or webpage whose cache you need to retrieve and see in Python.

Question: Give a detailed explanation about setting up the database in Django.

Answer: The process of setting up a database is initiated by using the command edit mysite/setting.py. This is a normal Python module with a module-level representation of Django settings. Django relies on SQLite by default, which is easy to be used as it doesn’t require any other installation.

SQLite stores data as a single file in the filesystem. Now, you need to tell Django how to use the database. For this, the project’s setting.py file needs to be used. Following code must be added to the file for making the database workable with the Django project:

```
DATABASES = {
 'default': {
 'ENGINE' : 'django.db.backends.sqlite3',
 'NAME' : os.path.join(BASE_DIR, 'db.sqlite3'),
 }
}
```

If you need to use a database server other than the SQLite, such as MS SQL, MySQL, and PostgreSQL, then you need to use the database's administration tools to create a brand new database for your Django project.

You have to modify the following keys in the DATABASE 'default' item to make the new database work with the Django project:

- **ENGINE** – For example, when working with a MySQL database replace 'django.db.backends.sqlite3' with 'django.db.backends.mysql'
- **NAME** – Whether using SQLite or some other database management system, the database is typically a file on the system. The NAME should contain the full path to the file, including the name of that particular file.

NOTE: - Settings like Host, Password, and User needs to be added when not choosing SQLite as the database.

Check out the advantages and disadvantages of Django.

Question: How will you differentiate between deep copy and shallow copy?

Answer: We use shallow copy when a new instance type gets created. It keeps the values that are copied in the new instance. Just like it copies the values, the shallow copy also copies the reference pointers.

Reference points copied in the shallow copy reference to the original objects. Any changes made in any member of the class affects the original copy of the same. Shallow copy enables faster execution of the program.

Deep copy is used for storing values that are already copied. Unlike shallow copy, it doesn't copy the reference pointers to the objects. Deep copy makes the reference to an object in addition to storing the new object that is pointed by some other object.

Changes made to the original copy will not affect any other copy that makes use of the referenced or stored object. Contrary to the shallow copy, deep copy makes execution of a program slower. This is due to the fact that it makes some copies for each object that is called.

Question: How will you distinguish between NumPy and SciPy?

Answer: Typically, NumPy contains nothing but the array data type and the most basic operations, such as basic element-wise functions, indexing, reshaping, and sorting. All the numerical code resides in SciPy.

As one of NumPy's most important goals is compatibility, the library tries to retain all features supported by either of its predecessors. Hence, NumPy contains a few linear algebra functions despite the fact that these more appropriately belong to the SciPy library.

SciPy contains fully-featured versions of the linear algebra modules available to NumPy in addition to several other numerical algorithms.

Question: Observe the following code:

```
A0 = dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))
A1 = range(10)A2 = sorted([i for i in A1 if i in A0])
A3 = sorted([A0[s] for s in A0])
A4 = [i for i in A1 if i in A3]
A5 =
A6 = [[i,i*i] for i in A1]
print(A0,A1,A2,A3,A4,A5,A6)
```

Write down the output of the code.
Answer:

A0 = {'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4} # the order may vary
A1 = range(0, 10)
A2 = []
A3 = [1, 2, 3, 4, 5]
A4 = [1, 2, 3, 4, 5]
A5 =
A6 = [[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7, 49], [8, 64], [9, 81]]

Question: **Python has something called the dictionary. Explain using an example.**

Answer: A dictionary in Python programming language is an unordered collection of data values such as a map. Dictionary holds key:value pair. It helps in defining a one-to-one relationship between keys and values. Indexed by keys, a typical dictionary contains a pair of keys and corresponding values.

Let us take an example with three keys, namely Website, Language, and Offering. Their corresponding values are hackr.io, Python, and Tutorials. The code for the example will be:

```
dict={‘Website’:‘hackr.io’,‘Language’:‘Python’:‘Offering’:‘Tutorials’}
print dict[Website] #Prints hackr.io
print dict[Language] #Prints Python
print dict[Offering] #Prints Tutorials
```

Question: Python supports negative indexes. What are they and why are they used?

Answer: The sequences in Python are indexed. It consists of positive and negative numbers. Positive numbers use 0 as the first index, 1 as the second index, and so on. Hence, any index for a positive number n is n-1.

Unlike positive numbers, index numbering for the negative numbers start from -1 and it represents the last index in the sequence. Likewise, -2 represents the penultimate index. These are known as negative indexes. Negative indexes are used for:

- Removing any new-line spaces from the string, thus allowing the string to except the last character, represented as S[:-1]
- Showing the index to representing the string in the correct order

Question: Suppose you need to collect and print data from IMDb top 250 Movies page. Write a program in Python for doing so. (NOTE: - You can limit the displayed information for 3 fields; namely movie name, release year, and rating.)

Answer:

```
from bs4 import BeautifulSoup
import requests
import sys
url = 'http://www.imdb.com/chart/top'
response = requests.get(url)
soup = BeautifulSoup(response.text)
tr = soup.findChildren("tr")
tr = iter(tr)
next(tr)
for movie in tr:
title = movie.find('td', {'class': 'titleColumn'} ).find('a').contents[0]
year = movie.find('td', {'class': 'titleColumn'} ).find('span', {'class':
'secondaryInfo'}).contents[0]
rating = movie.find('td', {'class': 'ratingColumn imdbRating'}
).find('strong').contents[0]
row = title + ' - ' + year + ' ' + ' ' + rating
print(row)
```

Question: Take a look at the following code:

```
try: if '1' != 1:
raise "someError"
else: print("someError has not occured")
except "someError": pr
int ("someError has occured")
```

What will be the output?
Answer: The output of the program will be "invalid code." This is because a new exception class must inherit from a BaseException.

Question: What do you understand by monkey patching in Python?

Answer: The dynamic modifications made to a class or module at runtime are termed as monkey patching in Python. Consider the following code snippet:

```
# m.py
class MyClass:
def f(self):
print "f()"
```

We can monkey-patch the program something like this:

```
import m
def monkey_f(self):
print "monkey_f()"
m.MyClass.f = monkey_f
obj = m.MyClass()
obj.f()
```

Output for the program will be monkey_f().

The examples demonstrate changes made in the behavior of f() in MyClass using the function we defined i.e. monkey_f() outside of the module m.

Question: What do you understand by the process of compilation and linking in Python?

Answer: In order to compile new extensions without any error, compiling and linking is used in Python. Linking initiates only and only when the compilation is complete.

In the case of dynamic loading, the process of compilation and linking depends on the style that is provided with the concerned system. In order to provide dynamic loading of the configuration setup files and rebuilding the interpreter, the Python interpreter is used.

Question: What is Flask and what are the benefits of using it?

Answer: Flask is a web microframework for Python with Jinja2 and Werkzeug as its dependencies. As such, it has some notable advantages:

- Flask has little to no dependencies on external libraries
- Because there is a little external dependency to update and fewer security bugs, the web microframework is lightweight to use.
- Features an inbuilt development server and a fast debugger.

Question: What is the map() function used for in Python?

Answer: The map() function applies a given function to each item of an iterable. It then returns a list of the results. The value returned from the map() function can then be passed on to functions to the likes of the list() and set().

Typically, the given function is the first argument and the iterable is available as the second argument to a map() function. Several tables are given if the function takes in more than one arguments.

Question: What is Pickling and Unpickling in Python?

Answer: The Pickle module in Python allows accepting any object and then converting it into a string representation. It then dumps the same into a file by means of the dump function. This process is known as pickling.

The reverse process of pickling is known as unpickling i.e. retrieving original Python objects from a stored string representation.

Question: Whenever Python exits, all the memory isn't deallocated. Why is it so?

Answer: Upon exiting, Python's built-in effective cleanup mechanism comes into play and try to deallocate or destroy every other object.

However, Python modules that are having circular references to other objects or the objects that are referenced from the global namespaces aren't always deallocated or destroyed.

This is because it is not possible to deallocate those portions of the memory that are reserved by the C library.

Question: Write a program in Python for getting indices of N maximum values in a NumPy array.

Answer:

```
import numpy as np
arr = np.array([1, 3, 2, 4, 5])
print(arr.argsort()[-3:][::-1])
Output:
[4 3 1]
```

Question: Write code to show randomizing the items of a list in place in Python along with the output.

Answer:

```
from random import shuffle
x = ['hackr.io', 'Is', 'The', 'Best', 'For', 'Learning', 'Python']
shuffle(x)
print(x)
Output:
['For', 'Python', 'Learning', 'Is', 'Best', 'The', 'hackr.io']
```

Question: Explain memory managed in Python?

Answer: Python private heap space takes place of memory management in Python. It contains all Python objects and data structures. The interpreter is responsible to take care of this private heap and the programmer does not have access to it. The Python memory manager is responsible for the allocation of Python heap space for Python objects. The programmer may access some tools for the code with the help of the core API. Python also provides an inbuilt garbage collector, which recycles all the unused memory and frees the memory and makes it available to heap space.

Question: What is the lambda function?

Answer: An anonymous function is known as a lambda function. This function can have only one statement but can have any number of parameters.

```
a = lambda x,y : x+y
print(a(5, 6))
```

Question: What are Python decorators?

Answer: A specific change made in Python syntax to alter the functions easily are termed as Python decorators.

Question: Differentiate between list and tuple.

Answer: Tuple is not mutable it can be hashed eg. key for dictionaries. On the other hand, lists are mutable.

Question: How are arguments passed in Python? By value or by reference?

Answer: All of the Python is an object and all variables hold references to the object. The reference values are according to the functions; as a result, the value of the reference cannot be changed.

Question: What are the built-in types provided by the Python?

Answer:

Mutable built-in types:

- Lists
- Sets
- Dictionaries

Immutable built-in types:

- Strings
- Tuples
- Numbers

Question: How a file is deleted in Python?

Answer: The file can be deleted by either of these commands:

```
os.remove(filename)
os.unlink(filename)
```

Question: What are Python modules?

Answer: A file containing Python code like functions and variables is a Python module. A Python module is an executable file with a .py extension.

Python has built-in modules some of which are:

- os
- sys
- math
- random
- data time
- JSON

Question: What is the // operator? What is its use?

Answer: The // is a Floor Divisionoperator used for dividing two operands with the result as quotient displaying digits before the decimal point. For instance, 10//5 = 2 and 10.0//5.0 = 2.0.

Question: What is the split function used for?

Answer: The split function breaks the string into shorter strings using the defined separator. It returns the list of all the words present in the string.

Question: Explain the Dogpile effect.

Answer: The event when the cache expires and websites are hit by multiple requests made by the client at the same time. Using a semaphore lock prevents the Dogpile effect. In this system when value expires, the first process acquires the lock and starts generating new value.

Question: What is a pass in Python?

Answer: No-operation Python statement refers to pass. It is a place holder in the compound statement, where there should have a blank left or nothing written there.

Question: Is Python a case sensitive language?

Answer: Yes Python is a case sensitive language.

Question: Define slicing in Python.

Answer: Slicing refers to the mechanism to select the range of items from sequence types like lists, tuples, strings.

Question: What are docstring?

Answer: Docstring is a Python documentation string, it is a way of documenting Python functions, classes and modules.

Question: What is [::-1} used for?

Answer: [::-1} reverses the order of an array or a sequence. However, the original array or the list remains unchanged.

```
import array as arr
Num_Array=arr.array('k',[1,2,3,4,5])
Num_Array[::-1]
```

Question: Define Python Iterators.

Answer: Group of elements, containers or objects that can be traversed.

Question: How are comments written in Python?

Answer: Comments in Python start with a # character, they can also be written within docstring(String within triple quotes)

Question: How to capitalize the first letter of string?

Answer: Capitalize() method capitalizes the first letter of the string, and if the letter is already capital it returns the original string

Question: What is, not and in operators?

Answer: Operators are functions that take two or more values and returns the corresponding result.

- is: returns true when two operands are true
- not: returns inverse of a boolean value
- in: checks if some element is present in some sequence.

Question: How are files deleted in Python?

Answer: To delete a file in Python:

1. Import OS module
2. Use os.remove() function

Question: How are modules imported in Python?

Answer: Modules are imported using the **import** keyword by either of the following three ways:

```
import array
import array as arr
from array import *
```

Question: What is monkey patching?

Answer: Dynamic modifications of a class or module at run-time refers to a monkey patch.

Question: Does Python supports multiple inheritances?

Answer: Yes, in Python a class can be derived from more than one parent class.

Question: What does the method object() do?

Answer: The method returns a featureless object that is base for all classes. This method does not take any parameters.

Question: What is pep 8?

Answer: Python Enhancement Proposal or pep 8 is a set of rules that specify how to format Python code for maximum readability.

Question: What is namespace in Python?

Answer: A naming system used to make sure that names are unique to avoid naming conflicts refers to as Namespace.

Question: Is indentation necessary in Python?

Answer: Indentation is required in Python if not done properly the code is not executed properly and might throw errors. Indentation is usually done using four space characters.

Question: Define a function in Python

Answer: A block of code that is executed when it is called is defined as a function. Keyword **def** is used to define a Python function.

Question: Define self in Python

Answer: An instance of a class or an object is **self** in Python. It is included as the first parameter. It helps to differentiate between the methods and attributes of a class with local variables.