USM's Shriram Mantri Vidyanidhi Info Tech Academy

Python Programming Question Bank eDBDA Sept 21

Contents

PYTHON

Q.1) What is the output of the following?
i = 1 while True: if i%0O7 == 0:
    break   print(i)   i += 1

a) 1 2 3 4 5 6   b) 1 2 3 4 5 6 7   c) error   d) none of the entioned

Q.2) What is the output of the following? x
= ['ab', 'cd'] for i in x:
    i.upper() print(x)

a) ['ab', 'cd'].   b) ['AB', 'CD']   c) [None, None].   d) none of the mentioned

Q.3)What is the output of the following?
    x = "abcdef" i = "a" while i in x: print('i', end = " ")
a) no output   **b) i i i i i i …**   c) a a a a a a …   d) a b c d e f

Q.4) What is the output of the following? x = "abcdef" i = "a" while i in x[1:]:
    print(i, end = " ")

a) a a a a a a   b) a   **c) no output**   d) error

Q.5) What is the output of the following?

x = 'abcd' for i in x: print(i.upper())

a) a b c d   **b) A B C D**   c) a B C D   d) error

Q.6) What is the output of the following code?

if None:

print("Hello")

a) False   b) Hello   **c) Nothing will be printed**   d) Syntax error

Q.7) The if...elif...else executes only one block of code among several blocks.

a) **True**   b) False   c) It depends on expression used.   d) There is no elif statement in Python.

Q.8) What is the output of the following code?

for i in [1, 0]:   print(i+1)

a) 2

   1

b) [2, 1]

c) 2 0

d) [2, 0]

Q.9)In Python, for and while loop can have optional else statement?

a) Only for loop can have optional else statement

b) Only while loop can have optional else statement

c) **Both loops can have optional else statement**

d) Loops cannot have else statement in Python

Q.10) What is the output of the following code? i = sum = 0 while i <= 4:   sum += i   i = i+1

print(sum)

a) 0   **b) 10**   c) 4   d) None of the above

Q.11) What is the output of the following code?

while 4 == 4:   print('4')

a) 4 is printed once   b) 4 is printed four times

c) 4 is printed infinitely until program closes   d) Syntax error

Q.12) Is it better to use for loop instead of while if you are iterating through a sequence (like: list)? a) No, it's better to use while loop.

b) Yes, for loop is more pythonic choice.

c) No, you cannot iterate through a sequence using while loop.

d) No, you cannot iterate through a sequence using loops.

Q.13) Which of the following statement is true?
a) The break statement terminates the loop containing it.
b) The continue statement is used to skip the rest of the code inside the loop.
c) The break and continue statements are almost always used with if, if...else and if...elif...else statements.
   d) **All of the above.**

Q.14) What is the output of the following code?
for char in 'PYTHON STRING':
if char == ' ':
    break print(char,
end='')   if char ==
'O':   continue
a) PYTHON   b) PYTHONSTRING   c) PYTHN   d) STRING

Q.15) Which of the following statement is true about the pass statement?
a) The Python interpreter ignores the pass statement like comments.
b) The pass statement terminates the loop containing it.
c) **It is used as a placeholder for future implementation of functions, loops etc** d) All of the above.

Q.16) What is the output of the code shown below? import
math
[str(round(math.pi)) for i in range (1, 6)]
a) ['3', '3', '3', '3', '3', '3']   b) ['3.1', '3.14', '3.142', '3.1416', '3.14159', '3.141582']
c) ['3', '3', '3', '3', '3']   d) ['3.1', '3.14', '3.142', '3.1416', '3.14159']

Q.17) What is the output of the code shown below? t=32.00
[round((x-32)*5/9) for x in t]
a) [0]   b) 0   c) [0.00]   **d) Error**

Q.18) What is the output of the following?
print([i.lower() for i in "HELLO"])
a) ['h', 'e', 'l', 'l', 'o'].   b) 'hello'   c) ['hello'].   D) hello

Q.19) Suppose list1 is [3, 5, 25, 1, 3], what is min(list1) ?
a) 3   b) 5   c) 25   d) 1

Q.20) Suppose list1 is [1, 3, 2], What is list1 * 2 ?
a) [2, 6, 4].   b) [1, 3, 2, 1, 3]   **c) [1, 3, 2, 1, 3, 2] .**   d) [1, 3, 2, 3, 2, 1]

Q.21) What is the output when the following code is executed ? "Welcome to Python".split()
a) ["Welcome", "to", "Python"].   b) ("Welcome", "to", "Python")
c) {"Welcome", "to", "Python"}   d) "Welcome", "to", "Python"

Q.22) What will be the output? names1 =

['Amir', 'Bala', 'Charlie'] names2 =
[name.lower() for name in names1]
print(names2[2][0])

a) None   b) a   c) b   **d) c**

Q.23) What will be the output? values
= [[3, 4, 5, 1], [33, 6, 1, 2]]
v = values[0][0] for lst in
values: for element in
lst: if v > element:
v = element print(v)

a) 1   b) 3   c) 5   d) 6

Q.24) What is the output of the following
code? import copy a=[10,23,56,[78]]
b=copy.deepcopy(a) a[3][0]=95 a[1]=34 print(b)

a) [10,34,56,[95]].   **b) [10,23,56,[78]].**   c) [10,23,56,[95]].   d) [10,34,56,[78]].

Q.25) What is the output of the following piece of code?
a=list((45,)*4)
print((45)*4) print(a)

a) 180[(45),(45),(45),(45)].   b) (45,45,45,45).[45,45,45,45].
c) 180[45,45,45,45].   d) Syntax error

Q.26) What is the output of the code shown below?
A = [[1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]]
[A[i][len(A)-1-i] for i in range(len(A))]

a) [1, 5, 9]   b) [4, 5, 6]   **c) [3, 5, 7]**   d) [2, 5, 8]

BASIC OPERATOR

1. Which is the correct operator for power(x^y)?

a) X^y   **b) X**y**   c) X^^y   d) None of the mentioned

Explanation: In python, power operator is x**y i.e. 2**3=8.

2. Which one of these is floor division?

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

Explanation: When both of the operands are integer then python chops out the fraction part and gives you the round off value, to get the accurate answer use floor division. This is floor division. For ex, 5/2 = 2.5 but both of the operands are integer so answer of this expression in python is 2.To get the 2.5 answer, use floor division.

3. What is the order of precedence in python?

i) arentheses   ii)
xponential   iii)
Multiplication
iv) Division
v) Addition
vi) Subtraction

a) i,ii,iii,iv,v,vi   b) ii,i,iii,iv,v,vi   c) ii,i,iv,iii,v,vi   d) i,ii,iii,iv,vi,v   **Explanation:** For order of precedence, just remember this PEMDAS (similar to BODMAS)

4. What is answer of this expression, 22 % 3 is?

a) 7   **b) 1**   c) 0   d) 5

Explanation: Modulus operator gives remainder. So, 22%3 gives the remainder, that is, 1.

5. Mathematical operations can be performed on a string. State whether true or false.

a) True   **b) False**

Explanation: You can't perform mathematical operation on string even if the string is in the form: '1234…'.

6. Operators with the same precedence are evaluated in which manner?

a) Left to Right   b) Right to Left   c) Cant say   d) None of the mentioned

7. What is the output of this expression, 3*1**3?

a) 27   b) 9   **c) 3**   d) 1

Explanation: First this expression will solve 1**3 because exponential have higher precedence than multiplication, so 1**3 = 1 and 3*1 = 3. Final answer is 3.

8. Which one of the following have the same precedence?

a) Addition and Subtraction   b) Multiplication and Division
c) Both a and b   d) None of the mentioned

9. The expression Int(x) implies that the variable x is converted to integer. State whether true or false.
   a) True   b) False

10. Which one of the following have the highest precedence in the expression?
 a) Exponential   b) Addition   c) Multiplication   **d) Parentheses**

Explanation: Just remember: PEDMAS, that is, Parenthesis, Exponentiation, Division, Multiplication, Addition, Subtraction. Note that the precedence order of Division and Multiplication is the same. Likewise, the order of Addition and Subtraction is also the same.

WHILE AND FOR LOOP

1. What is the output of the following? x
   = ['ab', 'cd'] for i in x: i.upper()
   print(x)

a) ['ab', 'cd'].   b) ['AB', 'CD'].   c) [None, None].   d) none of the mentioned View
   Explanation: The function upper() does not modify a string in place, it returns a new string which isn't being stored anywhere.

2. What is the output of the following?   x
   = ['ab', 'cd'] for i in x:
       x.append(i.upper()) print(x)
   a) ['AB', 'CD']   b) ['ab', 'cd', 'AB', 'CD']
   c) ['ab', 'cd']   **d) none of the mentioned**
   Explanation: The loop does not terminate as new elements are being added to the list in each iteration.

3. What is the output of the following?   i
   = 1 while
       True: if i%3 == 0:
       break
       print(i)
       i + = 1
   a) 1 2   b) 1 2 3   **c) error**
   Explanation: SyntaxError, there shouldn't be a space between + and = in +=.

4. What is the output of the following?
   i = 1 while
   True: if i%0O7 == 0:   break   print(i)   i += 1
   a) 1 2 3 4 5 6   b) 1 2 3 4 5 6 7   c) error   d) none of the mentioned **Explanation:**
   Control exits the loop when i becomes 7.

5. What is the output of the following?
   i = 5 while True:
      if i%0O11 == 0:   break   print(i)   i += 1
   a) 5 6 7 8 9 10   **b) 5 6 7 8**   c) 5 6   d) error
   Explanation: 0O11 is an octal number.

6. What is the output of the following?   i
   = 5 while
       True: if i%0O9 ==
       0:        break
       print(i)   i += 1
   a)   5 6 7 8   b) 5 6 7 8 9   c) 5 6 7 8 9 10 11 12 13 14 15 …   **d) error**
   Explanation: 9 isn't allowed in an octal number.

7. What is the output of the following?   i
   = 2 while T rue: if i%3 == 0:
     break
       print(i)
       i += 2
   a)   2 4 6 8 10 …   **b) 2 4**   c) 2 3   d) error

Explanation: The numbers 2 and 4 are printed. The next value of i is 6 which is divisible by 3 and hence control exits the loop.

8. What is the output of the following?
True = False while True: print(True)   break
a)   True   b) False   c) None   **d) none of the mentioned**
Explanation: SyntaxError, True is a keyword and it's value cannot be changed.

DICTIONARY

1. Which of these about a dictionary is false?
a) The values of a dictionary can be accessed using keys
b) **The keys of a dictionary can be accessed using values**
c) Dictionaries aren't ordered
d) Dictionaries are mutable View Answer
Explanation: The values of a dictionary can be accessed using keys but the keys of a dictionary can't be accessed using values.

2. Which of the following is not a declaration of the dictionary?
a) {1: 'A', 2: 'B'}   b) dict([[1,"A"],[2,"B"]])   **c) {1,"A",2"B"}**   d) { }
Explanation: Option c is a set, not a dictionary.

3. What is the output of the following
code?   a={1:"A",2:"B",3:"C"} for i,j in
a.items():   print(i,j,end=" ")
a) 1 A 2 B 3 C   b) 1 2 3   c) A B C   d)1:"A" 2:"B" 3:"C"
Explanation: In the above code, variables i and j iterate over the keys and values of the dictionary respectively.

4. What is the output of the following piece of
code? a={1:"A",2:"B",3:"C"} print(a.get(1,4))
a) 1   **b) A**   c) 4   d) Invalid syntax for get method
Answer: b
Explanation: The get() method returns the value of the key if the key is present in the dictionary and the default value(second parameter) if the key isn't present in the dictionary.

5. What is the output of the following code?
a={1:"A",2:"B",3:"C"} print(a.get(5,4))
a) Error, invalid syntax   b) A   c) 5   **d) 4**
Explanation: The get() method returns the default value(second parameter) if the key isn't present in the dictionary.

6. What is the output of the following code?
a={1:"A",2:"B",3:"C"} print(a.setdefault(3))
a) {1: 'A', 2: 'B', 3: 'C'}   **b) C**

c) {1: 3, 2: 3, 3: 3}   d) No method called setdefault() exists for dictionary

Explanation: setdefault() is similar to get() but will set dict[key]=default if key is not already in the dictionary.

7. What is the output of the following code?
a={1:"A",2:"B",3:"C"}
a.setdefault(4,"D") print(a)
a) {1: 'A', 2: 'B', 3: 'C', 4: 'D'}.   b) None.   c) Error.   d) [1,3,6,10].
Explanation: setdefault() will set dict[key]=default if key is not already in the dictionary.

8. What is the output of the following code?
a={1:"A",2:"B",3:"C"} b={4:"D",5:"E"}
a.update(b) print(a)
a) {1: 'A', 2: 'B', 3: 'C'}   b) Method update() doesn't exist for dictionaries
c) {1: 'A', 2: 'B', 3: 'C', 4: 'D', 5: 'E'}   d) {4: 'D', 5: 'E'}
Explanation: update() method adds dictionary b's key-value pairs to dictionary a. Execute in python shell to verify.

9. What is the output of the following
code? a={1:"A",2:"B",3:"C"} b=a.copy()
b[2]="D" print(a)
a) Error, copy() method doesn't exist for dictionaries
b) **{1: 'A', 2: 'B', 3: 'C'}**
c) {1: 'A', 2: 'D', 3: 'C'}
d)"None" is printed
Explanation: Changes made in the copy of the dictionary isn't reflected in the original one.

10. What is the output of the following code?
a={1:"A",2:"B",3:"C"
} a.clear() print(a)
a) None   b) { None:None, None:None, None:None}
c) {1:None, 2:None, 3:None}   **d) { }**
Explanation: The clear() method clears all the key-value pairs in the dictionary.

11. Which of the following isn't true about dictionary keys?
a) More than one key isn't allowed
b) Keys must be immutable
c) **Keys must be integers**
d) When duplicate keys encountered, the last assignment wins
Explanation: Keys of a dictionary may be any data type that is immutable.

12. What is the output of the following
code? a={1:5,2:3,3:4} a.pop(3) print(a)
a) {1: 5}   **b) {1: 5, 2: 3}**   c) Error, syntax error for pop() method   d) {1: 5, 3: 4}
Explanation: pop() method removes the key-value pair for the key mentioned in the pop() method.

13. What is the output of the following code?
a={1:5,2:3,3:4} print(a.pop(4,9))

a) 9   b) 3   c) Too many arguments for pop() method   d) 4

Explanation: pop() method returns the value when the key is passed as an argument and otherwise returns the default value(second argument) if the key isn't present in the dictionary.

14. What is the output of the following code?
a={1:"A",2:"B",3:"C"} for i in a: print(i,end=" ")

a) 1 2 3

b) 'A' 'B' 'C'

c) 1 'A' 2 'B' 3 'C'

d) Error, it should be: for i in a.items():

Explanation: The variable i iterates over the keys of the dictionary and hence the keys are printed.

15. Execute the following in Python shell?
>>> a={1:"A",2:"B",3:"C"}
>>> a.items()

a) Syntax error   b) dict_items([('A'), ('B'), ('C')])

c) dict_items([(1,2,3)])   **d) dict_items([(1, 'A'), (2, 'B'), (3, 'C')])**

Explanation: The method items() returns list of tuples with each tuple having a key-value pair.

FILES

1. To open a file c:\scores.txt for reading, we use

a) infile = open("c:\scores.txt", "r")   **b) infile = open("c:\\scores.txt", "r")**

c) infile = open(file = "c:\scores.txt", "r")   d) infile = open(file = "c:\\scores.txt", "r") **Explanation:** Execute help(open) to get more details.

2. To open a file c:\scores.txt for writing, we use

a) outfile = open("c:\scores.txt", "w")   **b) outfile = open("c:\\scores.txt", "w")**

c) outfile = open(file = "c:\scores.txt", "w")   d) outfile = open(file = "c:\\scores.txt", "w") **Explanation:** w is used to indicate that file is to be written to.

3. To open a file c:\scores.txt for appending data, we use

a) outfile = open("c:\\scores.txt", "a")   b) outfile = open("c:\\scores.txt", "rw")

c) outfile = open(file = "c:\scores.txt", "w")   d) outfile = open(file = "c:\\scores.txt", "w") **Explanation:** a is used to indicate that data is to be apended.

4. Which of the following statements are true?

a) When you open a file for reading, if the file does not exist, an error occurs

b) When you open a file for writing, if the file does not exist, a new file is created

c) When you open a file for writing, if the file exists, the existing file is overwritten with the new file   d) **All of the mentioned**

5. To read two characters from a file object infile, we use

a) infile.read(2)   b) infile.read()   c) infile.readline()   d) infile.readlines()

6. To read the entire remaining contents of the file as a string from a file object infile, we use
a) infile.read(2)   **b) infile.read()**   c) infile.readline()   d) infile.readlines() View
Explanation: read function is used to read all the lines in a file.

7. What is the output?
1.   f = None
2.   for i in range (5):
3.   with open("data.txt", "w") as f:
4.   if i > 2:
5.   break
6.   print(f.closed)
a) True   b) False   c) None   d) Error Answer: a
Explanation: The WITH statement when used with open file guarantees that the file object is closed when the with block exits.

8. To read the next line of the file from a file object infile, we use
a) infile.read(2)   b) infile.read()   **c) infile.readline()**   d) infile.readlines()
Explanation: Execute in the shell to verify.

9. To read the remaining lines of the file from a file object infile, we use
a) infile.read(2)   b) infile.read()   C) infile.readline()   **d) infile.readlines()**
View Answer
Explanation: Execute in the shell to verify.

10. The readlines() method returns
a) str   **b) a list of lines**   c) a list of single characters   d) a list of integers
Explanation: Every line is stored in a list and returned.

TUPLES

Q.1) Which of the following is a Python tuple?
a) [1, 2, 3].   **b) (1, 2, 3)**   c) {1, 2, 3}   d) {}

Q.2) Suppose t = (1, 2, 4, 3), which of the following is incorrect?
a) print(t[3])   **b) t[3] = 45**   c) print(max(t))   d) print(len(t))

Q.3) What will be the output?
>>>t=(1,2,4,3)
>>>t[1:3]
a) (1, 2)   b) (1, 2, 4)   **c) (2, 4)**   d) (2, 4, 3)

Q.4) What will be the output?
>>>t=(1,2,4,3)

>>>t[1:-1]

a) (1, 2)   b) (1, 2, 4)   **c) (2, 4)**   d) (2, 4, 3)

Q.5) What will be the output? >>>t
= (1, 2, 4, 3, 8, 9)
>>>[t[i] for i in range(0, len(t), 2)]
a) [2, 3, 9]   b) [1, 2, 4, 3, 8, 9]   **c) [1, 4, 8]**   d) (1, 4, 8)

Q.6) What will be the output?
d = {"john":40, "peter":45} d["john"]
a) 40   b) 45   c) "john"   d) "peter"

Q.7) What will be the output?
>>>t1 = (1, 2, 4, 3)
>>>t2 = (1, 2, 3, 4)
>>>t1 < t2

a) True   **b) False**   c) Error   d) None

Q.8) What will be the output?
>>>my_tuple = (1, 2, 3, 4)
>>>my_tuple.append((5, 6, 7))
>>>print len(my_tuple)

a) 1   b) 2   c) 5   **d) Error**

Q.9) What will be the output?
numberGames = {} numberGames[(1,2,4)] = 8 numberGames[(4,2,1)] = 10 numberGames[(1,2)] = 12
sum = 0 for k in numberGames:
    sum += numberGames[k] print len(numberGames) + sum   a)
30   b) 24   **c) 33**   d) 12

Q. 10)What is the data type of (1)?
a)Tuple   **b)Integer**   c)List   d)Both tuple and integer

Q.11) If a=(1,2,3,4), a[1:-1] is
a)Error, tuple slicing doesn't exist   b)[2,3].   c)(2,3,4)   **d)(2,3)**

Q.12)What is the output of the following piece of code when executed in Python shell? >>>
a=("Check")*3
>>> a
a) ('Check','Check','Check')   b) * Operator not valid for tuples

c) ('CheckCheckCheck')   d) Syntax error

Q.13)What is the output of the following code?
>>> a=(1,2,3,4)
>>> del(a[2])
a) Now, a=(1,2,4)   b) Now, a=(1,3,4)   c) Now a=(3,4)   **d) Error as tuple is immutable**

Q.14)What is the output of the following code?
>>> a=(2,3,4)
>>> sum(a,3)
a)Too many arguments for sum() method
b)The method sum() doesn't exist for tuples   **c)12**
d)9

Q.15)Is the following piece of code valid?
>>> a=(1,2,3,4)
>>> del a
a) No because tuple is immutable   b) Yes, first element in the tuple is deleted
c) Yes, the entire tuple is deleted   d) No, invalid syntax for del method

Q.16)What type of data is: a=[(1,1),(2,4),(3,9)]?
a) Array of tuples   **b) List of tuples**   c) Tuples of lists   d) Invalid type

Q.17)What is the output of the following piece of code?
>>> a=(0,1,2,3,4)
>>> b=slice(0,2)
>>> a[b]
a) Invalid syntax for slicing   b) [0,2]   **c) (0,1)**   d) (0,2)

Q.18)Is the following piece of code valid?
>>> a=(1,2,3)
>>> b=('A','B','C')
>>> c=zip(a,b)
a)Yes, c will be ((1,2,3),('A','B','C'))   b)Yes, c will be ((1,2,3),('A','B','C'))
c)No because tuples are immutable   d)No because the syntax for zip function isn't valid

Q.19)Is the following piece of code valid?
>>> a,b,c=1,2,3
>>> a,b,c
a) Yes, [1,2,3] is printed   b) No, invalid syntax   **c) Yes, (1,2,3) is printed**   d) 1 is printed

Q.20)What is the output of the following piece of code?
a = ('check',) n = 2
for i in
range(int(n)):   a =

(a,)   print(a)

a) Error, tuples are immutable **b) (('check',),) ((('check',),),).**

c) (('check',)'check',) d) (('check',)'check',) ((('check',)'check',)'check',)

Q.21)Is the following line of code valid?

>>> a,b=1,2,3

a)Yes, this is an example of tuple unpacking. a=1 and b=2

b)Yes, this is an example of tuple unpacking. a=(1,2) and b=3

c)No, too many values to unpack

d)Yes, this is an example of tuple unpacking. a=1 and b=(2,3)

Q.22) What is the output of the following piece of code when executed in Python shell?

>>> a=(1,2) >>> b=(3,4)

>>> c=a+b

>>> c

a) (4,6)   **b) (1,2,3,4)**   c) Error as tuples are immutable   d) None

Q.23)What is the output of the following

code? >>> a,b=6,7 >>> a,b=b,a

>>> a,b

a) (6,7)   b) Invalid syntax   **c) (7,6)**   d) Nothing is printed

Q.24)What is the output of the following code?

>>> import collections

>>> a=collections.namedtuple('a',['i','j'])

>>> obj=a(i=4,j=7)

>>> obj

a) a(i=4, j=7)   b) obj(i=4, j=7)   c) (4,7)   d) An exception is thrown

Q.25)Tuples can't be made keys of a dictionary. True or False?

a) True   **b) False**

Q.26)Is the following piece of code valid?

>>> a=2,3,4,5

>>> a

a) Yes, 2 is printed   b) Yes, [2,3,4,5] is printed

c) No, too many values to unpack   **d) Yes, (2,3,4,5) is printed**

Q.27)What is the output of the following piece of code?

>>> a=(2,3,1,5)

>>> a.sort()

>>> a

a) (1,2,3,5)   b) (2,3,1,5)   c) None   **d) Error, tuple has no** attribute sort

Q.28)Is the following piece of code
valid? >>> a=(1,2,3) >>> b=a.update(4,)
a) Yes, a=(1,2,3,4) and b=(1,2,3,4)

b) Yes, a=(1,2,3) and b=(1,2,3,4)

c) **No because tuples are immutable**

d) No because wrong syntax for update() method

Q.29)What is the output of the following piece of code?
>>> a=[(2,4),(1,2),(3,9)]
>>> a.sort()
>>> a

a) [(1, 2), (2, 4), (3, 9)].

b) [(2,4),(1,2),(3,9)].

c) Error because tuples are immutable

d) **Error, tuple has no sort attribute**

FUNCTION

1. Which of the following is the use of function in python?
a) Functions are reusable pieces of programs
b) Functions don't provide better modularity for your application
c) you can't also create your own functions
d) All of the mentioned View Answer
Explanation: Functions are reusable pieces of programs. They allow you to give a name to a block of statements, allowing you to run that block using the specified name anywhere in your program and any number of times.

2. Which keyword is use for function?
a) Fun   b) Define   **c) Def**   d) Function

3. What is the output of the below program?
1.   def sayHello():
2.   print('Hello World!')
3.   sayHello()   4.   sayHello()
a) Hello World!
Hello World!
b) 'Hello
World!' 'Hello
World!' c)
Hello
Hello
d) None of the mentioned
Explanation: Functions are defined using the def keyword. After this keyword comes an identifier name for the function, followed by a pair of parentheses which may enclose some names of variables, and by the final colon that ends the line. Next follows the block of statements that are part of this function.
1.   def sayHello():
2.   print('Hello World!') # block belonging to the function
3.   # End of function #

4.
5. sayHello() # call the function
6. sayHello() # call the function again

4. What is the output of the below program?
1. def printMax(a, b):
2. if a > b:
3. print(a, 'is maximum')   4. elif a == b:
5. print(a, 'is equal to', b)
6. else:
7. print(b, 'is maximum')
8. printMax(3, 4)
a) 3   b) 4   **c) 4 is maximum**   d) None of the mentioned
Explanation: Here, we define a function called printMax that uses two parameters called a and b. We find out the greater number using a simple if..else statement and then print the bigger number.

5. What is the output of the below program ?
1.   x = 50
2.   def func(x):
3.   print('x is', x)
4.   x = 2
5.   print('Changed local x to', x)
6.   func(x)
7.   print('x is now', x)
a) x is now 50   b) x is now 2   c) x is now 100   d) None of the mentioned
Explanation: The first time that we print the value of the name x with the first line in the function's body, Python uses the value of the parameter declared in the main block, above the function definition. Next, we assign the value 2 to x. The name x is local to our function. So, when we change the value of x in the function, the x defined in the main block remains unaffected.
With the last print function call, we display the value of x as defined in the main block, thereby confirming that it is actually unaffected by the local assignment within the previously called function.

6. What is the output of the below program?
1.   x = 50
2.   def func():
3.   global x
4.   print('x is', x)
5.   x = 2
6.   print('Changed global x to', x)
7.   func()
8.   print('Value of x is', x)
a) x is 50
Changed global x to 2
Value of x is 50   **b) x**
is 50

Changed global x to 2 Value of x is 2   c)
x is 50
Changed global x to 50
Value of x is 50
d) None of the mentioned
Explanation: The global statement is used to declare that x is a global variable – hence, when we assign a value to x inside the function, that change is reflected when we use the value of x in the main block.

7. What is the output of below program?
1. def say(message, times = 1):
2. print(message * times)
3. say('Hello') 4. say('World', 5)
a) Hello WorldWorldWorldWorldWorld
b) Hello World 5
c) Hello
World,World,World,World,World
d) Hello
HelloHelloHelloHelloHello

Explanation: For some functions, you may want to make some parameters optional and use default values in case the user does not want to provide values for them. This is done with the help of default argument values. You can specify default argument values for parameters by appending to the parameter name in the function definition the assignment operator (=) followed by the default value.
The function named say is used to print a string as many times as specified. If we don't supply a value, then by default, the string is printed just once. We achieve this by specifying a default argument value of 1 to the parameter times.
In the first usage of say, we supply only the string and it prints the string once. In the second usage of say, we supply both the string and an argument 5 stating that we want to say the string message 5 times.

8. What is the output of the below program?
1.   def func(a, b=5, c=10):
2.   print('a is', a, 'and b is', b, 'and c is', c)
3.
4. func(3, 7)
5. func(25, c = 24) 6.
   func(c = 50, a = 100)

a) a is 7 and b is 3 and c is 10 a is 25 and b is 5 and c is 24 a is 5 and b is 100 and c is 50
b) a is 3 and b is 7 and c is 10 a is 5 and b is 25 and c is 24 a is 50 and b is 100 and c is 5
c) **a is 3 and b is 7 and c is 10 a is 25 and b is 5 and c is 24 a is 100 and b is 5 and c is 50**   d) None of the mentioned
Explanation: If you have some functions with many parameters and you want to specify only some of them, then you can give values for such parameters by naming them – this is called keyword arguments – we use the name (keyword) instead of the position (which we have been using all along) to specify the arguments to the function.
The function named func has one parameter without a default argument value, followed by two parameters with default argument values.

In the first usage, func(3, 7), the parameter a gets the value 3, the parameter b gets the value 7 and c gets the default value of 10.

In the second usage func(25, c=24), the variable a gets the value of 25 due to the position of the argument. Then, the parameter c gets the value of 24 due to naming i.e. keyword arguments. The variable b gets the default value of 5.

In the third usage func(c=50, a=100), we use keyword arguments for all specified values. Notice that we are specifying the value for parameter c before that for a even though a is defined before c in the function definition.

9. What is the output of below program?
1. def maximum(x, y):
2. if x > y:
3. return x
4. elif x == y:
5. return 'The numbers are equal' 6.   else:
7.   return y
8.
9.   print(maximum   (2,   3))

a) 2   **b) 3**   c) The numbers are equal   d) None of the mentioned

Explanation: The maximum function returns the maximum of the parameters, in this case the numbers supplied to the function. It uses a simple if..else statement to find the greater value and then returns that value.

10. Which of the following is a features of DocString?
a) Provide a convenient way of associating documentation with Python modules, functions, classes, and methods
b) All functions should have a docstring
c) Docstrings can be accessed by the __doc__ attribute on objects
d) All of the mentioned Answer: d

Explanation: Python has a nifty feature called documentation strings, usually referred to by its shorter name docstrings. DocStrings are an important tool that you should make use of since it helps to document the program better and makes it easier to understand.

ARGUMENT

1. What is the output of the following code? def
foo(k):   k = [1] q = [0] foo(q) print(q)

a) [0].   b) [1]   c) [1, 0].   d) [0, 1].

Explanation: A new list object is created in the function and the reference is lost. This can be checked by comparing the id of k before and after k = [1].

2. How are variable length arguments specified in the function heading?
a) one star followed by a valid identifier
b) one underscore followed by a valid identifier
c) two stars followed by a valid identifier

d) two underscores followed by a valid identifier View Answer   **Explanation:** Refer documentation.

3. Which module in the python standard library parses options received from the command line?
a) getopt   b) os   c) getarg   d) main
Explanation: getopt parses options received from the command line.

4. What is the type of sys.argv?
a) set   **b) list**   c) tuple   d) string View Answer
Explanation: It is a list of elements.

5. What is the value stored in sys.argv[0]?
a) null   b) you cannot access it   **c) the program's name**   d) the first argument
Explanation: Refer documentation.

6. How are default arguments specified in the function heading?
a) identifier followed by an equal to sign and the default value
b) identifier followed by the default value within back-ticks (")
c) identifier followed by the default value within square brackets ([])   d) identifier
Explanation: Refer documentation.

7. How are required arguments specified in the function heading?
a) identifier followed by an equal to sign and the default value
b) identifier followed by the default value within back-ticks (")
c) identifier followed by the default value within square brackets ([])   d) **identifier**

8. What is the output of the following code?
def foo(x):   x[0] = ['def']   x[1] = ['abc']   return id(x) q = ['abc', 'def']
print(id(q) == foo(q))
a) True   b) False   c) None   d) Error
Explanation: The same object is modified in the function.

9. Where are the arguments received from the command line stored?
a) sys.argv   b) os.argv   c) argv   d) none of the mentioned
Explanation: Refer documentation.

10. What is the output of the following?
def foo(i, x=[]):
    x.append(x.append(i))   return x for i in range(3):
y = foo(i) print(y)
a) [[[0]], [[0]], [1]], [[0]], [[0]], [1]], [2]]].
b) [[0], [[0], 1], [[0], [[0], 1], 2]].
c) **[0, None, 1, None, 2, None].**
d) [[[0]], [[0]], [1]], [[0]], [[0]], [1]], [2]]].

Explanation: append() returns None.

EXCEPTION HANDLING

Q.1) How many except statements can a try-except block have?

a) zero   b) one   c) more than one   **d) more than zero**

Q.2) When will the else part of try-except-else be executed?   a) always

b) when an exception occurs

c) **when no exception occurs**

d) when an exception occurs in to except block

Explanation: The else part is executed when no exception occurs.

Q.3) Is the following code valid? try:

   # Do something except:

Do something else:

Do something

a) no, there is no such thing as else

b) no, else cannot be used with except

c) no, else must come before except Explanation: Refer documentation.   d) **yes**

Q.4) Can one block of except statements handle multiple exception?

a) yes, like except TypeError, SyntaxError [,...].

b) yes, like except [TypeError, SyntaxError].

c) no

d) none of the mentioned

Q.5) When is the finally block executed?

a) when there is no exception

b) when there is an exception

c) only if some condition that has been specified is satisfied   **d) always**

Q.6) What is the output of the code shown below?

#generator def f(x):   yield x+1 g=f(8)

print(next(g))

a) 8   **b) 9**   c) 7   d) Error

Q.7) What is the output of the code shown below?

def f(x):   yield x+1   print("test")   yield x+2

g=f(10) print(next(g)) print(next(g))

a) No output   **b)11 test 12**   c) 11 test   d) 11

Q.8) What is the output of the following

code? def a():   try: f(x, 4)   finally:

print('after f')   print('after f?') a()

a) No output   b) after f?   **c) error**   d) after f

Q.9) The output of the code shown below is:
int('65.43')
a)ImportError   **b)ValueError**   c)TypeError   d)NameError

Q.10) Syntax errors are also known as parsing errors. Is this statement true or false?
a) True   b) False

Q.11) Which of the following blocks will be executed whether an exception is thrown or not?
a)except   b)else   **c)finally**   d)assert

Q.12) What is the output of the code shown below?
def getMonth(m):   if m<1 or
m>12:   raise
ValueError("Invalid")
print(m) getMonth(6)
a)ValueError   b)Invalid   **c)6**   d) ValueError("Invalid")

CORE DATA TYPES

1. What error occurs when you execute?
apple =mango
a) SyntaxError   **b) NameError**   c) ValueError   d) TypeError

2. Which of these in not a core datatype?
a) Lists   b) Dictionary   c) Tuples   **d) Class**

3. Given a function that does not return any value, What value is thrown by default when executed in shell.
a) int   b) bool   c) void   **d)None**
Explanation: Python shell throws a NoneType object back.

4. Following set of commands are executed in shell, what will be the output? .>>>str="hello"
>>>str[:2]
>>>
a) he   b) lo   c) olleh   d) hello View Answer
Explanation: We are printing only the 1st two bytes of string and hence the answer is "he".

5. Which of the following will run without errors ?
a) round(45.8)
b) round(6352.898,2,5)
c) round()
d) round(7463.123,2,1) View Answer

Explanation: Execute help(round) in the shell to get details of the parameters that are passed into the round function.

6. What is the return type of function id ?
a) int   b) float   c) bool   d) dict
Explanation: Execute help(id) to find out details in python shell.id returns a integer value that is unique.

7. In python we do not specify types,it is directly interpreted by the compiler, so consider the following operation to be performed.
1. >>>x = 13 ? 2   objective is to make sure x has a integer value, select all that
apply (python 3.xx) a) x = 13 // 2
b) x = int(13 / 2)
c) x = 13 % 2
d) **All of the mentioned**
Explanation: // is integer operation in python 3.0 and int(..) is a type cast operator.

8. Carefully observe the code and give the answer.
def example(a):
a = a + '2' a =
a*2
return a
>>>   example ("hello")
a) indentation Error
b) cannot perform mathematical operation on strings c)
   hello2
d) hello2hello2 View Answer
Explanation: Python codes have to be indented properly.

Q.9 What dataype is the object below ?   L
= [1,23,'hello',1].
a) list   b) dictionary   c) array   d) tuple View Answer
Explanation: List datatype can store any values within it.

Q.10 In order to store values in terms of key and value we use what core datatype.
a) list   b) tuple   c) class   **d) dictionary**
Explanation: Dictionary stores values in terms of keys and values.

11. Which of the following results in a SyntaxError ?
a) '"Once upon a time…", she said.'   b) "He said, 'Yes!'"
c) '3\'   d) "'That's okay"' View Answer
Explanation: Carefully look at the colons.

12. The following is displayed by a print function call:
1. tom
2. dick

3. harry
Select all of the function calls that result in this output
a) print('''tom \ndick \nharry''')
b) print('''tomdickharry''')
c) **print('tom\ndick\nharry')**
d) print('tom dick harry') View Answer   **Explanation:** The \n adds a new line.

13. What is the average value of the code that is executed below ?
1. >>>grade1 = 80
2. >>>grade2 = 90
3. >>>average = (grade1 + grade2) / 2
a) 85   **b) 85.1**   c) 95   d) 95.1
Explanation: Cause a decimal value to appear as output.

14. Select all options that print hello-how-are-you
a)   print('hello', 'how', 'are', 'you')
b)   print('hello', 'how', 'are', 'you' + '-' * 4)
c)   **print('hello-' + 'how-are-you')**
d)   print('hello' + '-' + 'how' + '-' + 'are' + 'you')   **Explanation:** Execute in the shell.

15. What is the return value of trunc() ?

a) int   b) bool   c) float   d) None View Answer
Explanation: Executle help(math.trunc) to get details.

CLASSES & OBJECT

focuses on "Classes and Objects – 1".
1. _____ represents an entity in the real world with its identity and behaviour.
a) A method   **b) An object**   c) A class   d) An operator View Answer
Explanation: An object represents an entity in the real world that can be distinctly identified. A class may define an object.

2. _____ is used to create an object.
a) class   **b) constructor**   c) User-defined functions   d) In-built functions View Answer
Explanation: The values assigned by the constructor to the class members is used to create the object.

3. What is the output of the following code?
class test: def
__init__(self,a="Hello
World"):   self.a=a
def display(self):
print(self.a) obj=test()
obj.display()
a) The program has an error because constructor can't have default arguments   b)
Nothing is displayed

c) "Hello World" is displayed
d) The program has an error display function doesn't have parameters View Answer
Explanation: The program has no error. "Hello World" is displayed. Execute in python shell to verify.

4. What is setattr() used for?
a) To access the attribute of the object   **b) To set an attribute**
c) To check if an attribute exists or not   d) To delete an attribute View Answer
Explanation: setattr(obj,name,value) is used to set an attribute. If attribute doesn't exist, then it would be created.

5. What is getattr() used for?
a) To access the attribute of the object   b) To delete an attribute
c) To check if an attribute exists or not   d) To set an attribute View Answer
Explanation: getattr(obj,name) is used to get the attribute of an object.

6. What is the output of the following code?
class change: def __init__(self, x, y, z):
self.a = x + y + z   x = change(1,2,3) y = getattr(x,
'a') setattr(x, 'a', y+1) print(x.a)
a) 6   **b) 7**   c) Error   d) 0
Explanation: First, a=1+2+3=6. Then, after setattr() is invoked, x.a=6+1=7.

7. What is the output of the following code?
 class test: def __init__(self,a):   self.a=a
     def display(self):   print(self.a) obj=test() obj.display()
a) Runs normally, doesn't display anything
b) Displays 0, which is the automatic default value
c) **Error as one argument is required while creating the object**
d) Error as display function requires additional argument View Answer
Explanation: Since, the __init__ special method has another argument a other than self, during object creation, one argument is required. For example: obj=test("Hello")

8. Is the following piece of code correct? >>>
class A: def __init__(self,b):
  self.b=b   def display(self):
             print(self.b)
>>> obj=A("Hello") >>> del obj
a) True   b) False
Explanation: It is possible to delete an object of the class. On further typing obj in the python shell, it throws an error because the defined object has now been deleted.

9. What is the output of the following code?
class test: def __init__(self):
     self.variable = 'Old'
self.Change(self.variable)
def Change(self, var):

var = 'New' obj=test()
print(obj.variable)
a) Error because function change can't be called in the __init__ function   b)
'New' is printed
c) 'Old' is printed
d) Nothing is printed
Explanation: This is because strings are immutable. Hence any change made isn't reflected in the original string.

10. What is Instantiation in terms of OOP terminology?
a) Deleting an instance of class   b) Modifying an instance of class
c) Copying an instance of class   **d) Creating an instance of class**
Explanation: Instantiation refers to creating an object/instance for a class.

11. What is the output of the following code?
class fruits: def __init__(self,
price):   self.price = price
obj=fruits(50) obj.quantity=10
obj.bags=2
print(obj.quantity+len(obj.__dict__))
a) 12   b) 52   **c) 13**   d) 60
Explanation: In the above code, obj.quantity has been initialised to 10. There are a total of three items in the dictionary, price, quantity and bags. Hence, len(obj.__dict__) is 3.

12. What is the output of the following code?
 class Demo:
def __init__(self):
pass   def
test(self):
     print(__name__)   obj
= Demo() obj.test()
a) Exception is thrown   **b) __main__**   c) Demo   d) test
Explanation: Since the above code is being run not as a result of an import from another module, the variable will have value "__main__".

INHERITANCE

1. What type of inheritance is illustrated in the following piece of code?
class A():   pass class B(A):   pass class C(B):   pass
  a) Multi-level inheritance   b) Multiple inheritance
c) Hierarchical inheritance   d) Single-level inheritance
Explanation: In multi-level inheritance, a subclass derives from another class which itself is derived from another class.

2. What does single-level inheritance mean?

a) A subclass derives from a class which in turn derives from another class
b) A single superclass inherits from multiple subclasses
c) **A single subclass derives from a single superclass**
d) Multiple base classes inherit a single derived class View Answer

Explanation: In single-level inheritance, there is a single subclass which inherits from a single superclass. So the class definition of the subclass will be: class B(A): where A is the superclass.

3. What is the output of the following piece of code?
```
class A:      def
__init__(self):
self.__i = 1        self.j
= 5      def
display(self):
print(self.__i, self.j) class B(A):
def __init__(self):
super().__init__()        self.__i
= 2        self.j = 7   c = B()
c.display()
```
a) 2 7   b) 1 5   **c) 1 7**   d) 2 5

Explanation: Any change made in variable i isn't reflected as it is the private member of the superclass.

4. Which of the following statements isn't true?
a) A non-private method in a superclass can be overridden
b) A derived class is a subset of superclass
c) **The value of a private variable in the superclass can be changed in the subclass**
d) When invoking the constructor from a subclass, the constructor of superclass is automatically invoked
   Explanation: If the value of a private variable in a superclass is changed in the subclass, the change isn't reflected.

5. What is the output of the following piece of code?
```
class A:      def __init__(self,x):
self.x = x    def count(self,x):
self.x = self.x+1 class B(A):      def
__init__(self, y=0):
A.__init__(self, 3)        self.y = y
def count(self):
    self.y += 1
def main():      obj
= B()
obj.count()
print(obj.x, obj.y)
main()
```
a) 3 0   **b) 3 1**   c) 0 1   d) An exception in thrown

Explanation: Initially x=3 and y=0. When obj.count() is called, y=1.

6. What is the output of the following piece of code when executed in the Python shell?

```
>>> class A:  pass  >>> class B(A):
        pass
>>> obj=B()
>>> isinstance(obj,A)
```
a) True
b) False
c) Wrong syntax for isinstance() method
d) Invalid method for classes
Explanation: isinstance(obj,class) returns True if obj is an object class.

7. Which of the following statements is true?
a) The __new__() method automatically invokes the __init__ method
b) The __init__ method is defined in the object class
c) **The __eq(other) method is defined in the object class**
d) The __repr__() method is defined in the object class View Answer
Explanation: The __eq(other) method is called if any comparison takes place and it is defined in the object class.

8. Method issubclass() checks if a class is a subclass of another class. True or False?
a) True   b) False
Explanation: Method issubclass() returns True if a class is a subclass of another class and False otherwise.

9. What is the output of the following piece of code?
```
class A:     def
__init__(self):
self.__x = 1  class  B(A):
def          display(self):
print(self.__x) def main():
obj = B()
obj.display(
) main()  a)
```
1
b) 0
c) Error, invalid syntax for object declaration
d) **Error, private class member can't be accessed in a subclass**
Explanation: Private class members in the superclass can't be accessed in the subclass.

10. What is the output of the following piece of
code? class A: def __init__(self):   self._x = 5
class B(A):   def display(self):   print(self._x) def
main():   obj
= B()
obj.display()
main()
a) Error, invalid syntax for object declaration
b) Nothing is printed

c) **5**

d) Error, private class member can't be accessed in a subclass

Explanation: The class member x is protected, not private and hence can be accessed by subclasses.

11. What is the output of the following piece of code?

```
class A:    def __init__(self,x=3):
 self._x = x        class B(A):
def __init__(self):
super().__init__(5)     def
display(self):

print(self._x)
def main():
obj = B()
obj.display()
main()
```
a) 5

b) Error, class member x has two values

c) 3

d) Error, protected class member can't be accessed in a subclass View Answer

Explanation: The super() method re-assigns the variable x with value 5. Hence 5 is printed.

11. What is the output of the following piece of code?

```
class A:     def test1(self):        print(" test of A called
") class B(A):     def test(self):
  print(" test of B called ")
class C(A):     def
test(self):
  print(" test of C called ")
class D(B,C):     def
test2(self):        print("
test of D called ")
obj=D() obj.test()
```
a) test of B called test of C called

b) test of C called test of B called

c) **test of B called**

d) Error, both the classes from which D derives has same method test()

Explanation: Execute in Python shell to verify. If class D(B,C): is switched is class D(C,B): test of C is called.

13. What is the output of the following piece of code?

```
class A:     def test(self):        print("test of A called")
class B(A):    def test(self):        print("test of B called")
super().test()    class C(A):     def test(self):
 print("test of C called")
super().test() class D(B,C):     def
test2(self):        print("test of D called")
obj=D() obj.test()
```

a) **test of B called test of C called test of A called**
b) test of C called test of B called
c) test of B called test of C called
d) Error, all the three classes from which D derives has same method test()
Explanation: Since the invoking method, super().test() is called in the subclasses, all the three methods of test() in three different classes is called.

EXTRA MCQ

1. The value of a in the following example is?
   >>> a = [1,2,3,4]
   >>> a = a.append(5)
   >>> print a

   A. [1,2,3]   B. [1,2,3,4]   **C. [1,2,3,4,5]**   D. None of the Above

2. In computer programming, _____ is the term used to describe sections of code that have to be included in many places with little or no alteration.
   A. shebang   B. REPL   **C. boilerplate**   D. header

3. When using sys.argv - The first argument points to the
   A. First parameter   B. Second parameter
   C. Third parameter   **D. script itself**

4. When a python file is run directly, the special variable "__name__" is set to .
   A. "__main_void__"   B. "__void_main__"   **C. "__main__"**   D. "__void__"

5. Suppose the file "binky.py" contains a "def foo()". The fully qualified name of that foo function is
   _____.
   A. "main.binky"   B. "binky.main"   C. "boo.binky"   **D. "binky.foo"**

6. Inside a python interpreter, the _____ command gives a quick list of the defined symbols in python.
   A. snapshot   B. view   C. help   **D. dir**

7. In python, the _____ of a piece of code affects its meaning.   **A. whitespace indentation**
   B. space between lines
   C. Both A and B
   D. D. None of the above

8. Logical Connectors are spelled out with _____.
   A.   Letters   B. Integers   **C. Symbols**   D. All of the above

9. In Python boolean operations, Empty string is counted as _____.
   A. True   **B. False**   C. None   D. None of the above.

10. A "raw" string literal is prefixed by an '_____' and passes all the chars through without special treatment of backslashes,

A. **r**   **B.** R   C. \r   D. \R

11. r'x\nx' evaluates to the length _____ string

A. 1   B. 2   C. 3   D. **4**

12. list.append() returns the special value _____

A. None   **B.** Error   C. True   D. Null

13. REPL stands for _____

A. Read-Evaluate-Parse-Loop   B. Read-Evaluate-Print-List

C. Read-Enter-Print-List   **D. Read-Evaluate-Print-Loop**

14. In computer programming, _____ is a method by which individual units of source code are tested to determine if they are fit for use.

A. Load Testing   B. Integration Testing   C. Stress Testing   D. **Unit Testing**

15. A _____ is the smallest testable part of an application.

A. Unit   B. Module   C. File   D. Library