-----TOP 100 PYTHON MCQ QUESTIONS FROM QT & IHUB-----

Q1) What is output of following code?

```
import time
import keyword as kw
print(kw.kwlist)
print ()
time.sleep(2)
print("End of an application")
```

a) Error
b) Indentation error
c) 33 or 35 keyword
d) None of above

Q2)

```
import time
   ABC_abc_12345=1600
   print (  ABC_abc_12345)
```

a) Indentation error
b) Error
c) None
d)1600

Q3)

```
Import time
X1= {}
print(X1)
print(type(X1))
```

a) Set
b) List
c) Tuple
d) None

Q4) What is PEP8?
a) Tool
b) Technology
c) Methodology
d) Python enhancement proposal document

Q5)

```
import time
str1=" core Python"
print(str1)
print(type(str1))
print ()
str1[0]="C"
print(str1)
print()
time.sleep(2)
print ("End of an application")
```

a) Core Python
b) Python
c) None
d) Core

Q6)

```
l=[1, 0, 2, 0, 'hello', '', []]
list(filter(bool, l))
```

a) [1, 0, 2, 'hello', '', []]
b) Error
c) [1, 2, 'hello']
d) [1, 0, 2, 0, 'hello', '', []]

Q7)

```
def f(x):
    def f1(*args, **kwargs):
        print("IHUB Factory for relaesing the sotware developer")
        return x(*args, **kwargs)
    return f1
```

a) any number of
b) 0
c) 1
d) 2

Q8)

```
for i in [1, 2, 3, 4][::-1]:
        print (i)
```

a) 4 3 2 1
b) error
c) 1 2 3 4
d) none of the mentioned

Q9)

```
x = 'abcd'
for i in range(len(x)):
    print(i)
```

a) error
b) 1 2 3 4
c) a b c d
d) 0 1 2 3

Q10)

```
  x = [[0], [1]]
print((' '.join(list(map(str, x))),))
```

a) 01b) [0] [1]
c) ('01')
d) ('[0] [1]',)

Q11)

```
Import time
 X1="Core Python"
 X1.sort()
 print(X1)
```

a)Sort the string
b)Error
c)None
d)core Python

Q12)

```
Import time
 T1= (1000,5,2,180,60)
 T1.add('E')
 Print(T1)
```

a) (1000,5,2,180,60,'E')
b) (1000,5,2,180,60)
c) Error
d) Tuple

Q13)

```
D1={pid:1001,pname:Mobile,price:17000,company:LG}
Print(D1)
```

a) {pid:1001,pname:Mobile,price:17000,company:LG}
b) " "
c) None
d) Error

Q14)Which of the following declarations is incorrect in python language?

a)xyzp = 5,000,000
b)x y z p = 5000 6000 7000 8000
c)x,y,z,p = 5000, 6000, 7000, 8000
d)x_y_z_p = 5,000,000

Q15) Which of the following operators is the correct option for power(ab)?

a)a ^ b
b)a**b
c)a ^ ^ b
d)a ^ * b

Q16) Read the following program

```
z = "xyz"
j = "j"
while j in z:
   print(j, end=" ")
```

Q14)

```
D1={0:100,1:200,2:300,3:400}
Print(D1)
Obj1=D1.get(0)
Print(Obj1)
```

a) Error
b) {0:100,1:200,2:300,3:400}
c) 100
d)None

Q15)

```
x = 'Ihub'
for i in range(len(x)):
    x[i].upper()
print (x)
```

a)PQRS
b)pqrs
c)qrs
d)None of these

Q16)

```
MANGO = APPLE
print(MANGO)
```

a)NameError
b)SyntaxError
c)TypeError
d)ValueError

Q17)

```
def QT1(a):
    aa = a + '1'
    aa = a*1
    return a
if(__name__=="__main__"):
    print(QT1("Data Sceinece"))
```

a)Data Sceince
b)Data
c)Scenice
d)None

Q18)

```
 print(print(print("Quality_Thought")))
```

a)Quality_Thought None None

b)None None Quality_Thought
c)None Quality_Thought None
d)Quality_Thought

Q19)

```
    int1 = 10
    int2 = 6
  if int != int2:
   int2 = ++int2
   print(int1 - int2)
```

a)2
b)4
c)6
d)None

Q17)

```
a = "CorePython"
print(*a)
```

a)CorePython
b)C o r e P y t h o n
c)Syntax
d)None

Q18)

```
   i = 2, 10
   j = 3, 5
  add = i + j
  print(add)
```

a)(5, 10)
b)20
c)(2, 10, 3, 5)
d)SyntaxError: invalid syntax

Q19)

```
print(int(6 == 6.0) * 3 + 4 % 5)
```

Q20)Which of the following arithmetic operators cannot be used with strings in python?
b)18
c)20
d)7

a)+
b)*
c)-
d)all above mention

Q21)

```
    print("Core", 'Python', sep='2')
```

a)CorePython2

b)Core2Python
c)Core
d)Python

Q22)
```
_ = '1 2 3 4 5 6'
print(_)
```

a)SyntaxError: EOL while scanning string literal
b)SyntaxError: invalid syntax
c)NameError: name '_' is not defined
d)1 2 3 4 5 6

Q23)Which of the following keywords is not reversed keyword in python?

a)None
b)class
c)goto
d)and

Q24)
```
  a = '1 2'
  print(a * 2)
  print(a * 0)
  print(a * -2)
```

a)1 2 1 2
b)2 4
c)0
d)-1 -2 -1 -2

Q25)
```
    a = "123789"
  while x in a:
     print(x, end=" ")
```

a)i i i i i i …
b)123789
c)SyntaxError
d)NameError

Q26) PVM is often called _____.

a)Python interpreter
b)Python compiler
c)Python volatile machine
d)Portable virtual machine

Q27)
```
  i = {4, 5, 6}
  i.update({2, 3, 4})
  print(i)
```

a)2 3 4 4 5 6
b)2 3 4 5 6
c)4 5 6 2 3 4
d)Error, duplicate element presents in list

Q28)

```
i=(12, 20, 1, 0, 25)
i.sort()
print(i)
```

a)0 1 12 20 25
b)1 12 20 25
c)FunctionError
d)AttributeError

Q29)

```
str1="python language"
str1.find("p")
print(str1)
```

a)Print the index value of the p.
b)p
c)python language
d)AttributeError

Q30)

```
    flag = ""
   a = 0
   i = 1
  while(a < 3):
    j = 1
    if flag:
       i = j * i + 5
    else:
       i = j * i + 1
  a = a + 1
 print(i)
```

a)12
b)4
c)11
d)16

31)

```
arr = [3 , 2 , 5 , 6 , 0 , 7, 9]
add1 = 0
add2 = 0
for elem in arr:
   if (elem % 1 == 0):
      add1 = add1 + elem
      continue
```

```
    if (elem % 3 == 0):
        add2 = add2 + elem
print(add1 , end=" ")
print(add2)
```

a)32 0
b)0 32
c)18 0
d)0 18

32)

```
    import time
    t1=(x*x for x in range(10)
    print(t1)
    print(type(t1)
```

a)square of number
b)generator objec
c)None
d)Error

33)

```
b = [1, 65, 23, 'Hello', 3.23]
if(len(b) == 0):
    print("Given List is Empty")
else:
    print("List is not empty")
```

a)Given List is Empty
b)List is not Empty
c)None
d)Error

34)

```
c = [True, 42, 9.23, 12, 22]
c.clear()
print(c)
```

a)[True, 42, 9.23, 12, 22]
b)[]
c)[True]
d)None

35)

```
d = [4, 3.12, False, "Python", 66]
count = 0
for i in d:
    count += 1
print(count)
```

36)

```
e = [14, 57, 2, 43, 29]
e.sort()
print(e[1])
```

a)57
b)14
c)43
d)None

37)
```
def simple_interest(p,t,r):
    print('The principal is', p)
    print('The time period is', t)
    print('The rate of interest is',r)

    si = (p * t * r)/100

    print('The Simple Interest is', si)
    return si

if(__name__=="__main__):

    simple_interest(8, 6, 8)
```

a)8 6 8 3.86
b)6 8 8 2.86
c)2 2 2 3.86
d)1 1 1 3.86

38)
```
def compound_interest(principal, rate, time):

    # Calculates compound interest
    Amount = principal * (pow((1 + rate / 100), time))
    CI = Amount - principal
    print("Compound interest is", CI)


if(__name__=="__main__"):
  compound_interest(10000, 10.25, 5)
```

a)Compound interest is 6288.946267774416
b)Compound interest is 6211.946267774416
c)Compound interest is 6223.946267774416
d)Compound interest is 6212.946267774416

39)
```
 s = string.split()[::-1]
 l = []
for i in s:
    l.append(i)
print(" ".join(l))

input_string = "Data Engineer"
```

a)reverse of a string
b)same string
c)Error
d)None

40)
MyString1 = "A geek in need is a geek indeed"

```
if "need" in MyString1:
   print("Yes! it is present in the string")
else:
   print("No! it is not present")
```

a)Error
b)Yes! it is present in the string
c)Not it is not present
d)None

41)
```
test_str = 'Gfg is best . Geeks are good and Geeks like Gfg'
res = {key: test_str.count(key) for key in test_str.split()}
print("The words frequency : " + str(res))
```

a)Empty dict
b) {‘Gfg’: 2, ‘is’: 1, ‘best’: 1, ‘.’: 1, ‘Geeks’: 2, ‘are’: 1, ‘good’: 1, ‘and’: 1, ‘like’: 1}
c)Error
d)None

42)
```
n="This is a python language"
s=n.split(" ")
for i in s:
  if len(i)%2==0:
    print(i)
```

a)This
is
python
language
b)Error
c)Empty string
d)b,c

43)
```
test_str = "Python is a is Python"
# printing original string
print ("The original string is : " + test_str)
# using naive method to get
# Least Frequent Character in String
all_freq = {}
for i in test_str:
 if i in all_freq:
```

```
  all_freq[i] += 1
 else:
  all_freq[i] = 1
res = min(all_freq, key = all_freq.get)
print ("The minimum of all characters in GeeksforGeeks is : " + str(res))
```

a)a
b)Python
c)Python is a is Python
d)Error

44)

```
def prime(x, y):
    prime_list = []
    for i in range(x, y):
        if i == 0 or i == 1:
            continue
        else:
            for j in range(2, int(i/2)+1):
                if i % j == 0:
                    break
            else:
                prime_list.append(i)
    return prime_list
starting_range = 2
ending_range = 7
lst = prime(starting_range, ending_range)
if len(lst) == 0:
    print("There are no prime numbers in this range")
else:
    print("The prime numbers in this range are: ", lst)
```

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

45)

```
if num > 11:
    # Iterate from 2 to n / 2
    for i in range(2, int(num/2)+1):
        # If num is divisible by any number between
        # 2 and n / 2, it is not prime
        if (num % i) == 0:
            print(num, "is not a prime number")
            break
    else:
        print(num, "is a prime number")
else:
    print(num, "is not a prime number")
```

a)Error
b)None

c)11 is a prime number
d)11 is not a prime number

46)

```
def Fibonacci(n):
    if n<= 0:
        print("Incorrect input")
    # First Fibonacci number is 0
    elif n == 1:
        return 0
    # Second Fibonacci number is 1
    elif n == 2:
        return 1
    else:
        return Fibonacci(n-1)+Fibonacci(n-2)
print(Fibonacci(10))
```

a)17
b)34
c)25
d)39

47)

```
def squaresum(n) :
    sm = 0
    for i in range(1, n+1) :
        sm = sm + (i * i)
    return sm
n = 4
print(squaresum(n))
```

a)24
b)30
c)21
d)167

48)

```
def multiplyList(myList):
    result = 1
    for x in myList:
        result = result * x
    return result
list1 = [1, 2, 3]
list2 = [3, 2, 4]
print(multiplyList(list1))
print(multiplyList(list2))
```

a)6,24
b)Erorr
c)None

d)24

49)
```
ist1 = [10, 20, 4, 45, 99]
list1.sort(reverse=True)
print("Smallest element is:", list1[-1])
```

a)10
b)4
c)45
d)99

50)
```
list1 = [11, 5, 17, 18, 23, 50]

for ele in list1:
    if ele % 2 == 0:
        list1.remove(ele)
print("New list after removing all even numbers: ", list1)
```

a)[11, 5, 17, 23]
b)[]
c)None
d)Error

51)
```
for num in range(4,15,2):
    print(num)
```

a)4 6 8 10 12 14
b)12345677
c)Error
d)None

52)
```
list1 = [11, -21, 0, 45, 66, -93]
for num in list1:
    if num & gt= 0:
        print(num, end=" ")
```

a)11 0 45 66
b)-21 -93
c)11, -21, 0, 45, 66, -93
d)Error

53)

```
def negativenumbers(a,b):
  out=[i for i in range(a,b+1) if i<0]
  print(*out)
```

```
# a -> start range
a=-4
# b -> end range
b=5
negativenumbers(a,b)
```

a)-4 -3 -2 -1
b)-1 -2 -3 -4
c)1 2 3 4
d)Error

54)

```
test_list = [5, 6, [], 3, [], [], 9]

# printing original list
print("The original list is : " + str(test_list))

# Remove empty List from List
# using list comprehension
res = [ele for ele in test_list if ele != []]

# printing result
print("List after empty list removal : " + str(res))
```

a)[5, 6, [], 3, [], [], 9],[5,6,3,9]
b)Error
c)None
d)b,c

55)

```
my_list = [1, 2, 3, 4, 5,
           6, 7, 8, 9]
start = 0
end = len(my_list)
step = 3
for i in range(start, end, step):
    x = i
    print(my_list[x:x+step])
```

a)[1,2,3] [4,5,6][7,8,9]
b)Error
c)None
d)[7,8,9]

56)

```
def Merge(dict1, dict2):
    return(dict2.update(dict1))
dict1 = {'a': 10, 'b': 8}
dict2 = {'d': 6, 'c': 4}
print(Merge(dict1, dict2))
```

a){'c': 4, 'a': 10, 'b': 8, 'd': 6}
b){'a': 10, 'b': 8}
c){'d': 6, 'c': 4}
d){}

57)

```
test_dict = {'month' : [1, 2, 3],
             'name' : ['Jan', 'Feb', 'March']}
print("The original dictionary is : " + str(test_dict))
res = dict(zip(test_dict['month'], test_dict['name']))
print("Flattened dictionary : " + str(res))
```

a)Error
b)None
c)The original dictionary is : {‘name’: [‘Jan’, ‘Feb’, ‘March’], ‘month’: [1, 2, 3]} Flattened dictionary : {1: ‘Jan’, 2: ‘Feb’, 3: ‘March’}
d)[]

58)

```
test_dict = {"Gfg" : 1, "is" :  3, "Best" : 2}
print("The original dictionary is : " + str(test_dict))
res = list(test_dict.keys()) + list(test_dict.values())
print("The ordered keys and values : " + str(res))
```

a)[1,2,3]
b)['Gfg', 'is', 'Best', 1, 3, 2]
c)['Gfg', 'is', 'Best']
d)Error

59)

```
# Function calling
def dictionairy():
    # Declare hash function
    key_value = {}
# Initializing value
    key_value[2] = 56
    key_value[1] = 2
    key_value[5] = 12
    key_value[4] = 24
    key_value[6] = 18
    key_value[3] = 323
    print("Task 1:-\n")
    print("key_value", key_value)
    # iterkeys() returns an iterator over the
    # dictionary’s keys.
    for i in sorted(key_value.keys()):
        print(i, end=" ")
def main():
    # function calling
    dictionairy()
```

```
# Main function calling
if __name__ == "__main__":
    main()
```

a)key_value {2: 56, 1: 2, 5: 12, 4: 24, 6: 18, 3: 323}
1 2 3 4 5 6
b)key_value {2: 56, 1: 2, 5: 12, 4: 24, 6: 18, 3: 323}
c)Error
d)None

60)

```
def Sort_Tuple(tup):
    # getting length of list of tuples
    lst = len(tup)
    for i in range(0, lst):
        for j in range(0, lst-i-1):
            if (tup[j][1] > tup[j + 1][1]):
                temp = tup[j]
                tup[j]= tup[j + 1]
                tup[j + 1]= temp
    return tup

# Driver Code
tup =[('for', 24), ('is', 10), ('Geeks', 28),
      ('Geeksforgeeks', 5), ('portal', 20), ('a', 15)]

print(Sort_Tuple(tup))
```

a)Output will generate
b)Error
c)[('Geeksforgeeks', 5), ('is', 10), ('a', 15), ('portal', 20), ('for', 24), ('Geeks', 28)]
d)None

61)

```
mytuple1=(5, 1, 7, 6, 2)
mytuple1.pop(2)
print(mytuple1)
```

a)5 1 7 6 2
b)No output
c)AttributeError
d)None of the these

62)What is package in python

a)Contains one or more than modules
b)collection of libary
c)collection of files
d)collection of framework

63)What is __init__.py indicates
a)It is a package in python
b)Error
c)None
d)sum of number

64)Pid=1001
    Pid>1001
    print(Pid)

a)1001
b)Error
c)Attribute error
d)None

65)Product_Id=1009
    assert Product_Id>1009
    print(Product_Id)

a)1009
b)assertion error
c)None
d)Error

66)import time
     class I_HUB:
        def __init_(self):
           print("ONE")
i1=I_HUB()

a)ONE
b)Attribute Error
c)Error
d)None

67)
    import time
    class I_HUB2:
        def m1(cls):
          print('A')
i1=I_HUB2()
i1.m1()

a)A
b)None
c)Error
d)404 Error Code

68)
    import time

```
    class QT1:
        def __init__(self,a):
            self.a=1001
            print(self.a)
q1=QT1()
```

a)1001
b)Typeerror
c)SynaxError
d)AttributeError

69)

```
import time
class I_HUB1:
   @staticmethod
   def m1(a,b):
      return a+b
i1=I_HUB1()
i1.m1(100,200)
```

a)300
b)Empty
c)Error
d)None

70)

```
import time
class I_HUB1:
   school_name="IHUB INNOVATIVE"
   def __init__(self):
      pass
print(I_HUB1.school_name)
```

a)Error
b)IHUB INNOVATIVE
c)None
d)IHUB

71)

```
import time
class I_HUB1:
   def __init__(self):
      self.id=1001
   def m1(self):
      self.id=1002
      print(self.id)
i1=I_HUB1()
i1.id=1003
i1.m1()
```

a)1001
b)1003
c)1002

d)Error

72)
```
import time
class I_HUB1:
    @classmethod
    def m1(cls):
        self.esal=17500
i1=I_HUB1()
i1.m1()
```

a)17500
b)NameError
c)AttributeError
d)None

73)
```
import time
class I_HUB1:
    @staticmethod
    def m2(obj1,obj2):
        return obj1//obj2
i1=I_HUB1()
print(i1.m2(1000,0))
```

a)return not allowed in oops
b)Attribute Error
c)Error
d)Exception

74)
```
import time
class I_HUB1:
    @staticmethod
    def m1(self):
        pass
i1=I_HUB1()
i1.m1()
print()
time.sleep(2)
print("End of an application ...")
```

a)SyntaxError
b)TypeError
c)Exception
d)None

75)
```
import time
```

```
class I_HUB1:
    def m1(self):
        print("Im main_method to write the business logic")
i1=I_HUB1()
I_HUB1.m1()
```

a)Error
b)SyntaxError
c)TypeError
d)Im main_method to write the business logic

76)

```
import time
class I_HUB1:
    def __init__(self):
        I_HUB1.a=1003
i1=I_HUB1()
print(i1.__dict__)
```

a){'a':1003}
b){}
c)Error
d)Exception

77)

```
import time
def Test_Case1(a=100,b=200,c=300):
    return(a,b,c)
if(__name__=="__main__"):
    print(Test_Case1("ONE HUNDRED","TWO HUNDERED","THREE HUNDRED"))
```

a)(100,200,300)
b)('ONE HUNDRED', 'TWO HUNDERED', 'THREE HUNDRED')
c)Error
d)Empty

78)

```
import time
class I_HUB1:
    def __init__(self):
        obj1=12000
    def m1(self):
        obj1=14000
        return obj1
i1=I_HUB1()
print(i1.m1())
```

a)12000
b)Empty
c)14000
d)Cannot upate local variable

79)
```
import time
class QT1:
    def __init__(self,eid):
        self.eid=eid
q1=QT1(1010)
print(q1)
```

a)1001
b)<__main__.QT1 object at 0x008DFCF0>
c)Error
d)Empty

80)
```
import time
class QT1:
    def __init__(self,a,b,c):
        self.a=a
        self.b=b
        self.c=c
        return(self.a+self.b+self.c)
i1=QT1(1,2,3)
print(i1)
```

a)6
b)TypeError
c)Error
d)return not acceptable

81)
```
import time
class QT1:
    def __init__(self,a,b,c):
        self.a=a
        self.b=b
        self.c=c
    def m1(self):
        return(self.a+self.b+self.c)
i1=QT1(1,2,3)
print(i1.m1())
```

a)Error
b)IdentitationError
c)6
d)None

82)
```
import time
class QT1:
    def __init__(self,a,b,c):
        self.a=a
```

```
        self.b=b
        self.c=c
    def m1(self):
        return(self.a+self.b+self.c)
i1=QT1([1,2,3],[4,5,6],[7,6,9])
print(i1.m1())
```

a)[1, 2, 3, 4, 5, 6, 7, 6, 9]
b)Cannot pass list as argument to non static method
c)Error
d)None

83)

```
import time
class QT1:
    def __init__(self,a,b,c):
        self.a=a
        self.b=b
        self.c=c
    def m1(self):
        return(self.b*5)
i1=QT1([1,2,3],[4,5,6],[7,6,9])
print(i1.m1())
```

a)[4,5,6],[4,5,6],[4,5,6],[4,5,6],[4,5,6]
b)[4, 5, 6, 4, 5, 6, 4, 5, 6, 4, 5, 6, 4, 5, 6]
c)Erorr
d)None

84)

```
import time
class QT1:
    def __init__(self,a,b,c):
        self.a=a
        self.b=b
        self.c=c
    def m1(self):
        return(self.b.items())
i1=QT1({0:100,1:200,2:300},{100:1000,200:2000,300:3000},{1000:10000,2000:20000,3000:30000})
print(i1.m1())
```

a)Error
b)None
c)dict_items([(100, 1000), (200, 2000), (300, 3000)])
d){}

85)

```
import time
class QT1:
    def __init__(self,a,b,c):
        self.a=a
        self.b=b
```

```
        self.c=c
    def m1(self):
        return(self.a+self.b+self.c)
i1=QT1(100+200j,200+300j,1000+2000j)
print(i1.m1())
```

a)(100+200j)
b)(1300+2500j)
c)Empty
d)Error

86)

```
import time
class QT1:
    def __init__(self,a,b,c):
        self.a=a
        self.b=b
        self.c=c
    def m1(self):
        return(self.b*self.c)
i1=QT1(4,5,'A')
print(i1.m1()
```

a)4,5,'A'
b)AAAAA
c)Error
d)None

87)

```
import time
class I_HUB1:
    def __init__(self):
        print("Hello how r u ?")
    def __del__(self):
        print("Clean memory block....")
i1=I_HUB1()
i2=i1
i3=i2
del i1
time.sleep(1)
print("i1 ref is gone and object is there")
print()
time.sleep(2)
print('End of an application ...')
```

a)
Hello how r u ?
i1 ref is gone and object is there
End of an application ...
Clean memory block....

b)
Hello how r u ?

End of an application ...
Clean memory block....
i1 ref is gone and object is there

c)Error
d)None

88)
```
import time
class I_HUB:
    print("I_HUB Compnay")
    class QT1:
        print("QT1 company")
        def m1(self):
            print("ONE TWO THREE")
i1=I_HUB().QT1().m1()
```

a)
I_HUB Compnay
QT1 company
ONE TWO THREE

b)I_HUB Company
c)Error
d)None

89)
```
import time
class Employee:
    def __init__(self):
        self.name="Arjun Reddy"
        self.dob=self.DOB()
    def m1(self):
        print('Employee_Name is:',self.name)
    class DOB:
        def __init__(self):
            self.day=16
            self.month=7
            self.year=1990
        def m2(self):
            print("Date of birth is:{}/{}/{}".format(self.day,self.month,self.year))
e1=Employee()
e1.m1()
```

a)Name the employee is:Arjun Reddy
b)Name the employee is:Arjun Reddy
  Date of birth is:16/7/1990
c)There is no object creation
d)Error

90)

  test1.py

```
import time
print("AI .....)
```

test2.py

```
import time
import test1
import test1
import test1
print("ML....")
```

a)AI,AI,AI,ML
b)Error
c)AI,ML
d)None

91)

```
import time
number=eval(input('Enter the number:'))
if(number%2==0):
   print(number)
   break
else:
   print(number)
print()
time.sleep(2)
print("End of an application ...")
```

a)even number
b)break outside the loop
c)odd number
d)None

92)

```
import time
for x in range(1,11):
    if(x%2==0):
        continue
    else:
        print(x)
```

a)Even number
b)Error
c)Odd number
d)None

93)Can we use for with else in python

a)Yes
b)No
c)Python deos not support

d)a,b as per the application reqn.

94)

```
    import time
    l1=[1,2,3,4,7,10,150,11,12,13,15]
    for x in l1:
          if(x>148):
             print("Stop")
              break
         print(x)
```

a)Script executed
b)Error
c)None
d)Emty

95)what is __name__=="__main__:

a)It is main method to executed set of blocks
b)Functions
c)classes
d)Objects

96)

```
     import time
class I_HUB1:
   def m1(self):
      print("QT .....")
class I_HUB2(I_HUB1):
   def m2(self):
      print("QT1111")
i1=I_HUB1()
i1.m1()
i1.m2()
```

a)QT & QT1111
b)QT & m2 is not define
c)Error
d)There is no proper inheritance

97)

```
import time
class A:
    eid=1001
    def __init__(self):
      self.name="Ajay"
class B(A):
   def m1(self):
      print(super().eid)
      print(super().ename)
b1=B()
b1.m1()
```

a)Error
b)None
c)1001,AttributeError
d)1001

98)

```
import time
a=1
while(a<=10):
    b=1
    while(b<=10):
        print(a*b,end=" ")
        b+=1
    print()
    a+=1
print()
time.sleep(2)
print("End of an application ...")
```

a)123445566666
b)1 to 10 table in row format
c)1 to 10 table in column format
d)Missing values

99)

```
    import time
import os
try:
    print("Core Python")
    print(100//0)
    os._exit(10000)
except:
    print("Advance Python")
else:
    print("Django")
finally:
    print("FastAPI")
print()
time.sleep(2)
print("End of an application ...")
```

a)Core Python
b)Core Python,Advance Python
c)Core Python Advance Python FastAPI
d)Only finally block

100)

```
import time
import os
try:
```

```
    print("Core Python")
except:
    print("Advance Python")
    os._exit("Microsoft")
else:
    print("Django")
finally:
    print("FastAPI")
print()
time.sleep(2)
print("End of an application ...")
```

a)Error
b)Core Python,Django,FastAPI
c)try block
d)try,except block