1. Rezultatul codului de mai jos este?

```
num_calls = 0


def exercitiu(x):
    global num_calls
    num_calls = 3
    num_calls += 1
    return x * x

print(exercitiu(4))
```

a. 9
b. 16
c. 4
d. error

2. Rezultatul codului de mai jos este:

```
x = 1


def f():
    return x


print(x)
print(f())
```

a. error
b. 1
c. 1 1
d. 0 1

3. Rezultatul codului este:

```
x = [1, 2, "hello", "world", ["another", "list"]]
print(len(x[3]))
```

a. TypeError: object of type 'int' has no len()
b. 5
c. 0
d. 2

4. Rezultatul codului este:
x = (1, 2, 3)

x[1] = 4

a. x = (1, 2, 4)
b. x = (1, 2, 3)
c. x = [1, 2, 3]
d. TypeError

5. Rezultatul codului este:

```
a = [1, 2, 3]
b = [4, 5]

print(a+b*3)
```

a. [1, 2, 3, 4, 5]
b. [1, 2, 3]
c. error
d. [1, 2, 3, 4, 5, 4, 5, 4, 5]

6. Rezultatul codului este:

```
x = [1, 2, 3, 4]
print(x[-1:])
```

a. [1, 2, 3]
b. [4]
c. [1, 2, 3, 4]
d. [3, 2, 1]

7. Rezultatul codului este:

```
x = [0, 1, [2]]
x[2][0] = 3
x[2].append(4)
x[2] = 2
print(x)
```

a. [0, 1, 3]
b. [1, 3, 2]
c. [0, 1, 2]
d. error

8. Rezultatul codului este:

```
def exercitiu(i):
    for i in range(i):
        return i

x = exercitiu(3)
print(x)
```

a. error
b. 0 1 2
c. 3
d. 0

9. Rezultatul codului este:

```
a = range(10)
y = [x*x for x in a if x%2 == 0]
print(y)
```

a. [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
b. [2, 4, 6, 8]
c. [0, 4, 16, 36, 64]
d. [0, 2, 16, 36, 64]

10. Rezultatul codului este:

```
def make_account():
    return {'balance': 0}


def deposit(account, amount):
    account['balance'] += amount
    return account['balance']

a = make_account()
print(deposit(a, 10))
```

a. error
b. 0
c. 10
d. None

11. Rezultatul codului este:

```
class BankAccount:
    def __init__(self):
        self.balance = 0

    def deposit(self, amount):
        self.balance += amount
        return self.balance

a = BankAccount()
b = BankAccount()
print(a.deposit(100))
```

a. 0
b. None
c. error
d. 100

12. Rezultatul codului este:
"foo" + 2

a. cannot concatenate 'str' and 'int' objects
b. name 'foo' is not defined
c. foo2
d. integer division or modulo by zero

13. Rezultatul codului este:

```
try:
  print("a")
```

```
except:
  print("b")
else:
  print("c")
finally:
  print("d")
```

a. a b c d
b. a b c
c. a c d
d. error

14. Rezultatul codului este:

```
for k in {"x": 1, "y": 2}:
    print(k)
```

a. {“x”:1, “y”: 2}
b. x y
c. 1 2
d. error

15. Rezultatul codului este:

```
print(list("python"))
```

a. [‘python’]
b. ‘p’, ‘y’, ‘t’, ‘h’, ‘o’, ‘n’
c. error
d. [‘p’, ‘y’, ‘t’, ‘h’, ‘o’, ‘n’]

16. Rezultatul codului este:

```
def func(*args):
    return 3 + len(args)


print(func(4, 4, 4))
```

a. 4
b. error
c. 6
d. 15

17. Rezultatul codului este:

```
count = (3, 2, 5, 4)
while len(count) < 5:
    count0 = count[0]+1
    print("Hello Geek")
```

a. Hello Geek
b. loop infinit in care se afiseaza Hello Geek
c. None
d. error

18. Rezultatul codului este:

```
def exercitiu(var):
    for letter in 'geeksforgeeks':
        if letter == 'e' or letter == 's':
            continue
        print('Current Letter :', letter)
        var = 10
        return var

print(exercitiu(20))
```

a. 10
b. None
c. Current Letter : g
   10
d. Current Letter: g
   20

19. Care este diferenta intre liste si tupluri:
a. Listele sunt mutabile, tuplurile sunt imutabile
b. Listele sunt imutabile, tuplurile sunt mutabile
c. Listele nu sunt indexabile, tuplurile sunt indexabile
d. niciuna din variantele de mai sus

20. Rezultatul codului este:

```
def f(a,list=[]):
    for i in range(a):
        list.append(i*i)
    print(list)

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

a. [0, 1, 4] [1,2,3,0,1] [0, 1, 4, 0,1]

b. [0, 1, 4] [1,2,3,0,1] [0, 1]
c. Error
d. [0, 1, 4]

21. Rezultatul codului este:

```
list = ['1', ‘2', '3', '4', '5']
print (list[12:])
```

a. index error
b. []
c. None
d. list este cuvant rezervat

22.
Ce returneaza interpretorul la rularea obj.Met1("Salut Lume")?

```
>>> class ClasaMea:
        def Met1(self,a):
                global var1
                var1=a


>>> obj=ClasaMea()
>>> obj.Met1("Salut Lume")
```

Select one:

a. Nu returneaza eroare.

Nu returneaza/afiseaza nimic, dar va crea variabila var1 ce va stoca sirul de caractere "Salut Lume"

b. Returneaza eroare. Nu pot folosi conceptul de global intr-o metoda a unei clase.

c. Nu returneaza eroare.

Returneaza/afiseaza sirul de caractere "Salut Lume"

d. Returneaza eroare. Nu pot folosi argumente (parametri de intrare) global intr-o metoda a unei clase.

23.
Cum creez atributul x ce stocheaza valoarea 77777 al obiectului obj_test123, obiect ce apartine clasei definite mai jos (Test123)? Obiectul nu este creat in avans.

```
>>> class Test123():
        def __str__(self):
                self.x=77777
                return str(self.x)


>>>
```

Select one:

a. obj_test123=test123()

obj_test123.y=77777

b. test123.x=77777

c. obj_test123=Test123()

print obj_test123

d. Nu pot crea atributul x deoarece interpretorul returneaza eroare

24.
Ce va returna Interpretorul daca apelez obiect1.Ad(1,2,3) ca in imaginea de mai jos?

```
>>> class X(object):
        """Clasa adunare"""
        def Ad(self,a,b,c):
                return self.a+self.b+self.c


>>> obiect1=X()
>>> obiect1.Ad(1,2,3)
```

Select one:

a. Interpretorul va returna numarul integer 6

b. Interpretorul va returna numarul float 6.0

c. Obiectul obiect1 nu exista deci Interpretorul ne va ridica eroare

d. Variabilele locale self.a, self.b si self.c nu sunt definite deci Interpretorul ne va ridica eroare

25.
Cum creez atributul y cu valoarea 77777 al obiectului obj_test123, obiect ce apartine clasei definite mai jos? Obiectul nu este creat.

```
>>> class test123(object):
        def rezultat(self):
                self.y=77777
```

Select one:

a. obj_test123=test123()

obj_test123.rezultat()

b. obj_test123=test123()

c. Nu pot crea atributul y deoarece interpretorul returneaza eroare

d. test123.y=77777

26.
Ce va returna Interpretorul la rularea obiect1=X() ca in imaginea de mai jos? Cum creez un obiect in Python? Alegeti doua variante.

```
>>> class X(object):
        """Clasa adunare"""
        def Ad():
                print "Imi place Python!"


>>> obiect1=X()
```

Select one or more:

a. Interpretorul va returna eroare deoarece metoda Ad ce creaza un obiect nu are nici un parametru.

b. Interpretorul va returna sirul de caractere <<Imi place Python!>>.

c. Pentru a crea un obiect al clasei X trebuie sa apelam Ad().

d. Interpretorul nu va returna nimic si va crea obiectul obiect1.

27.

Ce returneaza interpretorul la rularea codului de mai jos?

```
>>> class Test123():
        def __init__(self,x):
                self.x=x


>>> obj_test123= Test123()
```

Select one:

a. Interpretorul returneaza eroare deoarece metoda speciala __init__ are doi parametrii (self,x), dar la apelare noi transmitem doar un singur parametru.

b. Interpretorul nu returneaza eroare. Interpretorul returneaza valoarea x.

c. Interpretorul nu returneaza eroare. Interpretorul nu returneaza nimic deoarece cream un obiect.

d. Interpretorul returneaza eroare deoarece metoda speciala __init__ nu are return.

28

Cum creez atributul x ce stocheaza valoarea 77777 al obiectului obj_test123, obiect ce apartine clasei definite mai jos (Test123)? Obiectul nu este creat in avans.

```
>>> class Test123():
        def __init__(self):
                self.x=77777


>>>
```

Select one:

a. obj_test123=test123()

obj_test123.y=77777

b. test123.x=77777

c. obj_test123=Test123()

d. Nu pot crea atributul x deoarece interpretorul returneaza eroare

29. Cate metode ar avea un obiect al clasei Test2 dupa creare ?

```
>>> class Test1():
        def Met1(self):
                self.unu="unu"


>>> class Test2(Test1):
        def Met2(self):
                self.unu="doi"


>>> class Test3(Test2):
        def Met2(self):
                self.doi="doi"
```

>>>obj=Test2()

Select one:

a. Doua metode

b. Zero metode

c. O metoda

d. Trei metode

30. Ce returneaza interpretorul la apelarea obj.Met1() ?

```
>>> class Test1():
        dynamic="ceva"
        def Met1(self):
                return Test1.dynamic.upper()


>>> obj=Test1()
>>> obj.Met1()
```

Select one:

a. Va returna eroare deoarece nu putem folosi sintaxa Test1.dynamic.upper() Forma corecta este self.dynamic.upper()

b. Va returna sirul de caractere "CEVA"

c. Va returna eroare deoarece nu putem returna un sir de caractere din metoda Met1. Doar metoda speciala __str__ poate returna un sir de caractere

d. Va returna sirul de caractere "ceva"

1. In this little assignment you are given a string of space separated numbers, and have to return the highest and lowest number.

Example:

high_and_low("1 2 3 4 5") # return "5 1"

high_and_low("1 2 -3 4 5") # return "5 -3"

high_and_low("1 9 3 4 -5") # return "9 -5"

Notes:

All numbers are valid Int32, no need to validate them.

There will always be at least one number in the input string.

Output string must be two numbers separated by a single space, and highest number is first.