Consider the below given Python code.

```
def func(my_lst,var1):
    new_lst=[]                              #Line1
    for num in my_lst:                      #Line2
        if(num%var1==0):                    #Line3
            new_lst.append(num//var1)       #Line4
        else:                               #Line5
            new_lst.append(0)               #Line6
    return new_lst                          #Line7
```

Which among the following test data when passed to function, func will cover the code written in Line 6.

Choose TWO CORRECT options.

- [ ] my_lst=[5,10,15,20,25], var1=5
- [x] my_lst=[2,5,8,11,14,17,21], var1=2
- [ ] my_lst=[18,24,30,36,42], var1=3
- [x] my_lst=[13,17,23,27,33,37], var1=7

```
def func(var1,var2,var3):
    if(var1>var2):                  #Line1
        return("1")                 #Line2
    elif(var2>var3):                #Line3
        if(var1>var3):              #Line4
            return("2")             #Line5
        else:                       #Line6
            return("3")             #Line7
    else:                           #Line8
        return("4")                 #Line9
```

Identify the line numbers that will be covered when the function, func is invoked using the below test data.

var1=3, var2=5 and var3=1

- Line1, Line3, Line4, Line5
- Line1, Line 2, Line6, Line7, Line8, Line9
- It will cover all the lines
- Line3, Line4, Line5

```
list1=[10,20,0,40,0]
def test():
    try:
        var1=3
        if(list1[var1]/list1[var1+1]>1):
            value=var1+1
    except ZeroDivisionError:
        print("1")
    except IndexError:
        print("2")
    finally:
        print("4")
    print("5")
test()
```

- 1 5
- 1 4 5
- 2 4 5

Choose the statements which are correct with respect to the below Python code.

Choose THREE CORRECT options.

```
def func1(arg1,*arg2):
    for num in arg2:
        if(arg1>=num):
            return num
    return 0
def func2(arg3,arg4=10):
    if(arg3<=arg4):
        return arg3
    return arg4

def func3(arg5,arg6):
    if(arg5!=arg6):
        return False
    return True
res2=func2(1)
res1=func1(res2,1,1,2,5,7,8)
print(func3(arg6=10,agr5=res2))
```

1.We cannot have a positional argument after arg2 in func1

```
def func3(arg5,arg6):
    if(arg5!=arg6):
        return False
    return True
res2=func2(1)
res1=func1(res2,1,1,2,5,7,8)
print(func3(arg6=10,agr5=res2))
```

1.We cannot have a positional argument after arg2 in func1
2.arg4 is a positional argument
3.arg5 and arg6 are keyword arguments
4.arg1 and arg3 are positional arguments
5.*argument_name indicates default argument

- [x] 1
- [ ] 2
- [x] 3
- [x] 4

Reset Save

What is the output of the below code snippet?

```
def func1(var1=1,var2=2):
    print(var1,end=" ")
    print(var2,end=" ")
func1(100,None)
func1(var2=20,var1=10)
func1(var2=10)
```

- 100 2 10 20 1 10
- 100 None 10 20 1 10
- Error
- 100 None 1 2 1 10

Reset Save

Consider the below Python code:

```
def fun(var1):
    if var1<1:
        return 0
    elif var1%2==0:
        return fun(var1-1)
    else:
        return var1+fun(var1-2)
```

From the given options, identify the functions calls which would return the value 25.
Choose TWO CORRECT options.

- [ ] fun(11)
- [ ] fun(12)
- [x] fun(9)
- [x] fun(10)

Consider the below given Python code.

```
def calculate(var1,var2):
    while(var1!=var2):
        if(var1>var2):
            return calculate(var1-var2, var2)
        else:
            return calculate(var1, var2-var1)
    return var1
```

From the given options, identify the function calls which would result in the same value.
Choose TWO CORRECT options.

- [x] calculate(10,55)
- [ ] calculate(60,30)
- [ ] calculate(27,47)
- [x] calculate(45,20)

What is the output of the below code snippet?

```
try:
    tupl1 = ([1,2], [3,4])
    list1 = [(1,2), (3,4)]
    list2 = tupl1[0]
    list2[0] = 5
    list1[1] = (6,7)
    print(tupl1, list1)
except TypeError:
    print("ERR")
```

- ([5, 2], [3, 4]) [(1, 2), (6, 7)]
- ([1,2], [3,4]) [(1,2), (6,7)]
- ValueError
- ([1,2], [3,4]) [(1,2), (3,4)]

Reset Save

What is the output of the below code snippet?

```
var1,var2=10,40
def func1(var1):
    global var2
    var1,var2=20,50
    print(var1,end=" ")
    print(var2,end=" ")
func1(30)
print(var1,end=" ")
print(var2,end=" ")
```

- 20 50 10 50
- 20 50 10 40
- 30 50 10 50
- Error

Reset Save

What would be the output of the below Python code?

```
def func(var1,var2):
    try:
        var3=(int)(var1)
        var2=var3+"A"
        print(var2)
    except TypeError:
        print("T")
    finally:
        print("IF")
try:
    func('R',13)
except ValueError:
    print("V")
finally:
    print("OF")
```

A) T
IF
V
OF

A) T
IF
V
OF

B) V
OF

C) IF
V
OF

D) T
IF
V

- ○ A
- ○ B
- ◉ C
- ○ D

Reset Save

What is the output of the below code snippet?

```
def func1():
    try:
        1/0
        return 1
    except ZeroDivisionError:
        "ABC"+1
        return 2
    finally:
        int('A')
        return 3
try:
    result=func1()
    print(result)
except:
    print(4)
```

- 1
- 2

```
            "ABC"+1
            return 2
        finally:
            int('A')
            return 3
try:
    result=func1()
    print(result)
except:
    print(4)
```

- 1
- 2
- 4
- 3

Reset Save

What is the output of the below code snippet?

```
def func1():
    try:
        dict1 = {"IN": "India", "US": "United States"}
        del dict1["IN"]
        value = 100 // (len(dict1) - 1)
        print(value)
    except ZeroDivisionError:
        print("ZD", end=" ")
        value = int(dict1[0])
    except KeyError:
        print("KE", end=" ")
    finally:
        print("FI", end=" ")
try:
    func1()
    print("TR")
except:
    print("CA")
```

ZD FI CA

```
        value = 100 // (len(dict1) - 1)
        print(value)
    except ZeroDivisionError:
        print("ZD", end=" ")
        value = int(dict1[0])
    except KeyError:
        print("KE", end=" ")
    finally:
        print("FI", end=" ")
try:
    func1()
    print("TR")
except:
    print("CA")
```

- ZD FI CA
- ZD FI
- ZD CA
- FI TR

Reset Save