In [ ]:
```
# 1st Midterm
```

In [94]:
```
# Exam first part

# Test 1.

#What is the value of the expression?

lst=[32,33,16,2,49,56,72,69,95,13,18,63]

lst[4 if len(lst)%2==1 else 5]  # this lst[i] is to list the number
```

Out[94]:
```
56
```

In [95]:
```
# Test 2.

# Suppose the value of the variables res is 27 before the following
# What is the value of res after it has run?

res=27
for i in range(5,7):
# This allow to the loops below to run 2 times coz of 7-5
# Here, j runs from 1 to 4 two times. i,e. 27+1+2+3+4+1+2+3+4=47

    for j in range(1,5):
        res = res + j
        print (res)
```

```
28
30
33
37
38
40
43
47
```

In [96]:
```
# Test 3.

# Suppose that the value of the variable res is 11 before the follo
# What is the value of res after it has run?

res =11
for i in range(5,7):
    for j in range(1,5):
        if j>3: # j in 1,2,3
            break
        res = res+j  # res = 11+2*(1+2+3)= 23
print(res)
```

```
23
```

In [97]:
```
# Second part


# Test 1.

# Write a function my_len() of one argument, that does what len() d
# (so you can't use the latter): given a list, it returns its lengt
# my answer

def my_len(a):
    cnt=0
    for i in a:
        cnt+=1
    return cnt


my_len([1,2,4,5])
```

Out[97]:
```
4
```

In [98]:
```
# Test 2.

# Write a function add_lists() of two arguments, it will be called

def add_lists(a,b):
    nl=[]
    for i in range(len(a)):
        nl.append(a[i]+b[i])
    return nl

def add_lists1(a,b):
    res_list=[]
    for i in range(0,len(a)):
        res_list.append(a[i] + b[i])
    return res_list

# both pieces of code are correct

add_lists1([1,2,4],[1,2,3])
```

Out[98]:
```
[2, 4, 7]
```

In [99]:
```
# Test 3.

# Write a function swap() of one argument, a list that is of even l
# from the original in that each of its member of even index is int
# with the member of odd index that follows it.


def swap(ls):
    for i in range(0,len(ls),2):
        t = ls[i]
        ls[i]=ls[i+1]
        ls[i+1]=t
    return ls
swap(list(range(10)))
```

Out[99]:
```
[1, 0, 3, 2, 5, 4, 7, 6, 9, 8]
```

In [ ]:
```
# 2nd Midterm
```

In [ ]:
```
# Part 2

# tuple, f-string, array, file operation

# Exercise 1

def concat_files(in_file1,in_file2,out_file):

    with open(in_file1) as file:
        data1 = file.read()
    with open(in_file2) as file:
        data2 = file.read()

    data1 += "\n"
    data1 += data2

    with open(out_file, 'w') as file:
        file.write(data1)


concat_files("in_file1.txt","in_file2.txt","out_file.txt")
```

In [ ]:
```
# Exercise 2

def longest_line(text_file):

    print(max(open(text_file), key=len))
```

In [43]:
```
# Exercise 3

def to_dict(list1,list2):
    return dict(zip(list1, list2))

print(to_dict(list(range(5)),list(range(10,15))))
```

```
{0: 10, 1: 11, 2: 12, 3: 13, 4: 14}
```

In [100]:
```
# Part 1

#Test 1.

#What is the value of the variable b

a,_,b,c=(1,0,2,3)
b
```

Out[100]: 2

In [101]:
```
# Test 2.

# Suppose the value of the variable index is [4,0,1,2,3], and the v
# What is the value of the following expression? data[index[2]]

index=[4,0,1,2,3]
data=[3,1,2,0,4]
data[index[2]]
```

Out[101]: 1

In [102]:
```
# Test 3.

# Which is the correct output?

print(f'|{3.14:8.2f}|');print('|12345678|')
```

```
|    3.14|
|12345678|
```

In [49]:
```
# 3rd Midterm
```

In [104]:
```
# Part 1

# Test 1.

#What is the value of the following expression?

sum([sum([x%y for x in range(3,7) if x**2<35]) for y in range(4,6)]

# so the code here will be 3mod4+4mod4+5mod4+3mod5+4mod5+5mod5= 3+0
```

Out[104]: 11

In [103]:
```
# Test 2.

#What is the return value of lists((8,10,1,3,8,4),4)?

def lists(m,n):
    try:
        m[n]=2
        return m[n]
    except:
        return m[n]

lists((8,10,1,3,8,4),4)
```

Out[103]: 8

In [53]:
```
# Test 3. Let f be defined as follows:

def f(*l,m=2):
    return len(l)*m  # be careful here, * stands for multiplication
# What is the return value of the call

f(8,10,1,3,8,4)
```

Out[53]: 12

In [107]:
```
# Part 2

# Exercise 1.

# Define a function apply() that can be called with n arguments, wh
# of n-1 arguments, and apply() applies this function to the rest o
# of this application. For example: apply(lamda x,y:x+y,2,3),apply(

def apply(fn,*args):
    return fn(*args)

apply(lambda x,y:x+y,2,3),apply(lambda:42)
```

Out[107]: (5, 42)

In [106]:
```
# Exercise 2.

# Define a function appall() that can be called with any number of
# first, are all functions of one argument.
# appall(c,f1,f2,...,fn)
# should return the list
# [f1(c),f2(c),...,fn(c)]

def appall(fn,*args):
    return [f(fn) for f in args]

appall(5,lambda x:2*x, lambda x:x**2,lambda x: -x)
```

Out[106]: [10, 25, -5]

In [105]:
```
# Exercise 3.

# Define a function deeprev() whose only argument will be list, who
#lists whose members are integers and lists, whose members are numb
# should return the reverse list but in the extend sense that every
#is also reveresed.

def deeprev(x):
    if not isinstance(x,list):
        return x
    return [deeprev(i) for i in x][::-1]

deeprev([8,3,[1,3,[5,6,7],6],4,2,1])
```

Out[105]: [1, 2, 4, [6, [7, 6, 5], 3, 1], 3, 8]

In [ ]: