# Lab – 09 Namespace and Scope, Module, Packages, Try Except Structure

Tasks:

1. Given the following directory structure of package and modules with source codes of some of the files are given below:



**Create a file called program\_q1.py first** and then create a folder called calculation in **Visual Studio Code** and the other corresponding file as shown in the above diagram.

1.

2.

**circle.py**

import math

def circle( radius ):

area = radius \*\* 2 \* math.pi

surface\_area = 4 \* math.pi \* radius \*\* 2

volume = 4 / 3 \* math.pi \* radius \*\* 3

return area, surface\_area, volume

print ("The following codes are used to test the circle library")

area, surface\_area, volume = circle(5)

print ("Area of circle is", area)

print ("Surface area of sphere is", surface\_area)

print ("Volume of sphere is", volume)

**rectangle.py**

def rectangle( length, width, height ):

area = length \* width

surface\_area = ( length \* width + length \* height + width \* height ) \* 2

volume = length \* width \* height

return area, surface\_area, volume

print ("The following codes are used to test the rectangle library")

area, surface\_area, volume = rectangle(2,4,6)

print ("Area of rectangle (length \* width) is", area)

print ("Surface area of rectangle is", surface\_area)

print ("Volume of rectangle is", volume)

**program\_q1.py**

import calculation

l = float ( input ( "Enter the length of rectangle: " ) )

w = float ( input ( "Enter the width of rectangle: " ) )

h = float ( input ( "Enter the height of rectangle: " ) )

area\_rectangle, surface\_area\_rectangle, volume\_rectangle = rectangle ( l, w, h )

1. Execute the circle and rectangle module directly and note the output of circle and rectangle modules in calculation package. (No need to answer this question)
2. The module program.py is now going to use both circle.py and rectangle.py modules in calculation package. **Modify program\_q1.py only** so that program\_q1.py can make use of rectangle module in calculation package.

Sample Output:

The following codes are used to test the rectangle library

Area of rectangle (length \* width) is 8

Surface area of rectangle is 88

Volume of rectangle is 48

Enter the length of rectangle: 2

User’s input

Enter the width of rectangle: 2

Enter the height of rectangle: 2

Area of rectangle (length \* width) is 4.0

Surface area of rectangle is 24.0

Volume of rectangle is 8.0

Answer:

import calculation.rectangle

calculation.rectangle.rectangle(2,4,6)

l = float ( input ( "Enter the length of rectangle: " ) )

w = float ( input ( "Enter the width of rectangle: " ) )

h = float ( input ( "Enter the height of rectangle: " ) )

area\_rectangle, surface\_area\_rectangle, volume\_rectangle = calculation.rectangle.rectangle( l, w, h )

print("Area of rectangle (length \* width) is",area\_rectangle)

print("Surface area of rectangle is ",surface\_area\_rectangle)

print("Volume of rectangle is ",volume\_rectangle)

1. In addition to the functions of calculating the area, surface area and volume of rectangle, the testing codes in rectangle module are also executed. Try to **modify rectangle module only** so that the testing codes inside rectangle module will not be executed when called by another module, while those codes will still be executed when user execute the rectangle module.

Sample Output:

Enter the length of rectangle: 2

User’s input

Enter the width of rectangle: 2

Enter the height of rectangle: 2

Area of rectangle (length \* width) is 4.0

Surface area of rectangle is 24.0

Volume of rectangle is 8.0

Answer:

def rectangle( length, width, height ):

    area = length \* width

    surface\_area = ( length \* width + length \* height + width \* height ) \* 2

    volume = length \* width \* height

    return area, surface\_area, volume

if \_\_name\_\_ == "\_\_main\_\_":

    print ("The following codes are used to test the rectangle library")

    area, surface\_area, volume = rectangle(2,4,6)

    print ("Area of rectangle (length \* width) is", area)

    print ("Surface area of rectangle is", surface\_area)

    print ("Volume of rectangle is", volume)

1. **Modify program\_q1.py** **only** to include the circle module to calculate the area of circle, surface area and volume of sphere.

Sample Output:

The following codes are used to test the circle library

Area of circle is 78.53981633974483

Surface area of sphere is 314.1592653589793

Volume of sphere is 523.5987755982989

Enter the length of rectangle: 2

User’s input

Enter the width of rectangle: 2

Enter the height of rectangle: 2

Enter the radius of circle: 2

Area of rectangle (length \* width) is 4.0

Surface area of rectangle is 24.0

Volume of rectangle is 8.0

Area of circle is 12.566370614359172

Surface area of sphere is 50.26548245743669

Volume of sphere is 33.510321638291124

Answer:

import calculation.circle

import calculation.rectangle

l = float ( input ( "Enter the length of rectangle: " ) )

w = float ( input ( "Enter the width of rectangle: " ) )

h = float ( input ( "Enter the height of rectangle: " ) )

r = float ( input ( "Enter the radius of circle: " ) )

area\_rectangle, surface\_area\_rectangle, volume\_rectangle = calculation.rectangle.rectangle( l, w, h )

area, surface\_area, volume = calculation.circle.circle(r)

print("Area of rectangle (length \* width) is",area\_rectangle)

print("Surface area of rectangle is ",surface\_area\_rectangle)

print("Volume of rectangle is ",volume\_rectangle)

print ("Area of circle is", area)

print ("Surface area of sphere is", surface\_area)

print ("Volume of sphere is", volume)

1. **Modify circle module only** so that testing codes will not be executed when called by program.py module.

Sample Output:

Enter the length of rectangle: 2

User’s input

Enter the width of rectangle: 2

Enter the height of rectangle: 2

Enter the radius of circle: 2

Area of rectangle (length \* width) is 4.0

Surface area of rectangle is 24.0

Volume of rectangle is 8.0

Area of circle is 12.566370614359172

Surface area of sphere is 50.26548245743669

Volume of sphere is 33.510321638291124

Answers:

import math

def circle( radius ):

    area = radius \*\* 2 \* math.pi

    surface\_area = 4 \* math.pi \* radius \*\* 2

    volume = 4 / 3 \* math.pi \* radius \*\* 3

    return area, surface\_area, volume

if \_\_name\_\_ == "\_\_main\_\_":

    print ("The following codes are used to test the circle library")

    area, surface\_area, volume = circle(5)

    print ("Area of circle is", area)

    print ("Surface area of sphere is", surface\_area)

    print ("Volume of sphere is", volume)

1. **Modify program\_q1.py only** using try and except so that the program can handle non-numeric input gracefully.

Sample Output:

Enter the length of rectangle: a

Ask user for length value again

Input a numeric value again

Enter the length of rectangle: 2

Enter the width of rectangle: a

Input a numeric value again

Ask user for width value again

Enter the width of rectangle: 2

Enter the height of rectangle: a

Input a numeric value again

Ask user for height value again

Enter the height of rectangle: 2

Enter the radius of circle: a

Input a numeric value again

Ask user for radius value again

Enter the radius of circle: 2

Area of rectangle (length \* width) is 4.0

Surface area of rectangle is 24.0

Volume of rectangle is 8.0

Area of circle is 12.566370614359172

Surface area of sphere is 50.26548245743669

Volume of sphere is 33.510321638291124

Answers:

import calculation.circle

import calculation.rectangle

while True:

    try:

        l = float ( input ( "Enter the length of rectangle: " ) )

        break

    except ValueError:

        print("Input a numeric value again")

while True:

    try:

        w = float ( input ( "Enter the width of rectangle: " ) )

        break

    except ValueError:

        print("Input a numeric value again")

while True:

    try:

        h = float ( input ( "Enter the height of rectangle: " ) )

        break

    except ValueError:

        print("Input a numeric value again")

while True:

    try:

        r = float ( input ( "Enter the radius of circle: " ) )

        break

    except ValueError:

        print("Input a numeric value again")

area\_rectangle, surface\_area\_rectangle, volume\_rectangle = calculation.rectangle.rectangle( l, w, h )

area, surface\_area, volume = calculation.circle.circle(r)

print("Area of rectangle (length \* width) is",area\_rectangle)

print("Surface area of rectangle is ",surface\_area\_rectangle)

print("Volume of rectangle is ",volume\_rectangle)

print ("Area of circle is", area)

print ("Surface area of sphere is", surface\_area)

print ("Volume of sphere is", volume)

1. Given the following directory structure of package and modules. Source code of program\_q2.py is given below:



**program\_q2.py**

# You are not allowed to modify all the codes in this file

import os

import os\_command.directory

def main():

list\_directory = os\_command.directory.get\_directory\_list(os.getcwd())

list\_file = os\_command.directory.get\_file\_list(os.getcwd())

print("List of directories in path", os.getcwd())

for i in list\_directory:

print (i)

print("List of files in path", os.getcwd())

for i in list\_file:

print (i)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Display all the directories and files in ascending order of current folder by completing the **directory.py** module. A sample output is given below.

Sample Output:

List of directories in path C:\Users\Kelvin\Source Files

calculation

os\_command

List of files in path C:\Users\Kelvin\Source Files

program\_q1.py

program\_q2.py

Answers:

import os

def get\_directory\_list(data):

    content = list()

    for item in os.scandir(data):

        if os.path.isdir(item):

            content.append(item.name)

    return content

def get\_file\_list(data):

    content = list()

    for item in os.scandir(data):

        if os.path.isfile(item):

            content.append(item.name)

    return content

Hints: Create an empty list to store every items from the contents of os.scandir() returns.

By calling os.scandir() function, it contains a list of DirEntry objects (More precisely speaking it should be ScandirIterator object) instead of list of strings. DirEntry objects cannot be sorted even if they are put into an ordinary list. To obtain the string of directory name or file name from the DirEntry object, use **item.name** property, where item is every items in os.scandir() returns.

1. Troubleshoot the following program. The following program is intended to find out the sum, average, minimum and maximum of user provided number of random numbers generated.

import random

def display(list\_numbers):

print ("Sum =", sum(list\_numbers))

print ("Average =", sum(list\_numbers)/len(list\_numbers))

print ("Minimum =", min(list\_numbers))

print ("Maximum =", max(list\_numbers))

def create\_random\_number(no\_of\_random\_numbers):

list\_random\_numbers = list()

for i in range(1, no\_of\_random\_numbers+1):

list\_random\_numbers.append( random.randint(1, 100) )

print (f"Number {i} is {list\_random\_numbers[i]}")

def main():

no\_of\_random\_numbers = input("Number of random numbers: ")

list\_random\_number = create\_random\_number(no\_of\_random\_numbers)

display(list\_random\_number)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Sample Output:

Number of random numbers: 5

Number 0 is 59

Number 1 is 19

Number 2 is 95

Number 3 is 46

Number 4 is 89

Sum = 308

Average = 61.6

Minimum = 19

Maximum = 95

Answer:

import random

def display(list\_numbers):

    print ("Sum =", sum(list\_numbers))

    print ("Average =", sum(list\_numbers)/len(list\_numbers))

    print ("Minimum =", min(list\_numbers))

    print ("Maximum =", max(list\_numbers))

def create\_random\_number(no\_of\_random\_numbers):

    list\_random\_numbers = list()

    for i in range(no\_of\_random\_numbers):

        list\_random\_numbers.append( random.randint(1, 100) )

        print (f"Number {i} is {list\_random\_numbers[i]}")

    return list\_random\_numbers

def main():

    no\_of\_random\_numbers = int(input("Number of random numbers: "))

    list\_random\_number = create\_random\_number(no\_of\_random\_numbers)

    display(list\_random\_number)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

1. Troubleshoot the following program. The following program is intended to calculate the roots of quadratic equation. A sample output is shown below:

def quadratic(a, b, c):

output = ""

if a==0:

output = "This is not a quadratic equation"

else:

discriminant = (b \* b - 4 \* a \* c)

if discriminant > 0: # two roots

x1= (-b + discriminant \*\* 0.5) / (2 \* a)

x2= (-b - discriminant \*\* 0.5) / (2 \* a)

output = "2 roots, x1 = "+str(x1)+", x2 = "+str(x2)

elif discriminant == 0: # one root:

x= -b / (2 \* a)

output = "1 root, x ="+str(x)

else: # no roots

output = "No real roots"

def main():

print (quadrtic(3,-4, 1))

print (quadratic(1,-2))

if \_\_name\_\_ == "\_\_main\_\_":

main()

Output:

2 roots, x1 = 1.0, x2 = 0.3333333333333333

1 root, x =1.0

Answer:

def quadratic(a=0, b=0, c=0):

    output = ""

    if a==0:

        output = "This is not a quadratic equation"

        return output

    else:

        discriminant = (b \* b - 4 \* a \* c)

        if discriminant > 0:   # two roots

            x1= (-b + discriminant \*\* 0.5) / (2 \* a)

            x2= (-b - discriminant \*\* 0.5) / (2 \* a)

            output = "2 roots, x1 = "+str(x1)+", x2 = "+str(x2)

            return output

        elif discriminant == 0: # one root:

            x= -b / (2 \* a)

            output = "1 root, x ="+str(x)

            return output

        else: # no roots

            output = "No real roots"

            return output

def main():

    print (quadratic(3,-4, 1))

    print (quadratic(1,-2))

if \_\_name\_\_ == "\_\_main\_\_":

    main()

1. Troubleshoot the following program. The following program is intended to accumulate all prime numbers in between 1 and user input n. Your program should display all prime numbers from 1 to n and the sum of these numbers. A sample output is shown below:

def is\_prime\_number(number):

if number < 2:

return False

for divisor in range (2, number):

if number % divisor == 0:

break

else:

return True

return False

def main():

n = int(input("Input the value n: "))

total = 0

output\_string = ""

print("The prime numbers are:")

for i in range(1,n):

if is\_prime\_number(i):

output\_string += str(i)+", "

total+=i

print (output\_string)

print("Sum of all prime numbers within 1 and",n,"=",total)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Output:

Input the value n: 101

User’s input

The prime numbers are:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101

Sum of all prime numbers within 1 and 101 = 1161

Answer:

def is\_prime\_number(number):

    #https://www.geeksforgeeks.org/python-program-to-check-whether-a-number-is-prime-or-not/

    if number > 1:

        # Iterate from 2 to n / 2

        for i in range(2, int(number/2)+1):

            # If num is divisible by any number between

            # 2 and n / 2, it is not prime

            if (number % i) == 0:

                break

        else:

            return True

def main():

    n = int(input("Input the value n: "))

    total = 0

    output\_string = ""

    print("The prime numbers are:")

    for i in range(1,n+1):

        if is\_prime\_number(i):

            output\_string += str(i)+", "

            total+=i

    print (output\_string.rsplit(',', 1)[0])

    print("Sum of all prime numbers within 1 and",n,"=",total)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

1. Troubleshoot the following program. The following function is intended to loop through and remove all even numbers from the numbers list, then the number list should be printed out with the items displayed in ascending order.

def remove\_even\_numbers(numbers):

for i in numbers:

if i%2==0:

numbers.remove(i)

numbers.sort()

print (numbers)

def main():

numbers = [

951, 402, 984, 651, 360, 69, 408, 319, 601, 485, 980, 507, 725,

547, 544, 615, 83, 165, 141, 501, 263, 617, 865, 575, 219, 390,

984, 592, 236, 105, 942, 941, 386, 462, 47, 418, 907, 344, 236,

375, 823, 566, 597, 978, 328, 615, 953, 345, 399, 162, 758, 219,

918, 237, 412, 566, 826, 248, 866, 950, 626, 949, 687, 217, 815,

67, 104, 58, 512, 24, 892, 894, 767, 553, 81, 379, 843, 831, 445,

742, 717, 958, 609, 842, 451, 688, 753, 854, 685, 93, 857, 440,

380, 126, 721, 328, 753, 470, 743, 527

]

remove\_even\_numbers(numbers)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Output:

[47, 67, 69, 81, 83, 93, 105, 141, 165, 217, 219, 219, 237, 263, 319, 345, 375, 379, 399, 445, 451, 485, 501, 507, 527, 547, 553, 575, 597, 601, 609, 615, 615, 617, 651, 685, 687, 717, 721, 725, 743, 753, 753, 767, 815, 823, 831, 843, 857, 865, 907, 941, 949, 951, 953]

Answer:

def remove\_even\_numbers(numbers):

    odd\_numbers\_list=list()

    for i in numbers:

        if i%2==0:

            pass

        else:

            odd\_numbers\_list.append(i)

    odd\_numbers\_list.sort()

    print (odd\_numbers\_list)

def main():

    numbers = [

        951, 402, 984, 651, 360, 69, 408, 319, 601, 485, 980, 507, 725,

        547, 544, 615, 83, 165, 141, 501, 263, 617, 865, 575, 219, 390,

        984, 592, 236, 105, 942, 941, 386, 462, 47, 418, 907, 344, 236,

        375, 823, 566, 597, 978, 328, 615, 953, 345, 399, 162, 758, 219,

        918, 237, 412, 566, 826, 248, 866, 950, 626, 949, 687, 217, 815,

        67, 104, 58, 512, 24, 892, 894, 767, 553, 81, 379, 843, 831, 445,

        742, 717, 958, 609, 842, 451, 688, 753, 854, 685, 93, 857, 440,

        380, 126, 721, 328, 753, 470, 743, 527

    ]

    remove\_even\_numbers(numbers)

if \_\_name\_\_ == "\_\_main\_\_":

    main()