Software Development - Session 1B

Python programming exercises

1. Write a program to produce the following output using for loop

|  |  |
| --- | --- |
| + | + |
| \ | / |
| / | \ |
| \ | / |
| / | \ |
| \ | / |
| / | \ |
| + | + |

def symbols():

    sym=["+-------+","\\\t/","/\t\\","\\\t/","/\t\\","\\\t/","/\t\\","+-------+"]

    for i in sym:

        print(i)

symbols()

1. Write a program to produce the following output using for loop

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

def special\_symbols():

    sym=["\*\*\*\*\*\*\*\*\*\*\*","\*\*\*\*\*\*\*\*\*\*\*","\*\*\*\*\*\*\*\*\*\*\*","\*\*\*\*\*\*\*\*\*\*\*","\*\*\*\*\*\*\*\*\*\*\*",]

    for i in sym:

        print(i)

special\_symbols()

1. Complete the code for the following for loop:

for in range(1,7):

//your code here

so that it prints the following numbers, one per line:

for num in range(1, 7):

     print(f"{num: <4} | {num \* 2: <4} | {num \* 15 - 11: <4} | {40 - 10 \* num: <4} | {-7 + 4

for a in range(1, 7):

    print(a)

for b in range(2, 13, 2):

    print(b)

for c in range(4, 80, 15):

    print(c)

for d in range(30, -21, -10):

    print(d)

for e in range(-1, 13, 4):

    print(e)

for f in range(97, 83, -3):

    print(f)

for g in range(-2, 86, 18):

    print(g)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a) | 1 | b) | 2 | c) | 4 | d) | 30 | e) | -7 | f) | 97 | g) | -4 |
|  | 2 |  | 4 |  | 19 |  | 20 |  | -3 |  | 94 |  | 14 |
|  | 3 |  | 6 |  | 34 |  | 10 |  | 1 |  | 91 |  | 32 |
|  | 4 |  | 8 |  | 49 |  | 0 |  | 5 |  | 88 |  | 50 |
|  | 5 |  | 10 |  | 64 |  | -10 |  | 9 |  | 85 |  | 68 |
|  | 6 |  | 12 |  | 79 |  | -20 |  | 13 |  | 82 |  | 86 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. Write a program to produce the following output using for loops. Then use a class constant to make it possible to change the number of lines in the figure.

1

22

333

4444

55555

666666

7777777

for row in range(1,8):

     for col in range(1,row+1):

        print(row,end=" ")

     print()

1. Write a method named pay that accepts two parameters: a real number for a TA's salary, and an integer for the number of hours the TA worked this week. The method should return how much money to pay the TA.

For example, the call

pay(5.50, 6)

should return

33.0.

def pay(salary, hours\_worked):

    if salary < 0 or hours\_worked < 0:

        return "Invalid input. Salary and hours worked must be non-negative."

    # Assuming salary is the hourly rate

    total\_payment = salary \* hours\_worked

return total\_payment

# Test the method

salary\_rate = 20.5  # Example hourly rate

hours\_worked = 20   # Example hours worked

result = pay(salary\_rate, hours\_worked)

print(f"The TA should be paid: ${result:.2f}")

The TA should receive "overtime" pay of 1 ½ normal salary for any hours above 8. For example, the call pay(4.00, 11) should return (4.00 \* 8) + (6.00 \* 3) or 50.0.

def pay(hourly\_rate, hours\_worked):

    if hourly\_rate < 0 or hours\_worked < 0:

        return "Invalid input. Hourly rate and hours worked must be non-negative."

    regular\_hours = min(hours\_worked, 8)

overtime\_hours = max(hours\_worked - 8, 0)

    total\_payment = (hourly\_rate \* regular\_hours) + (1.5 \* hourly\_rate \* overtime\_hours)

return total\_payment

# Test the method

hourly\_rate = 4.00

hours\_worked = 9

result = pay(hourly\_rate, hours\_worked)

print(f"The TA should be paid: ${result:.2f}")

1. Consider the following method for converting milliseconds into days:

// converts milliseconds to days def toDays(millis):

return millis / 1000.0 / 60.0 / 60.0 / 24.0

Write a similar method named area that takes as a parameter the radius of a circle and that returns the area of the circle. For example, the call

area(2.0);

should return

12.566370614359172.

import math

def area(r):

    circle= r\*r\*math.pi

    print(circle)

area(2)

Recall that area can be computed as π times the radius squared and that Python has a constant called math.pi

1. **Copy and paste** the following code into pythons IDLE script environment.

low = 1

high = 1001

sum = 0

for i in range(low,high): sum += i

print("sum = " , sum)

**Modify** the code to use a input to prompt the user for the values of low and high. Below is a sample execution in which the user asks for the same values as in the original program (1 through 1000):

low? 1

high? 1001

sum = 500500

Below is an execution with different values for low and high:

low? 300

high? 5298

sum = 13986903

You should exactly reproduce this format.

low\_q=int(input("low? "))

high\_q=int(input("high? "))

low = low\_q

high = high\_q

sum = 0

for i in range(low,high):

  sum += i

print("sum = " , sum)

1. Write a program using while loop that prompts the user for numbers until the user types 0, then outputs their sum

def ab():

    list1=[]

    x=int(input('enter numbers\n'))

    list1.append(x)

    while x!=0:

        x=int(input('enter numbers\n'))

        list1.append(x)

        if x==0:

            print(list1)

            i=0

            sum=0

            while i<len(list1):

                sum+=list1[i]

                i=i+1

            print('Sum is',sum)

            break

ab()

def abc():

    list1=[]

    x=int(input('enter numbers\n'))

    list1.append(x)

    while x!=0:

        x=int(input('enter numbers\n'))

        list1.append(x)

        if x==0:

            print(list1)

            add=sum(list1)#sum is an inbuilt function

            print('Sum is',add)

            break

abc()

1. Write a program using while loop that prompts the user for numbers until the user types -1, then outputs their sum.

def abc():

    list1=[]

    x=int(input('enter numbers\n'))

    list1.append(x)

    while x!=-1:

        x=int(input('enter numbers\n'))

        list1.append(x)

        if x==-1:

            print(list1)

            add=sum(list1)#sum is an inbuilt function

            print('Sum is',add)

            break

abc()

1. Write a method named repl that accepts a String and a number of repetitions as parameters and returns the String concatenated that many times. For example, the call repl("hello", 3) returns "hellohellohello". If the number of repetitions is 0 or less, an empty string is returned.

def repl(input\_str, repetitions):

    if repetitions <= 0:

        return ""

    result\_str = input\_str \* repetitions

    return result\_str

# Test the method

input\_string = "Hello"

repetition\_count = 3

result = repl(input\_string, repetition\_count)

print(result)

1. Write a method called printRange that accepts two integers as arguments and prints the sequence of numbers between the two arguments, separated by spaces. Print an increasing sequence if the first argument is smaller than the second; otherwise, print a decreasing sequence. If the two numbers are the same, that number should be printed by itself. Here are some sample calls to printRange:

printRange(2, 7)

printRange(19, 11)

printRange(5, 5)

The output produced should be the following:

2 3 4 5 6 7

19 18 17 16 15 14 13 12 11

5

def printRange(num1, num2):

    if num1 < num2:

        for num in range(num1, num2 + 1):

            print(num, end=" ")

    elif num1 > num2:

        for num in range(num1, num2 - 1, -1):

            print(num, end=" ")

    else:

        print(num1)

# Sample calls to printRange

print("Increasing sequence:")

printRange(1, 10)

print("\nDecreasing sequence:")

printRange(11, 5)

print("\nSingle number:")

printRange(3, 3)

1. Write a method named smallestLargest that asks the user to enter numbers, then prints the smallest and largest of all the numbers typed in by the user. You may assume the user enters a valid number greater than 0 for the number of numbers to read. Here is an example dialogue:

How many numbers do you want to enter? 4

|  |  |  |
| --- | --- | --- |
| Number | 1: | 5 |
| Number | 2: | 11 |
| Number | 3: | -2 |
| Number | 4: | 3 |

Smallest = -2

Largest = 11

def smallestLargest():

    num\_of\_numbers = int(input("Enter the number of numbers to read: "))

    if num\_of\_numbers <= 0:

        print("Please enter a valid number greater than 0.")

        return

numbers = []

    for i in range(num\_of\_numbers):

        try:

            num = float(input(f"Enter number {i + 1}: "))

            numbers.append(num)

        except ValueError:

            print("Invalid input. Please enter a valid number.")

            Return

    if len(numbers) > 0:

        smallest = min(numbers)

        largest = max(numbers)

        print(f"The smallest number is: {smallest}")

        print(f"The largest number is: {largest}")

    else:

        print("No numbers entered.")

# Call the method to test

smallestLargest()

1. Write a method called printAverage that uses a sentinel loop to repeatedly prompt the user for numbers. Once the user types any number less than zero, the method should display the average of all nonnegative numbers typed. Display the average as a double. Here is a sample dialogue with the user:

Type a number: 7 Type a number: 4 Type a number: 16 Type a number: –4 Average was 9.0

If the first number that the user types is negative, do not print an average:

Type a number: –2

def printAverage():

    total = 0

    count = 0

    while True:

        try:

            num = float(input("Enter a number (negative number to calculate average): "))

            if num < 0:

                break  # Exit the loop if a negative number is entered

            total += num

            count += 1

        except ValueError:

            print("Invalid input. Please enter a valid number.")

    if count > 0:

        average = total / count

        print(f"The average of {count} nonnegative numbers is: {average:.2f}")

    else:

        print("No nonnegative numbers entered.")

# Call the method to test

printAverage()

1. Write a method named numUnique that takes three integers as parameters and returns the number of unique integers among the three. For example, the call numUnique(18, 3, 4) should return 3 because the parameters have three different values. By contrast, the call numUnique(6, 7, 6) should return 2 because there are only two unique numbers among the three parameters: 6 and 7.

def numUnique(num1, num2, num3):

    unique\_numbers = set([num1, num2, num3])

    return len(unique\_numbers)

# Test the method

result1 = numUnique(18, 3, 4)

print(result1)  # Output: 3

result2 = numUnique(6, 6, 6)

print(result2)  # Output: 2

1. A Random object generates pseudo-random numbers. Find out how to use the Random class and write a program that simulates rolling of two 6- sided dice until their combined result comes up as 7. One possible output can be seen as below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | + | 4 | = | 6 |
| 3 | + | 5 | = | 8 |
| 5 | + | 6 | = | 11 |
| 1 | + | 1 | = | 2 |
| 4 | + | 3 | = | 7 |

You won after 5 tries!

import random

def roll\_dice():

return random.randint(1, 6)

def simulate\_dice\_rolls():

    total = 0

rolls = 0

    while total != 7:

        dice1 = roll\_dice()

        dice2 = roll\_dice()

        total = dice1 + dice2

        print(f"Roll {rolls + 1}: {dice1} + {dice2} = {total}")

        rolls += 1

print(f"\nIt took {rolls} rolls to get a total of 7.")

# Simulate dice rolls

simulate\_dice\_rolls()