Lab 1: Expressions, Statements, and Control structures

Name: Ziya Shaheer University Number: [redacted]

Exercise 1: Surface Area and Volume of a Hexagonal Pyramid

AIM:

Write a Python program that prompts the user for side length l of the base and height h in cm of a hexagonal pyramid (i.e. a pyramid with a hexagonal base and six isosceles triangular faces that intersect at the apex), computes the surface area A and volume V of the pyramid using the formulas $A = 3l(\sqrt{3}l + \sqrt{3l^2 + 4h^2})/2$ and $V = \sqrt{3}hl^2/2$, and finally outputs the results. Here are the sample input and output of this program:

```
Enter the side length l of the base of the pyramid in cm: 3
Enter the height h of the pyramid in cm: 4
The surface area of the pyramid is 66.3099499659424 cm^2.
The volume of the pyramid is 31.17691453623979 cm^3.
```

ALGORITHM:

1. Begin program
2. Read in the length of the base & height of the pyramid from the user
3. Calculate the surface area (A) & volume (V) of the pyramid using provided formulas
4. Output the surface area and volume of the pyramid to the user

PROGRAM:

```
# Surface Area & Volume of a (Regular) Hexagonal Pyramid
# Prompts the user for the length of the base & height of the pyramid
# Shaheer Ziya (Last updated: Jan 26,2022)
import math


#Read in the length of the base (L) and the height of the pyramid (H)
from the user
L = float(input("Enter the side length of the base of the pyramid in cm:
"))
H = float(input("Enter the height of the pyramid in cm: "))
```

```
#Calculate the surface area (A) and volume (V) of the pyramid
A = (3*L) * ((math.sqrt(3) * L) + math.sqrt(3 * (L ** 2 ) + 4 * (H ** 2))) / 2
V = (math.sqrt(3) * H * (L**2)) / 2


#Output the surface area (A) and volume(V) to the user
print()
print(f"The surface area of the pyramid is {A} cm^2.")
print()
print(f"The volume of the pyramid is {V} cm^3.")
```

OUTPUT:

```
In [6]: runfile('C:/Users/shaheer/.spyder-py3/temp.py', wdir='C:/
Users/shaheer/.spyder-py3')

Enter the side length of the base of the pyramid in cm: 3

Enter the height of the pyramid in cm: 4

The surface area of the pyramid is 66.3099499659424 cm^2.

The volume of the pyramid is 31.17691453623979 cm^3.
```

Exercise 2: Operation of Two Integers

AIM:

Write a Python program that prompts for two integer operands and one of the binary operators +, –, *, /, or % from the user, performs the operation on the operands with the operator using an `if-elif-else` statement, and finally prints the result. Your program should check whether the input operator is valid. Here are the sample input and output of this program:

```
Enter the integer A: 5
Enter the integer B: 4
Enter a binary operator (+,-,*,/,%): +
5 + 4 = 9
Enter the integer A: 2
Enter the integer B: 3
Enter a binary operator (+,-,*,/,%): =
Invalid input. The operator must be one of the followings: +,-,*,/,%.
```

ALGORITHM:

1. Begin program
2. Ask user for integer A, integer B & the binary operator (string)
3. Run through conditionals, if the operator matches with the specified one then perform the calculation using that operator and output the result to the user

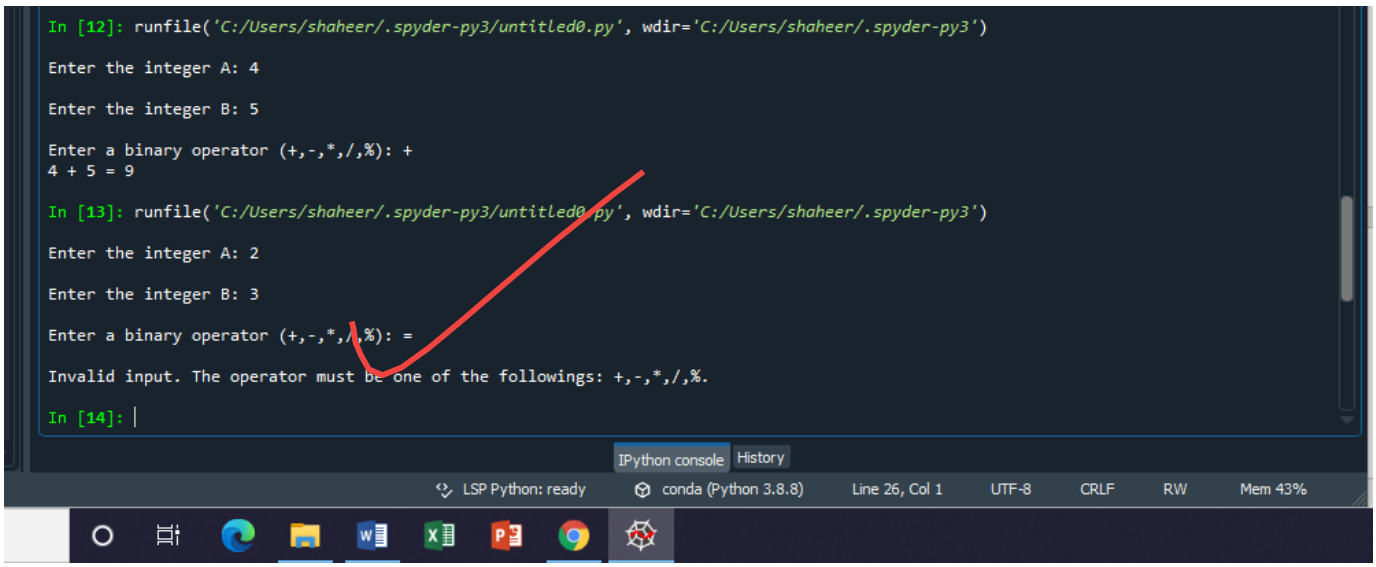
PROGRAM:

```
#Calculator.py
#Prompts the user for two integers & a binary operand then outputs the result.
#Shaheer Ziya (Last updated: Jan 26,2022)

#Read Inputs
A = int(input("Enter the integer A: "))
B = int(input("Enter the integer B: "))
operand = input("Enter a binary operator (+,-,*,/,%): ")
```

```
#Performs the calculation and print the result
if operand == "+":
    print(f"{A} {operand} {B} = {A+B}")
elif operand == "-":
    print(f"{A} {operand} {B} = {A-B}")
elif operand == "*":
    print(f"{A} {operand} {B} = {A*B}")
elif operand == "/":
    print(f"{A} {operand} {B} = {A/B}")
elif operand == "%":
    print(f"{A} {operand} {B} = {A%B}")
else:
    print()
    print("Invalid input. The operator must be one of the followings:
+,-,*,/,%.")
```

OUTPUT:


In [12]: runfile('C:/Users/shaheer/.spyder-py3/untitled0.py', wdir='C:/Users/shaheer/.spyder-py3')
Enter the integer A: 4
Enter the integer B: 5
Enter a binary operator (+,-,*,/,%): +
4 + 5 = 9
In [13]: runfile('C:/Users/shaheer/.spyder-py3/untitled0.py', wdir='C:/Users/shaheer/.spyder-py3')
Enter the integer A: 2
Enter the integer B: 3
Enter a binary operator (+,-,*,/,%): =
Invalid input. The operator must be one of the followings: +,-,*,/,%.
In [14]:
IPython console
History
LSP Python: ready
conda (Python 3.8.8)
Line 26, Col 1
UTF-8
CRLF
RW
Mem 43%

20

Exercise 3: Taylor Series of $x(1+x^2)^{1/2}$

AIM:

Write a Python program that prompts for a real number x in the open interval (–1, 1) and a positive integer n from the user, computes the sum of the first n terms of the Taylor series

$$x\sqrt{1+x^2} = \sum_{i=0}^{n-1} \frac{(-1)^{i-1}(2i)!}{4^i(i!)^2(2i-1)} x^{2i+1}$$

for any $x \in (-1, 1)$ using a `for` loop, and finally output the result. Your program should check whether the user input is valid. Here are the sample input and output of this program:

```
Enter a real number x in (-1, 1): 0.6
Enter a positive integer n: 5
The sum of first 5 terms of the Taylor series of x*(1+x^2)^(1/2)
for x = 0.6 is 0.6996359400000001
Enter a real number x in (-1, 1): 2.2
Enter a positive integer n: 7
Invalid input. x must have absolute value less than 1!
Enter a real number x in (-1, 1): -0.9
Enter a positive integer n: -5
Invalid input. n must be a positive integer!
```

ALGORITHM:

1. Begin program
2. Prompt user for float that is in (-1,1) and a positive integer
3. If the inputs satisfy provided conditions, then
4. Initialize the _sum variable (to 0) that records the approximate value of the function
5. Enter a for loop (running n-times)
6. Calculate the i-th term of the Taylor series for the function
7. Add the i-th term for the Taylor series of the function to the _sum variable
8. End loop after n iterations
9. Output result to the user

10. If the inputs are invalid (i.e. do not satisfy pre-ordained conditions) then inform the user that their inputs are invalid.

PROGRAM:

```
#Taylor_Series.py
#Approximate the value of the function x(1+x^2)^1/2 using its Taylor series for the first n terms
#Shaheer Ziya (Last updated: Jan 26,2022)

import math

#Read inputs
x = float(input("Enter a real number x in (-1,1): "))
n = int(input("Enter a positive integer n: "))

#Loop and Sum of first n terms of Taylor Series for the function
if (-1 < x < 1) and (n > 0):
    _sum = 0
    for i in range(n):
        numerator = (-1)**(i-1) * math.factorial(2*i) * x**((2*i)+1)
        denominator = (4**i) * (math.factorial(i)**2) * ((2*i)-1)
        ith_term = numerator / denominator
        _sum += ith_term

    #Output the result to the user
    print()
    print(f"The sum of the first {n} terms of the Taylor series of x*(1+x^2)^(1/2) for x = {x} is {_sum}")

#Handle the case when x is not in (-1,1)
elif (not(-1 < x < 1)):
    print("Invalid input. x must have am absolute value less than 1!")
#Handle the case when x is not positive
```

```
elif (n <= 0):
    print()
    print("Invalid input. n must be a positive integer!")
#Doomsday---The End of Time as we know it
else:
    print()
    print("Something has gone horribly wrong! Try again.")

#Comments: Error handling can be improved using Try/Except clauses
```

OUTPUT:

```
In [28]: runfile('C:/Users/shaheer/.spyder-py3/untitled1.py', wdir='C:/Users/shaheer/.spyder-py3')

Enter a real number x in (-1,1): 0.6

Enter a positive integer n: 5

The sum of the first 5 terms of the Taylor series of x*(1+x^2)^(1/2) for x = 0.6 is 0.6996359400000001

In [29]: runfile('C:/Users/shaheer/.spyder-py3/untitled1.py', wdir='C:/Users/shaheer/.spyder-py3')

Enter a real number x in (-1,1): 2.2

Enter a positive integer n: 7
Invalid input. x must have am absolute value less than 1!

In [30]: runfile('C:/Users/shaheer/.spyder-py3/untitled1.py', wdir='C:/Users/shaheer/.spyder-py3')

Enter a real number x in (-1,1): -0.9

Enter a positive integer n: -5

Invalid input. n must be a positive integer!
```

20

Exercise 4: Greatest Common Divisor

AIM:

The Greatest Common Divisor (GCD) of two integers is the largest positive integer that divides both of them without leaving a remainder. An efficient way for finding the GCD of two natural numbers (i.e. non-negative integers) is the Euclidean algorithm which works as follows:

(a) If one of the numbers is zero, then the GCD is the other number and we can stop.

(b) Compute the remainder of the larger number divided by the smaller one and then replace the larger number by the remainder

(c) Repeat step (b) until the remainder is zero. The GCD is the smaller number in this case.

Write a Python program that prompts the user for two natural numbers, find their GCD with the Euclidean algorithm using a `while` loop, an `if-else` statement, and `if-elif-else` statements, and finally outputs the result. Your program should check whether the user input is valid. Here are the sample input and output of this program:

```
Enter a natural number x: 168
Enter another natural number y: 180
The GCD of 168 and 180 is 12
Enter a natural number x: 25
Enter another natural number y: -10
Invalid input. Both x and y must be integers >= 0!
Enter a natural number x: 18
Enter another natural number y: 18
Invalid input. x and y must be different numbers!
```

ALGORITHM:

1. Begin
2. Read two natural numbers from the user
3. Continue asking for natural numbers until valid inputs are provided using try/except clauses
4. Determine which of x or y is larger and assign the larger of the two to large and the smaller one to small respectively
5. Initialize the remainder variable to zero
6. Begin while loop with condition s.t. it ends when remainder is zero

7. Find the remainder of the larger number divided by the smaller number and assign it to the variable remainder
8. Assign the previous small variable value to large and the remainder just calculated to small so that the variable large is always larger than small.
9. When remainder equals zero end loop and initialize the variable GCD to the value of large (since small will now equal zero)
10. Output the result to the user

PROGRAM:

```
#GCD.py
#Find the Greatest Common Divisor of two natural numbers using the
Euclidean algorithm for GCD
#Shaheer Ziya (Last updated: Jan 26,2022)

#Read two natural numbers from the user
while True:
    #Continue looping and asking until valid inputs are provided
    try:
         x = int(input("Enter a natural number x: "))
         y = int(input("Enter another natural number y: "))
         #If invalid inputs then raise Error and skip to Except block
         if (x <= 0) or (y <= 0):
             raise ValueError
         #If valid inputs then break out of loop
         break
    except:
        if x == y:
            print("Invalid Input. x and y must be different numbers!")
        else:
             print("Invalid input. Both x and y must be integers larger
than 0!")

#Determine which number is larger
if x > y:
    large, small = x, y
```

```
else:
    small, large = x, y

#Begin looping
remainder = -1
while remainder != 0:
    remainder = large % small
    large, small = small, remainder
GCD = large

print(f"The GCD of {x} and {y} is {large}")
```

OUTPUT:

```
In [53]: runfile('C:/Users/shaheer/.spyder-py3/untitled2.py', wdir='C:/Users/shaheer/.spyder-py3')

Enter a natural number x: 168

Enter another natural number y: 180
The GCD of 168 and 180 is 12

In [54]: runfile('C:/Users/shaheer/.spyder-py3/untitled2.py', wdir='C:/Users/shaheer/.spyder-py3')

Enter a natural number x: 25

Enter another natural number y: -10
Invalid input. Both x and y must be integers larger than 0!
```

Exercise 5: Displaying a Number Pattern

AIM:

Write a Python program that prompts the user for a line number n which is a positive integer < 10 and then displays a number pattern of $2n-1$ lines with the following format:

1
121
12321
1234321
123454321
1234321
12321
121
1

using nested `for` loops. Your program should check whether the input value of n is valid.

ALGORITHM:

PROGRAM:

```
#pattern.py
#Prints out a desired pattern
#Shaheer Ziya (Last updated: Jan 26,2022)

n = 5
x = "1"
pattern = []
print(f"{x :^{2*n-1}}")
for i in range(1,n):
```

```
    x = str(x) + str(int(i) + 1)
    line = x + x[-2::-1]
    print(f"{line:^{2*n-1}}")
    pattern.append(line)

for j in range(n):
    print(n-j-1)
    print(f"{str(pattern[n-j-1]): ^{2*n-1}}")
```

OUTPUT:

```
In [110]: runfile('C:/Users/shaheer/.spyder-py3/untitled3.py', wdir='C:/Users/shaheer/.spyder-py3')
    1
   121
  12321
 1234321
123454321
5
Traceback (most recent call last):

  File "C:\Users\shaheer\.spyder-py3\untitled3.py", line 18, in <module>
    print(f"{str(pattern[n-j]): ^{2*n-1}}")

IndexError: list index out of range


In [111]: runfile('C:/Users/shaheer/.spyder-py3/untitled3.py', wdir='C:/Users/shaheer/.spyder-py3')
    1
   121
  12321
 1234321
123454321
4
Traceback (most recent call last):

  File "C:\Users\shaheer\.spyder-py3\untitled3.py", line 18, in <module>
    print(f"{str(pattern[n-j-1]): ^{2*n-1}}")

IndexError: list index out of range


In [112]:
```