Python Problems

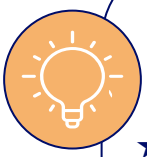
Introduction Coding Booklet

Session 3 OISE/TDSB
Coding Workshop

Problem 1

Write a program that accepts the perimeter and length of a rectangle and outputs the corresponding width.

Note: If length inputted is not possible for the perimeter of the rectangle given, an error message should be printed to let the user know their information needs to be checked.



Input length and perimeter

- ★ **input** function to request user for data
- ★ **int** function to convert to an integer

```
# Calculate the Width of a Rectangle
# width = (perimeter - length * 2) / 2
length = int(input("length: "))
perimeter = int(input("perimeter: "))
if length * 2 < perimeter:
  width = (perimeter - length * 2) / 2
  print(width)
else:
  print("Error: Please check input for length! ")
```

```
length: 4
perimeter: 9
0.5
```

Conditional Statement

- ★ **if** function for when length is possible for perimeter calculation
- ★ **else** for the error and width cannot be calculated

Problem 2

A rectangular piece of cardboard can be turned into a rectangular prism box by cutting equal square corners and then folding the sides up.

Write a program that accepts the length of the cardboard, the width of the cardboard, and the side length of the squares. The program should use the information provided to calculate the volume of the box created.

Input Height, Length, Width

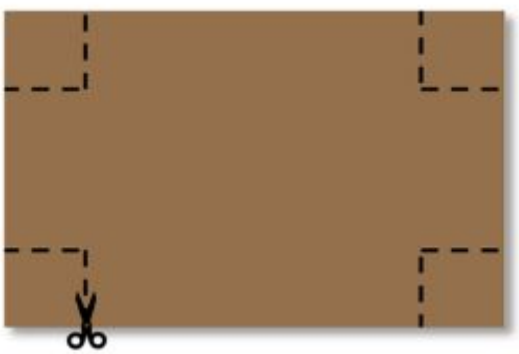
- ★ **input** function to request user for data
- ★ **int** function to convert to an integer

```
# Calculate Volume
# volume = height * width * length

height = int(input("What is the side length of one square corner? "))
length = int(input("What is the length of the cardboard? "))
width = int(input("What is the width of the cardboard? "))

volume = height * (length - height * 2) * (width - height * 2)
print("The volume of the box is", volume)
```

```
What is the side length of one square corner? 1
What is the length of the cardboard? 8
What is the width of the cardboard? 4
The volume of the box is 12
```

Printing Solutions

- ★ Use **comma** to combine strings and numerical values when presenting the solution



Problem 3

Write a program that asks for a value of x and calculates the corresponding value of y on the line $y = -\frac{2}{3}x - 11$

Comment to Label Your Work

★ Use the **hashtag #** function to keep notes and label for yourself and other users

```
# Solving for Y
# y = -2/3 * x - 11

x = float(input("What is the value for x? "))

y = -2/3 * x - 11

print("The correponding value of y is", y)
```

```
What is the value for x? -35.6
The correponding value of y is 12.733333333333334
```

Not an integer?

★ Use the **float** function to convert to a non-integer number (decimal)

Problem 4

Consider the sum: 1 + 1/3 + 1/6 + 1/9 + 1/12 ....
Write a program that will calculate the sum up to the inputted number of terms.

For Loops

- ★ **for** loops are used to calculate and print values
- ★ **range** function to determine the start and end of the loop

```
# Calculating Sum
n=int(input("Enter the number of terms: "))
sum = 0
for x in range(0,n):
  if x == 0:
    denom = 1
  else:
    denom = x * 3
  sum = sum + (1/denom)
print("The sum of series is", sum)
```

```
Enter the number of terms: 2
The sum of series is 1.3333333333333333
```

Conditional Statements

- ★ **if** function for when the denominator is 1 (as the first term is 1)
- ★ **else** for the rest of the sequence denominators following 1/3x

Problem 5

Consider the sum: 1 + 1/2 + 1/4 + 1/8 + 1/16 + ...
Write a program that will calculate the sum for as many terms as the user would like. What do you notice?

For Loops

- ★ **for** loops are used to calculate and print values
- ★ **range** function to determine the start and end of the loop

```
# Calculating Sum
n=int(input("Enter the number of terms: "))
sum1 = 0
for x in range(0,n):
  if x == 0:
    denom1 = 1
  else:
    denom1 = x * 2
  sum1 = sum1 + 1/denom1
print("The sum of series is", sum1)
```

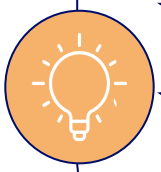
```
Enter the number of terms: 26
The sum of series is 2.907979088876753
```

Conditional Statements

- ★ **if** function for when the denominator is 1 (as the first term is 1)
- ★ **else** for the rest of the sequence denominators following 1/2x

Problem 6a

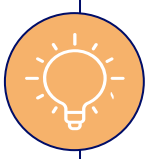
Consider the the perfect squares: 1, 4, 9, 16, 25, ...
Write a program that will calculate the perfect squares of the base numbers between 1 and 10.



While Loops

- ★ **while** loops are used to calculate and print values
- ★ **counter** is used to repeat and end the loop

```
#Define lists to store base and perfect square values
baseValues = []
squareValues = []

#Calculates the base and perfect square values
base = 1
while base < 11:
  baseValues.append(base)
  squareValues.append(base*base)
  base = base + 1

#Output the base number and their perfect squares
counter = 0
while counter <10:
  print("The perfect square of ", baseValues[counter], "is ", squareValues[counter])
  counter = counter + 1
```



Lists

- ★ **_List = []** defines there is a list
- ★ **_List.append()** puts items in the list
- ★ Lists start at 0, therefore **loop** function can reorder list to start from first item

```
The perfect square of  1 is  1
The perfect square of  2 is  4
The perfect square of  3 is  9
The perfect square of  4 is  16
The perfect square of  5 is  25
The perfect square of  6 is  36
The perfect square of  7 is  49
The perfect square of  8 is  64
The perfect square of  9 is  81
The perfect square of  10 is  100
```

Problem 6b

The volume of a cylinder is $V = \pi r^2 h$. Write a program that accepts a height as input from the user and prints out volumes with radiuses from 1 cm to 10 cm.

Is the pattern shown a linear or non-linear pattern? Does this answer change for difference heights?

```python
# Change of Radius Volume of Cylinder
# volume = pi * radius^2 * height

import math

#Define Lists
h = int(input("height: "))
rList = []
volumeList = []

for r in range (1, 10):
  volume = round(math.pi * r**2 * h, 2)
  rList.append(r)
  volumeList.append(volume)

print("Volumes and Radiuses with Height: ", h)

#Printing Lists Side by Side
for i in range(1, 10):
  index = i - 1
  radius = rList[index]
  volume = volumeList[index]
  print("Radius: ", radius, ", Volume: ", volume)
```

Rounding

- **round** function rounds down the value
- The n value in round(, n) represents the decimal places to round to

Printing Lists

- Lists start at 0, therefore **loop** function can reorder list to start from first item

```
height: 5
Volumes and Radiuses with Height:  5
Radius:  1 , Volume:  15.71
Radius:  2 , Volume:  62.83
Radius:  3 , Volume:  141.37
Radius:  4 , Volume:  251.33
Radius:  5 , Volume:  392.7
Radius:  6 , Volume:  565.49
Radius:  7 , Volume:  769.69
Radius:  8 , Volume:  1005.31
Radius:  9 , Volume:  1272.35
```