Example 1:

```
length = 10 #variable1
width = 5 #variable2
area = length * width #This is called an expression
print(area)
```

```
50
```

Example 2:

```
print(7+8.5) # 7 is int type & 8.5 is float type, but mixing these two programs run well
```

```
15.5
```

Example 3:

```
base = 6
height = 2
area = base*height/2
print("Area is " + str(area)) #changed, int type, 'area' to string
```

```
Area is 6.0
```

1. Write a program to prompt the user for hours and rate per hour using input( ) to compute gross pay. Pay the hourly rate for the hours up to 40 and 1.5 times the hourly rate for all hours worked above 40 hours. Use 45 hours and a rate of 10.50 per hour to test the program (the pay should be 498.75).

```
hrs = input("Enter Hours:")
h = float(hrs)
rate = input("Enter the rate per hour:")
r = float(rate)
if h <= 40:
  grossPay = h * r
else:
  regularGrossPay = 40 * r
  overtimeGrossPay = (h-40)*r*1.50
  grossPay = regularGrossPay + overtimeGrossPay
print(grossPay)
```

```
Enter Hours:45
Enter the rate per hour:10.50
498.75
```

Example 4:

```
def greeting (name): #passing parameters/argumets to the function
 print("welcome "+ name)
greeting("Diparshan")
```

```
welcome Diparshan
```

Example 5:

```
## getting values out of the function(return values)
def area_traingle(b, h):
 return b*h/2
area = area_traingle(4, 8)
print("area of traingle is: " + str(area))
```

```
area of traingle is: 16.0
```

Example 6:

2. Write a function that compares two numbers and returns them in increasing order.

```
def order_numbers(number1, number2):
  if number2 > number1:
   return number1, number2
  else:
    return number2, number1
smaller, bigger = order_numbers(100, 99)
print(smaller,",", bigger)
```

```
99 , 100
```

Example 7:

```
#Even or Odd
def is_Even(number):
 if (number%2 == 0):
  return True
 else:
  return False
is_Even(7)
```

```
False
```

Example 8:

```
def number_group(number):
 if number > 0:
  return "Positive"
 elif number == 0:
  return "Zero"
 else:
  return "Negative"
print(number_group(10))
print(number_group(0))
print(number_group(-5))
```

```
Positive
Zero
Negative
```

Example 9:

```
#while loop
x = 0 # initialzation of variable Why=?
while x<5:
 print("Not there yet, x= " + str(x))
 x = x+1
print("x= "+str(x))
```

```
Not there yet, x= 0
Not there yet, x= 1
Not there yet, x= 2
Not there yet, x= 3
Not there yet, x= 4
x= 5
```

Example 10:

```
def count_down(start_number):
 current=3 #initialzation of variable
 while (current > 0):
  print(current)
  current -= 1
 print("Zero!")
count_down(3)
```

```
3
2
1
Zero!
```

Example 11:

```
#Infinite loops and how to break them?
while x%2 == 0:
 x = x/2
#To solve this problem we can add break to the program
while x%2 == 0:
  if x == 0:
    break
  x = x/2
```

Example 12:

```
#Solution of Example 11
if x != 0:
 while x%2 == 0:
  x = x/2
#or
while x != 0 and x%2 == 0:
 x = x/2
```

Example 13:

```
# for loop, iterates over a sequence of values
for x in range(5):
 print(x)
```

```
0
1
2
3
4
```

Example 14:

```
def square(n):
 return n*n
def sum_squares(x):
 sum = 0
 for n in range(10):
  sum += square(n)
 return sum
print(sum_squares(10))

#The program calculates the sum and average of the numbers
#stored in the list called values. It uses a for loop to
#iterate over each integer in the list and adds them together
#to calculate the sum. Additionally, it keeps track of the number
#of integers encountered during the iteration to determine the length
#of the list. Finally, it displays both the sum and the average by dividing
#the sum by the length of the list.
```

```
285
```

Example 15:

```
values = [10, 20, 30, 40, 50]
Sum = 0
length = 0
for value in values:
 Sum = Sum+value
 length = length+1
print("Total sum= "+str(Sum)+"and the Average="+str(Sum/length))

#This program calculates the sum and average of the values in the "Values" list.
#It iterates over each value, accumulates the sum, and keeps track of the count.
#Finally, it prints the total sum and average of the values.
```

```
Total sum= 150and the Average=30.0
```

Example 16:

```
#nested for loops
for left in range(7):
  for right in range(left, 7):
    print("[" + str(left) + "|" + str(right) + "]",end=" ")
  print()
# 1.This program generates and prints a pattern of pairs of numbers in the form "[left|right]"
# 2.The outer loop iterates over the numbers from 0 to 6 (7-1) and assigns each number to the variable "left".
# 3.The inner loop iterates over the numbers from "left" to 6 (7-1) and assigns each number to the variable "right".
# 4.Inside the inner loop, it prints the pair of numbers in the format "[left|right]".
# 5.After printing all pairs for the current value of "left", it goes to the next line using the print() function.
# 6.The program continues this pattern, incrementing the "left" value in the outer loop and printing the pairs in
#   the inner loop until it has printed all pairs for each value from 0 to 6.
```

```
[0|0] [0|1] [0|2] [0|3] [0|4] [0|5] [0|6] 
[1|1] [1|2] [1|3] [1|4] [1|5] [1|6] 
[2|2] [2|3] [2|4] [2|5] [2|6] 
[3|3] [3|4] [3|5] [3|6] 
[4|4] [4|5] [4|6] 
[5|5] [5|6] 
[6|6] 
```

Exercises:1

a. Write a function fib (n) that prompts the number of terms “n” from the user and displays first “n” terms of Fibonacci sequence iteratively.

```
def fib():
  initial = [0,1]
  if n<=2:
    return initial
  for i in range(2,n):
    next_term = initial[i-2] + initial[i-1]
    initial.append(next_term)
  return initial
n = int(input("Enter the number of terms for fibonacci sequence: "))
print("Fibonacci sequence upto",n,"terms: ",fib())
```

```
Enter the number of terms for fibonacci sequence: 8
Fibonacci sequence upto 8 terms:  [0, 1, 1, 2, 3, 5, 8, 13]
```

b. Write a function myPow(x, n) that returns x^n, where n is a non-negative integer. Do not use the ** operator or the math.pow function - use one while loop.

```
def myPow(x, n):
    result = 1

    while n > 0:
        if n % 2 == 1:
            result *= x
        x *= x
        n //= 2

    return result

# Test the function
x = float(input("Enter the base x: "))
n = int(input("Enter the exponent n: "))
print("The result would be: ",myPow(x,n))
```

```
Enter the base x: 2
Enter the exponent n: 5
The result would be:  32.0
```

c. Write a function SumDigits(n) that displays the sum of digits of “n”. Hint: SumDigits(234) will be 2+3+4=9

```
def SumDigits(n):
  sum = 0
  if len(str(n))==1:
    print("The sum of given digit:",n)
  else:
    for i in str(n):
      sum += int(i)
    return sum
n = int(input("Enter a number: "))
print("Sum of digits:",SumDigits(n))
```

```
Enter a number: 345
Sum of digits: 12
```

d. Write a function squaredSum that returns the sum of the squares of an array/list of integers such that squaredSum([2,3,4]) gives 29. Write your solution in pure Python to show the logic in your solution, do not use any library functions.

```
def squaredsum(arr):
  sum=0
  for num in arr:
    square = num**2
    sum += square
  return sum
numbers = [2,3,4]
print("Sum of squares: ",squaredsum(numbers))
```

```
Sum of squares:  29
```

e. Write a function called sumDigits that is given an integer num and returns the sum of the digits of num. Eg. sumDigits(6) gives 21. Do not use any library functions (if there is one).

```
def sumDigits(num):
    digit_sum = 0
    num = abs(num)
    while num > 0:
        digit = num % 10
        digit_sum += digit
        num //= 10
    return digit_sum
number = int(input("Enter a number: "))
print("Sum of digits:",sumDigits(number))
```

```
Enter a number: 678
Sum of digits: 21
```