Heading-1

Heading-2

Heading-3

Heading-4

Heading-5

Heading-6

Heading-1

Heading-2

Heading-3

Heading-4

Heading-5

Heading-6

This is a text in BOLD...

This is a text in BOLD...

Inputs and Outputs

In [4]:
```
num1 = input("Please enter the first number: ")
num2 = input("Please enter the second number: ")
print (num1, type(num1), num2, type(num2))
result = num1 + num2
print ("So the result is", result)
print ("End of the program...")
```

```
100 <class 'str'> 300 <class 'str'>
So the result is 100300
End of the program...
```

In [5]:
```
num1 = int(input("Please enter the first number: "))
num2 = int(input("Please enter the second number: "))
print (num1, type(num1), num2, type(num2))
result = num1 + num2
print ("So the result is", result)
print ("End of the program...")
```

```
100 <class 'int'> 300 <class 'int'>
So the result is 400
End of the program...
```

In [8]:
```
num1 = int(input("Please enter the first number: "))
num2 = int(input("Please enter the second number: "))
result = num1 + num2
print (num1, type(num1), num2, type(num2))
```

```
100 <class 'int'> 400 <class 'int'>
```

In [26]:
```
print ("So the sum of", num1, "+", num2, "=", result)
print ("So the sum of " + str(num1) + " + " + str(num2) + " = " + str(result))
print ("So the sum of {} + {} = {}".format(num1, num2, result))   # {} denotes place holder
print ("So the sum of {0} + {1} = {2}".format(num1, num2, result))  # numbered place holder
print ("So the sum of {2} + {0} = {1}".format(num2, result, num1))
print ("So the sum of {fnum} + {snum} = {tot}".format(fnum = num1, snum = num2, tot = result))   # labeled placehold
print ("So the sum of {fnum} + {snum} = {tot}".format(snum = num2, fnum = num1, tot = result))
print ("So the sum of %d + %d = %d"%(num1, num2, result))
print ("So the sum of %d + %f = %d"%(num1, num2, result))
print ("So the sum of %d + %8.2f = %d"%(num1, num2, result))
print (f"So the sum of {num1} + {num2} = {result}")    # smart formatting
```

```
So the sum of 100 + 400 = 500
```

```
So the sum of 100 + 400 = 500
So the sum of 100 + 400 = 500
So the sum of 100 + 400 = 500
So the sum of 100 + 400 = 500
So the sum of 100 + 400 = 500
So the sum of 100 + 400 = 500
So the sum of 100 + 400 = 500
So the sum of 100 + 400.000000 = 500
So the sum of 100 +   400.00 = 500
So the sum of 100 + 400 = 500
```

In [147…

```
result = eval(input("Please enter your expression: "))
print (result)
```

```
700
```

Operators

In [28]:

```
# Arithmetic operators: + - * / // ** %
print (100 + 40)    # addition
print (100 - 40)    # subtraction
print (100 * 40)    # multiplication
print (100 / 40)    # float division
print (100 // 40)   # integer division
print (100 ** 4)    # exponentiation, a to the power of b
print (100 % 40)    # modulus, remainder of the division
```

```
140
60
4000
2.5
2
100000000
20
```

In [31]:

```
# logical operators: and or not
print (True and True, True and False, False and True, False and False)   # True, False, None keywords starts with ca
print (True or True, True or False, False or True, False or False)
print (not True, not False)
```

```
True False False False
True True True False
False True
```

In [32]:

```
# Relational operators: > >= < <= != ==                -gt -ge -lt -le -ne -eq,    .gt. .ge. .lt. .le. .ne. .eq.
print (100 > 50, 100 >= 100, 100 < 400, 100 <= 500, 100 != 200, 400 == 400)
```

```
True True True True True True
```

In [49]:

```
# Ternary operator:
# Operator classifications: Unary (one operand), Binary (two operands) and Ternary (three operands)
# Unary: +10 -20, Binary: 10 + 20, 30 * 4, Ternary: as shown below (True part, Condition and False part)
num1 = 100
result = "EVEN Number" if (num1 % 2 == 0) else "ODD Number"    # result = num1 % 2? "True": "False"   (in C, C++, Ja
print (result)

num1 = 101
result = "EVEN Number" if (num1 % 2 == 0) else "ODD Number"
print (result)
```

```
EVEN Number
ODD Number
```

In [48]:

```
# Assignment and special assignment operators: = += -= /= //= *= **= %=
num1 = 100
print (num1)
num1 += 10
print (num1)
num1 -= 10
print (num1)
num1 *= 10
print (num1)
num1 **= 2
print (num1)
```

```
100
110
100
1000
1000000
```

In [46]:
```
# Bitwise operators: & | ^ ~
# A => 65 => 64 + 1      => 0100 0001
#                        or 0010 0000 => 32
#                           ---------
# a => 97 => 64 + 32 + 1 => 0110 0001
mychar = 'A'
print (mychar, ord(mychar))
mychar = chr(ord(mychar) | 32)
print (mychar, ord(mychar))
```

```
A 65
a 97
```

In [47]:
```
# Bitwise operators: & | ^ ~
# a => 97 => 64 + 32 + 1 => 0110 0001
#                       and 1101 1111 => 255 - 32 = 223
#                           ---------
# A => 65 => 64 + 1      => 0100 0001
mychar = 'a'
print (mychar, ord(mychar))
mychar = chr(ord(mychar) & 223)
print (mychar, ord(mychar))
```

```
a 97
A 65
```

ASCII Codes

In [ ]:
```
ASCII = > American Standard Code for Information Interchange (8 bits code or representation)
So ASCII codes can have the value ranging from 0 to 255 (2^8 - 1)
ASCII codes can be divided into two categories:
    1) Normal ASCII Codes (Printable): 0 to 127
    2) Extended ASCII Codes (Non-Printable): 128 to 255 (These ASCII codes can be used for control characters)
a => 97, b => 98, ..., z => 122
A => 65, B => 66, ..., Z => 90
0 => 48, 1 => 49, ..., 9 => 57
Tab => 8, Back Space => 9, Enter => 13, Esc => 27, Space Bar => 32 and so on.
```

In [42]:
```
# two functions to deal with ASCII codes: chr(), ord()
print (chr(65), chr(66), chr(90), chr(97), chr(98), chr(122))
print (ord("A"), ord("B"), ord("Z"), ord("a"), ord("b"), ord("z"))
```

```
A B Z a b z
65 66 90 97 98 122
```

Conditional Statements

In [62]:
```
# Problem Statement: Take three numbers from the keyboard as input and find the maximum of them and print the maximu
num1 = int(input("Please enter the first number: "))
num2 = int(input("Please enter the second number: "))
num3 = int(input("Please enter the third number: "))
if (num1 > num2):
    if (num1 > num3):
        print ("The first number is the maximum number...")
        print ("The maximum number is", num1)
    else:
        print ("The third number is the maximum number...")
        print ("The maximum number is", num3)
elif (num2 > num3):
    print ("The second number is the maximum number...")
    print ("The maximum number is", num2)
else:
    print ("The third number is the maximum number...")
    print ("The maximum number is", num3)
print ("End of the program...")
```

```
The second number is the maximum number...
The maximum number is 300
End of the program...
```

In [65]:
```
# Problem Statement: Take three numbers from the keyboard as input and find the maximum of them and print the maximu
num1 = int(input("Please enter the first number: "))
num2 = int(input("Please enter the second number: "))
num3 = int(input("Please enter the third number: "))
if (num1 >= num2 and num1 >= num3):
    print ("The first number is the maximum number...")
    print ("The maximum number is", num1)
elif (num2 > num3):
    print ("The second number is the maximum number...")
```

```
    print ("The maximum number is", num2)
else:
    print ("The third number is the maximum number...")
    print ("The maximum number is", num3)
print ("End of the program...")
```

```
The first number is the maximum number...
The maximum number is 300
End of the program...
```

In [125…

```
num1 = 100;
if (num1 % 2 == 0):
    print ("This is an EVEN number...");
    print ("EVEN numbers are divisible by 2...");
else:
    print ("This is an ODD number...");
    print ("ODD numbers are not divisible by 2...");

print()
num1 = 101
if (num1 % 2 == 0):
    print ("This is an EVEN number...")
    print ("EVEN numbers are divisible by 2...")
else:
    print ("This is an ODD number...")
    print ("ODD numbers are not divisible by 2...")
```

```
This is an EVEN number...
EVEN numbers are divisible by 2...

This is an ODD number...
ODD numbers are not divisible by 2...
```

In [131…

```
num1 = 100
if (num1 % 2 == 0): print ("This is an EVEN number..."); print ("EVEN numbers are divisible by 2...")
else: print ("This is an ODD number..."); print ("ODD numbers are not divisible by 2...")

print()
num1 = 101
if (num1 % 2 == 0): print ("This is an EVEN number..."); print ("EVEN numbers are divisible by 2...")
else: print ("This is an ODD number..."); print ("ODD numbers are not divisible by 2...")
```

```
This is an EVEN number...
EVEN numbers are divisible by 2...

This is an ODD number...
ODD numbers are not divisible by 2...
```

In [149…

```
result = ((5, 7)[5 > 7])
print (result)

result = ((5, 7)[55 > 7])
print (result)
```

```
5
7
```

Misc. Concepts

In [ ]:

```
Variables initialized with any one of the values ranging from -5 to 256 (inclusive) will generate same id for same v
```

In [72]:

```
num1 = 256
print (num1, type(num1), id(num1))
num2 = 256
print (num2, type(num2), id(num2))
num3 = num1
print (num3, type(num3), id(num3))
```

```
256 <class 'int'> 140729659377424
256 <class 'int'> 140729659377424
256 <class 'int'> 140729659377424
```

In [74]:

```
num1 = 257
print (num1, type(num1), id(num1))
num2 = 257
print (num2, type(num2), id(num2))
num3 = num1
print (num3, type(num3), id(num3))
num4 = 257
print (num4, type(num4), id(num4))
```

```
257 <class 'int'> 2051892168272
257 <class 'int'> 2051892170512
257 <class 'int'> 2051892168272
257 <class 'int'> 2051899705296
```

In [75]:

```
num1 = -5
print (num1, type(num1), id(num1))
num2 = -5
print (num2, type(num2), id(num2))
num3 = num1
print (num3, type(num3), id(num3))
```

```
-5 <class 'int'> 140729659369072
-5 <class 'int'> 140729659369072
-5 <class 'int'> 140729659369072
```

In [78]:

```
num1 = -6
print (num1, type(num1), id(num1))
num2 = -6
print (num2, type(num2), id(num2))
num3 = num1
print (num3, type(num3), id(num3))
num4 = -6
print (num4, type(num4), id(num4))
```

```
-6 <class 'int'> 2051899704720
-6 <class 'int'> 2051899705296
-6 <class 'int'> 2051899704720
-6 <class 'int'> 2051899704560
```

In [83]:

```
print ("Hello", "to", "all", "of", "you")
print ("Welcome")
```

```
Hello to all of you
Welcome
```

In [80]:

```
help(print)  # ellipsis operator is (...)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

In [86]:

```
print ("Hello", "to", "all", "of", "you", sep = ", ")
print ("Welcome")
```

```
Hello, to, all, of, you
Welcome
```

In [88]:

```
print ("Hello", "to", "all", "of", "you", sep = ", ", end = " - ")
print ("Welcome")
```

```
Hello, to, all, of, you - Welcome
```

In [97]:

```
print ("Hello " * 5)
print ('Welcome ' * 7)
```

```
Hello Hello Hello Hello Hello
Welcome Welcome Welcome Welcome Welcome Welcome Welcome
```

In [95]:

```
# python supports variant datatype for variables, depending upon the value got assigned to the variable will decide
var1 = 100
print (var1, type(var1), id(var1))
var1 = "Celebration"
print (var1, type(var1), id(var1))
var1 = 1234.56
print (var1, type(var1), id(var1))
var1 = True
print (var1, type(var1), id(var1))
```

```
100 <class 'int'> 140729659372432
Celebration <class 'str'> 2051886709872
```

```
1234.56 <class 'float'> 2051899706416
True <class 'bool'> 140729659086672
```

In [103…

```
# inbuilt collection classes and implicit objects
var1 = [1001, "Amit", "Male", "Developer", 30000.55, True]
print (var1, len(var1), type(var1), id(var1))
var1 = (1001, "Amit", "Male", "Developer", 30000.55, True)
print (var1, len(var1), type(var1), id(var1))
var1 = {"empid":1001, "empname":"Amit", "empgender":"Male", "empjob":"Developer", "empsal":30000.55, "empmarried":Tr
print (var1, len(var1), type(var1), id(var1))
var1 = {1001, "Amit", "Male", "Developer", 1001, "Amit", "Male", "Developer", 30000.55, True}
print (var1, len(var1), type(var1), id(var1))
var1 = frozenset([1001, "Amit", "Male", "Developer", 1001, "Amit", "Male", "Developer", 30000.55, True])
print (var1, len(var1), type(var1), id(var1))
```

```
[1001, 'Amit', 'Male', 'Developer', 30000.55, True] 6 <class 'list'> 2051885903808
(1001, 'Amit', 'Male', 'Developer', 30000.55, True) 6 <class 'tuple'> 2051889094176
{'empid': 1001, 'empname': 'Amit', 'empgender': 'Male', 'empjob': 'Developer', 'empsal': 30000.55, 'empmarried': Tru
e} 6 <class 'dict'> 2051886834368
{True, 1001, 30000.55, 'Developer', 'Male', 'Amit'} 6 <class 'set'> 2051895477184
frozenset({True, 1001, 30000.55, 'Developer', 'Male', 'Amit'}) 6 <class 'frozenset'> 2051895476736
```

Loop Constructs

In [113…

```
for i in range(10):
    print (i, end = ", ")
else:
    print ("\nLoop got executed successfully...")

for i in range(0, 10):
    print (i, end = ", ")
else:
    print ("\nLoop got executed successfully...")

for i in range(0, 10, 1):
    print (i, end = ", ")
else:
    print ("\nLoop got executed successfully...")

for i in range(-10, 0, 1):
    print (i, end = ", ")
else:
    print ("\nLoop got executed successfully...")

for i in range(1, 10, 2):
    print (i, end = ", ")
else:
    print ("\nLoop got executed successfully...")
```

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
Loop got executed successfully...
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
Loop got executed successfully...
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
Loop got executed successfully...
-10, -9, -8, -7, -6, -5, -4, -3, -2, -1,
Loop got executed successfully...
1, 3, 5, 7, 9,
Loop got executed successfully...
```

In [118…

```
# break continue and pass statements
# using the break statement
for i in range(10):
    print (i, end = ", ")
    if (i == 5): break
else:
    print ("\nLoop got executed successfully...")
print ("\nEnd of the program...")
```

```
0, 1, 2, 3, 4, 5,
End of the program...
```

In [120…

```
# break continue and pass statements
# using the continue statement
for i in range(10):
    if (i == 5 or i == 7 or i == 8):
        print ("Continuing the loop...")
        continue
    print ("i =", i)
else:
    print ("Loop got executed successfully...")
print ("End of the program...")
```

```
i = 0
i = 1
i = 2
i = 3
i = 4
Continuing the loop...
i = 6
Continuing the loop...
Continuing the loop...
i = 9
Loop got executed successfully...
End of the program...
```

In [134…

```python
num1 = 100
if (num1 % 2 == 0):
    pass      # pass does nothing, pass is a statement placeholder
else:
    print ("This is an ODD number...")
    print ("ODD numbers are not divisible by 2...")

num1 = 101
if (num1 % 2 == 0):
    pass      # pass does nothing, pass is a statement placeholder
else:
    print ("This is an ODD number...")
    print ("ODD numbers are not divisible by 2...")
```

```
This is an ODD number...
ODD numbers are not divisible by 2...
```

In [135…

```python
# using the break statement in while block
i = 1
while (i <= 10):
    print (i, end = ", ")
    if (i == 5): break
    i = i + 1
else:
    print ("\nLoop got executed successfully...")
print ("\nEnd of the program...")
```

```
1, 2, 3, 4, 5,
End of the program...
```

In [136…

```python
# using the continue statement in while block
i = 1
while (i <= 10):
    if (i == 5 or i == 7 or i == 8):
        print ("Continuing the loop...")
        i += 1
        continue
    print ("i =", i)
    i += 1
else:
    print ("Loop got executed successfully...")
print ("End of the program...")
```

```
i = 1
i = 2
i = 3
i = 4
Continuing the loop...
i = 6
Continuing the loop...
Continuing the loop...
i = 9
i = 10
Loop got executed successfully...
End of the program...
```

In [139…

```python
# example of nested loop
print ("Visit Doctor...")    # 1 time
day = 1
medicount = 0
while (day <= 5):
    print ("Good morning to my Family Members...")     # 5 times
    for medi in range(1, 4):
        print (f"Day No. = {day} and Medicine No. = {medi}...")     # 15 times
        medicount += 1
    print ("Good night to my Family Members...")     # 5 times
    print ("-----------------------------------------")     # 5 times
    day += 1
print ("Thanks to Doctor...")
print (f"Total number of medicine consumed is {medicount}...")
print ("End of the story...")
```

```
Visit Doctor...
Good morning to my Family Members...
Day No. = 1 and Medicine No. = 1...
Day No. = 1 and Medicine No. = 2...
Day No. = 1 and Medicine No. = 3...
Good night to my Family Members...
-----------------------------------------
Good morning to my Family Members...
Day No. = 2 and Medicine No. = 1...
Day No. = 2 and Medicine No. = 2...
Day No. = 2 and Medicine No. = 3...
Good night to my Family Members...
-----------------------------------------
Good morning to my Family Members...
Day No. = 3 and Medicine No. = 1...
Day No. = 3 and Medicine No. = 2...
Day No. = 3 and Medicine No. = 3...
Good night to my Family Members...
-----------------------------------------
Good morning to my Family Members...
Day No. = 4 and Medicine No. = 1...
Day No. = 4 and Medicine No. = 2...
Day No. = 4 and Medicine No. = 3...
Good night to my Family Members...
-----------------------------------------
Good morning to my Family Members...
Day No. = 5 and Medicine No. = 1...
Day No. = 5 and Medicine No. = 2...
Day No. = 5 and Medicine No. = 3...
Good night to my Family Members...
-----------------------------------------
Thanks to Doctor...
Total number of medicine consumed is 15...
End of the story...
```

Class Assignment - 1

Find the sum of the following series for the first n number of terms, where this n will be given by the user.

total = 1 + 2 + 3 + 4 + 10 + 5 + 6 + 7 + 8 + 26 + 9 + 10 + ... n terms

> Also find the 99th term of the series.
>
> For n = 5, then total = 20
>
> For n = 7, then total = 31
>
> For n = 10, then total = 72

Algorithm for the Series Problem:

input n fsum = tsum = 0 term = 1 for i = 1 to n if (i % 5 == 0) then fsum = fsum + tsum tsum = 0 else fsum = fsum + term tsum = tsum + term term = term + 1 end if end for print "So the final sum of the series is", fsum

In [142...

```python
# Python code
n = int(input("Please enter the number of terms of the series: "))
fsum = tsum = 0
term = 1
for i in range(1, n + 1):
    if (i % 5 == 0):
        fsum = fsum + tsum
        print ("Adding the temporary sum term", tsum)
        tsum = 0
    else:
        fsum = fsum + term
        print ("Adding the term", term)
        tsum = tsum + term
        term = term + 1
print ("So the final sum of the series is", fsum)
print ("End of the program...")
```

```
Adding the term 1
Adding the term 2
Adding the term 3
Adding the term 4
Adding the temporary sum term 10
Adding the term 5
Adding the term 6
Adding the term 7
Adding the term 8
Adding the temporary sum term 26
Adding the term 9
Adding the term 10
```

```
So the final sum of the series is 91
End of the program...
```

In [ ]:

```
Pattern Printing - 1
--------------------
n = 6
.....*
....***
...*****
..*******
.*********
***********
```

i	.	*	
1	5	1	(i, n)
2	4	3	. => (n - i)
3	3	5	
4	2	7	* => (2 * i - 1)
5	1	9	
6	0	11	

Tracing Table

In [144…

```
n = int(input("Please enter the number of layers: "))
for i in range(1, n + 1):
    print ("." * (n - i) + "*" * (2 * i - 1))
print ("End of the program...")
```

```
......*
.....***
....*****
...*******
..*********
.***********
*************
End of the program...
```

In [ ]:

```
Pattern Printing - 2
--------------------
n = 6
***********
.*********
..*******
...*****
....***
.....*
```

i	.	*	
1	0	11	(i, n)
2	1	9	. => (i - 1)
3	2	7	
4	3	5	* => (2 * (n - i) + 1)
5	4	3	
6	5	1	

Tracing Table

In [146…

```
n = int(input("Please enter the number of layers: "))
for i in range(1, n + 1):
    print ("." * (i - 1) + "*" * (2 * (n - i) + 1))
print ("End of the program...")
```

```
*****************
.***************
..*************
...***********
....*********
.....*******
......*****
.......***
........*
End of the program...
```

Class Assignment - 2

Form the following pattern using a single loop construct.

In [ ]:

```
Pattern Printing - 3
--------------------
n = 11 (It should be always an ODD number)   m = (n + 1) // 2 = 6
.....*
....***
...*****
..*******
.*********
***********
.*********
..*******
...*****
....***
.....*
```

i	.	*	
1	5	1	(i, n, m)
2	4	3	. => (m - i)
3	3	5	
4	2	7	* => (2 * i - 1)
5	1	9	
6	0	11	
7	1	9	
8	2	7	. => (i - m)
9	3	5	
10	4	3	* => (2 * (n - i) + 1)
11	5	1	

```
---------------
Tracing Table
```

In [197…

```
while (True):
    n = int(input("Please enter the ODD number of layers: "))
    if (n % 2 == 1): break
m = (n + 1) // 2
for i in range(1, n + 1):
    if (i > m): dot = (i - m); star = (2 * (n - i) + 1)
    else: dot = (m - i); star = (2 * i - 1)
    print ("." * dot + "*" * star)
print ("End of the program...")
```

```
.....*
....***
...*****
..*******
.*********
***********
.*********
..*******
...*****
....***
.....*
End of the program...
```

In [ ]:

Python Functions

In [153…

```
# function takes no input arguments and returns no output arguments
def funct1():
    print ("Welcome...", end = " ")
    print ("To all...")

funct1()
funct1()
funct1()
funct1()
print (type(funct1), id(funct1))
```

```
Welcome... To all...
Welcome... To all...
Welcome... To all...
Welcome... To all...
<class 'function'> 2051889753248
```

In [156…

```
# function takes input arguments and returns no output arguments
def funct2(msg, times):
    print (msg * times)

funct2("Hello ", 5)
funct2("Welcome ", 10)
funct2("Good Bye !!! ", 7)
print (type(funct2), id(funct2))
```

```
Hello Hello Hello Hello Hello
Welcome Welcome Welcome Welcome Welcome Welcome Welcome Welcome Welcome Welcome
Good Bye !!! Good Bye !!! Good Bye !!! Good Bye !!! Good Bye !!! Good Bye !!! Good Bye !!!
<class 'function'> 2051889752816
```

In [184…

```
# function takes input arguments and also returns output arguments
def funct3(msg, times):
    return msg * times

result = funct3("Hello ", 5)
print (result)

result = funct3(msg = "Welcome ", times = 10)
print (result)

result = funct3(times = 7, msg = "Good Bye !!! ")
print (result)

print (type(funct3), id(funct3), type(result), id(result))
```

```
Hello Hello Hello Hello Hello
Welcome Welcome Welcome Welcome Welcome Welcome Welcome Welcome Welcome Welcome
Good Bye !!! Good Bye !!! Good Bye !!! Good Bye !!! Good Bye !!! Good Bye !!! Good Bye !!!
<class 'function'> 2051887717824 <class 'str'> 2051889332560
```

In [168…

```
# function takes input arguments and returns multiple values in collection class object as output arguments
def funct4(num1, num2):
    total = num1 + num2
    difference = num1 - num2
    product = num1 * num2
    quotient = num1 / num2
    remainder = num1 % num2
    return total, difference, product, quotient, remainder

tt, dd, pp, qq, rr = funct4(100, 40)
print (f"Total = {tt}, Difference = {dd}, Product = {pp}, Quotient = {qq}, Remainder = {rr}")
print ()
result = funct4(100, 40)
print (f"Total = {result[0]}, Difference = {result[1]}, Product = {result[2]}, Quotient = {result[3]}, Remainder = {
print ()
print (result, type(result), len(result), id(result))
```

```
Total = 140, Difference = 60, Product = 4000, Quotient = 2.5, Remainder = 20

Total = 140, Difference = 60, Product = 4000, Quotient = 2.5, Remainder = 20

(140, 60, 4000, 2.5, 20) <class 'tuple'> 5 2051887530288
```

In [188…

```
# function with default arguments
def funct5(par1 = 111, par2 = 222, par3 = 333):    # positional parameters
    print (f"par1 = {par1}, par2 = {par2} and par3 = {par3}...")

funct5(100, 200, 300)    # positional arguments
funct5(100, 200)
funct5(100)
funct5()
funct5(par1 = 100, par3 = 300)
funct5(par3 = 300, par1 = 100)
```

```
par1 = 100, par2 = 200 and par3 = 300...
par1 = 100, par2 = 200 and par3 = 333...
par1 = 100, par2 = 222 and par3 = 333...
par1 = 111, par2 = 222 and par3 = 333...
par1 = 100, par2 = 222 and par3 = 300...
par1 = 100, par2 = 222 and par3 = 300...
```

In [186…

```
def funct6(par1 = 100, par2 = None):
    if (par2 == None):
        return par1 + par1
    else:
        return par1 + par2
print (funct6(100, 400))
print (funct6(100))
print (funct6(par2 = 400, par1 = 100))
print (funct6(par1 = 100))
```

```
500
200
500
200
```

In [190…

```
# function with variable number of input arguments
def funct7(*arg):    # *arg defines forefully that arg is a tuple object
    print (arg, len(arg), type(arg), id(arg))

funct7("Joydeep", "Tester")
funct7("Joydeep", "Tester", "Pune", 55000)
funct7("Joydeep", "Tester", "Pune", 55000, "Male", True)
```

```
('Joydeep', 'Tester') 2 <class 'tuple'> 2051892000128
('Joydeep', 'Tester', 'Pune', 55000) 4 <class 'tuple'> 2051888648000
('Joydeep', 'Tester', 'Pune', 55000, 'Male', True) 6 <class 'tuple'> 2051899638976
```

In [191…

```
# function with variable number of input arguments
def funct8(**kwarg):    # **kwarg defines forefully that kwarg is a dictionary object, kwarg means keyword argument
    print (kwarg, len(kwarg), type(kwarg), id(kwarg))

funct8(empname = "Joydeep", empjob = "Tester")
funct8(empname = "Joydeep", empjob = "Tester", emploc = "Pune", empsal = 55000)
funct8(empname = "Joydeep", empjob = "Tester", emploc = "Pune", empsal = 55000, empgender = "Male", empmarried = Tru
```

```
{'empname': 'Joydeep', 'empjob': 'Tester'} 2 <class 'dict'> 2051885950912
{'empname': 'Joydeep', 'empjob': 'Tester', 'emploc': 'Pune', 'empsal': 55000} 4 <class 'dict'> 2051887626560
{'empname': 'Joydeep', 'empjob': 'Tester', 'emploc': 'Pune', 'empsal': 55000, 'empgender': 'Male', 'empmarried': Tru
e} 6 <class 'dict'> 2051885889088
```

In [193…

```
# function with variable number of input arguments
```

```
def funct9(*arg, **kwarg):
    print (arg, len(arg), type(arg), id(arg))
    print (kwarg, len(kwarg), type(kwarg), id(kwarg))

funct9("Joydeep", "Tester", "Pune", empsal = 55000, empgender = "Male", empmarried = True)
print ()
funct9(empsal = 55000, empgender = "Male", empmarried = True)
print ()
funct9("Joydeep", "Tester", "Pune")
```

```
('Joydeep', 'Tester', 'Pune') 3 <class 'tuple'> 2051887956352
{'empsal': 55000, 'empgender': 'Male', 'empmarried': True} 3 <class 'dict'> 2051887136256

() 0 <class 'tuple'> 2051793551424
{'empsal': 55000, 'empgender': 'Male', 'empmarried': True} 3 <class 'dict'> 2051887135616

('Joydeep', 'Tester', 'Pune') 3 <class 'tuple'> 2051887956352
{} 0 <class 'dict'> 2051887133504
```

In [198…

```
def funct10():
    i = 100
    print ("Printing from within the function:", i, id(i))

i = 10
print ("Printing before calling the function:", i, id(i))
funct10()
print ("Printing after calling the function:", i, id(i))
```

```
Printing before calling the function: 10 140729659369552
Printing from within the function: 100 140729659372432
Printing after calling the function: 10 140729659369552
```

In [199…

```
i = 10
def funct10():
    i = 100
    print ("Printing from within the function:", i, id(i))

print ("Printing before calling the function:", i, id(i))
funct10()
print ("Printing after calling the function:", i, id(i))
```

```
Printing before calling the function: 10 140729659369552
Printing from within the function: 100 140729659372432
Printing after calling the function: 10 140729659369552
```

In [201…

```
def funct10():
    global i
    print ("Printing from within the function:", i, id(i))
    i = 100
    print ("Printing from within the function:", i, id(i))

i = 10
print ("Printing before calling the function:", i, id(i))
funct10()
print ("Printing after calling the function:", i, id(i))
```

```
Printing before calling the function: 10 140729659369552
Printing from within the function: 10 140729659369552
Printing from within the function: 100 140729659372432
Printing after calling the function: 100 140729659372432
```

In [208…

```
# non-recursive factorial function
def factorial_nr(num):
    if (num == 0 or num == 1): return 1
    fact = num
    for i in range(2, num):
        fact = fact * i
        print (f"So the current value in i = {i} and fact = {fact}...")
    return fact

n = 5
result = factorial_nr(n)
print (f"Non-Recursive: Factorial of {n} is {result}...")
print()
n = 7
result = factorial_nr(n)
print (f"Non-Recursive: Factorial of {n} is {result}...")
```

```
So the current value in i = 2 and fact = 10...
So the current value in i = 3 and fact = 30...
So the current value in i = 4 and fact = 120...
Non-Recursive: Factorial of 5 is 120...
```

```
So the current value in i = 2 and fact = 14...
So the current value in i = 3 and fact = 42...
So the current value in i = 4 and fact = 168...
So the current value in i = 5 and fact = 840...
So the current value in i = 6 and fact = 5040...
Non-Recursive: Factorial of 7 is 5040...
```

In [209…

```python
# recursive factorial function
def factorial_r(num):
    if (num == 0 or num == 1): return 1   # this is called base case where the recursion will terminate.
                                          # base case means for certain inputs outputs are pre-known to us
    return num * factorial_r(num - 1)     # recursive case, where the function will call itself

n = 5
result = factorial_r(n)
print (f"Recursive: Factorial of {n} is {result}...")
print()
n = 7
result = factorial_r(n)
print (f"Recursive: Factorial of {n} is {result}...")
```

```
Recursive: Factorial of 5 is 120...

Recursive: Factorial of 7 is 5040...
```

In [ ]:

```
5! = 5 * 4!
         4 * 3!
             3 * 2!
                 2 * 1!
                     1  (Base case)
                 2
             6
         24
      120
```

Python Lambda or Anonymous function

In [210…

```python
mysquare = lambda num: num * num

print (type(mysquare), id(mysquare))
print (mysquare(10))
print (mysquare(9))
```

```
<class 'function'> 2051889335600
100
81
```

In [211…

```python
myaddition = lambda num1, num2: num1 + num2

print (type(myaddition), id(myaddition))
print (myaddition(100, 900))
print (myaddition(400, 600))
```

```
<class 'function'> 2051887714512
1000
1000
```

In [213…

```python
def funct11(num):
    myprod = lambda n: n * num
    return myprod

var10 = funct11(10)
var20 = funct11(20)
print (var10(3))
print (var20(4))
print (type(var10), id(var10), type(var20), id(var20))
```

```
30
80
<class 'function'> 2051889335168 <class 'function'> 2051889336176
```

In [217…

```python
# Lambda function with recursion
myfact = lambda num: 1 if (num == 0 or num == 1) else num * myfact(num - 1)
print (myfact(5))
print (myfact(7))
```

```
120
5040
```

In [ ]:

In [ ]: