1) How to receive the input string from user?

Use **input** () function, this function **Displays a prompt for the input** and **receives a string of keystrokes, called characters, entered at the keyboard and returns the string to the shell.**

```
1   name = input("Enter your Name: ")
2   print(name)
```

How to receive the input integer or any data type else?

The programmer must convert the input from strings to the appropriate numeric types.

Function	What it Does
`float(<a string of digits>)`	Converts a string of digits to a floating-point value
`int(<a string of digits>)`	Converts a string of digits to an integer value
`input(<a string prompt>)`	Displays the string prompt and waits for keyboard input; returns the string of characters entered by the user
`print(<expression>, ...,<expression>)`	Evaluates the expressions and displays them, separated by one space, in the console window
`<string 1> + <string 2>`	Glues the two strings together and returns the result

Example:1

```
1   degree = int(input("Enter The subject Degree: "))
2   print(degree)
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS
PS C:\Users\elsaw\AppData\Local\Programs\Microsoft VS Code> & C:\Users\elsaw\AppData\Lo
Enter The subject Degree: 88
88
```

Example:2

Prompts the user for the width and height of a rectangle, computes its area, and outputs the result:

```
1   width = float(input("Enter width = " ))
2   hight = float(input("Enter hight = " ))
3   area = width * hight
4   print("The area = ",area)
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS
PS C:\Users\elsaw\AppData\Local\Programs\Microsoft VS Code> & C:\Users\e
Enter width = 5
Enter hight = 10
The area =  50.0
```

Note:

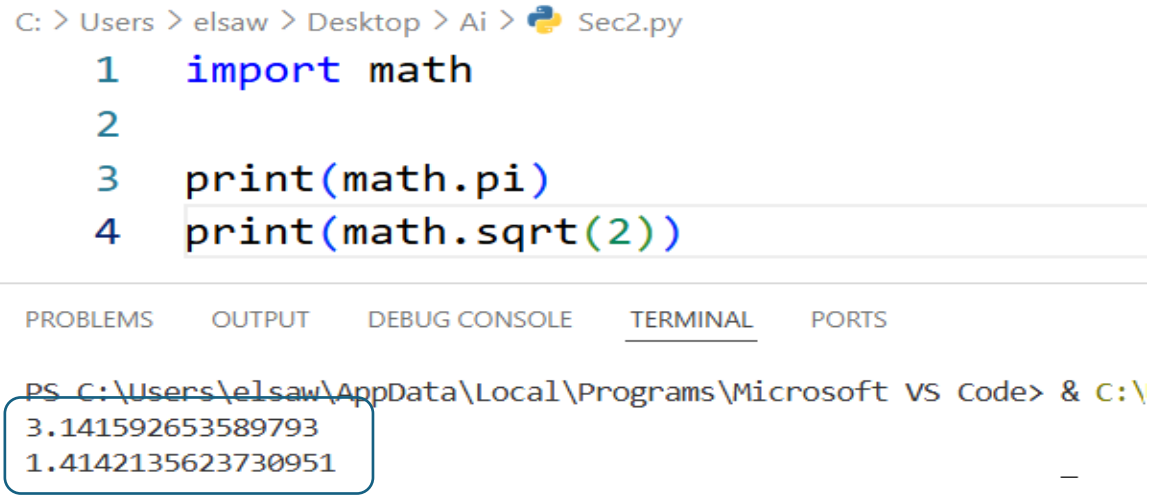
- To convert characters to their numeric ASCII codes we use **ord()** function.
- To convert numeric ASCII codes to their characters we use **chr()** function.

Use basic mathematical operations in python:

Functions and other resources are coded in components called modules.

Functions like abs and round from the __builtin__ module are always available for use, whereas the programmer must explicitly import other functions from the modules where they are defined. The **math module** includes several functions that perform basic mathematical operations.

Ex:



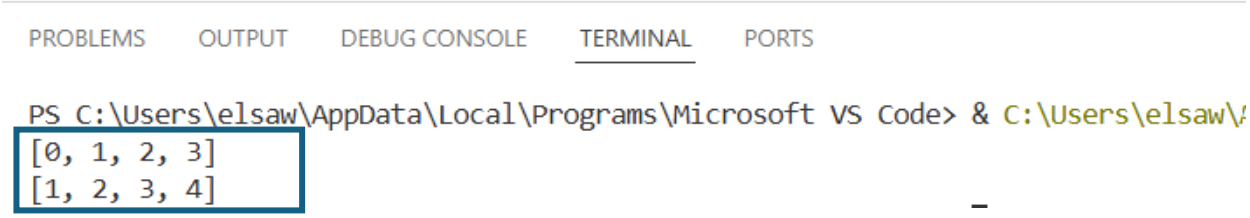
2) Augmented Assignment:

```
a = 17

s = "hi"

a += 3          # Equivalent to a = a + 3

a -= 3          # Equivalent to a = a - 3

a *= 3          # Equivalent to a = a * 3

a /= 3          # Equivalent to a = a / 3

a %= 3          # Equivalent to a = a % 3

s += " there"          #Equivalent to s = s + " there"
```

3) Traversing the Contents of a Data Sequence

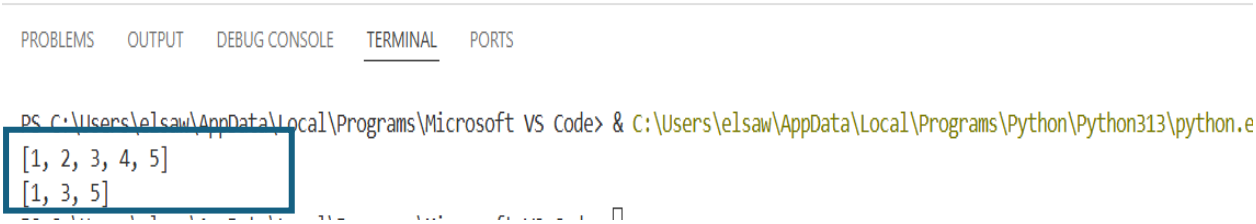
We have been using the for loop as a simple count-controlled loop, the loop itself visits each number in a sequence of numbers generated by the **range function**. The next code segment shows what these sequences look like:

```
1  import math
2  first_list = list(range(4))
3  print(first_list)
4
5  second_list = list(range(1, 5))
6  print(second_list)
7
```



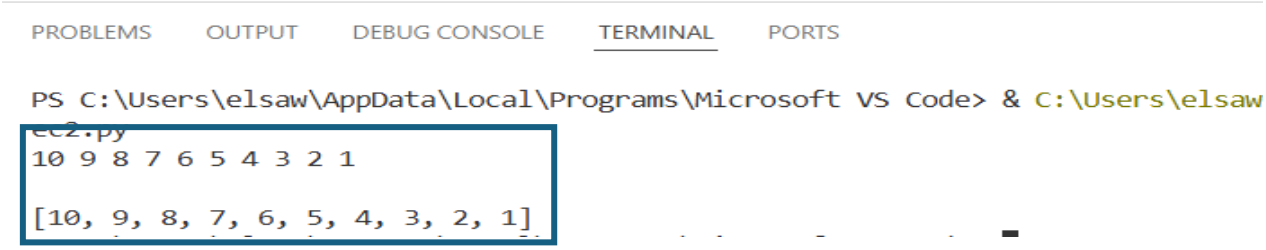
4) Specifying the Steps in the Range

```
1  list1 = list(range(1, 6, 1)) #start num , End num , incremental step value
2  print(list1)
3
4  list2 = list(range(1, 6, 2))
5  print(list2)
```



5) Loops That Count Down

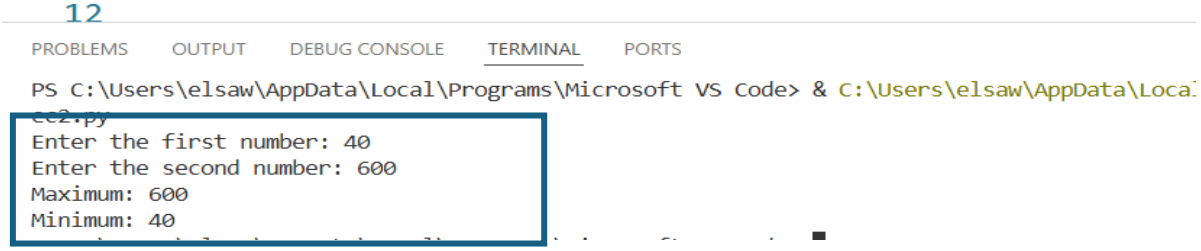
```python
# first way
for i in range(10 , 0 , -1):
    print(i ,end= " ")

print ("\n")

#second way
my_list = list(range(10 , 0 , -1))
print(my_list)
```



6) Example: prints the maximum and minimum of two input numbers using two ways:

1) Using if.... elseif... else condition

```python
#first way to print the max and min numbers
first = int(input("Enter the first number: "))
second = int(input("Enter the second number: "))
if first > second:
    maximum = first
    minimum = second
else:
    maximum = second
    minimum = first
print("Maximum:", maximum)
print("Minimum:", minimum)
```



2) Using max and min built-in function

```python
#second way to print the max and min numbers
first = int(input("Enter the first number: "))
second = int(input("Enter the second number: "))
print("Maximum:", max(first, second))
print("Minimum:", min(first, second))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\elsaw\AppData\Local\Programs\Microsoft VS Code> & C:\Users\elsaw\AppData\L
oc2.py
Enter the first number: 700
Enter the second number: 220
Maximum: 700
Minimum: 220
```

7) Array in Python:

- All elements in an array must be of the same data type. This type is specified when creating the array.
- arrays from the array module are more memory-efficient than Python lists for large collections of numerical data.
- Elements are accessed using integer indices, starting from 0 for the first element.

Ex:

```python
#create an array in python
arr_fruits = ["Apple" , "banana", "kiwi"]
#access the element of an array
print(arr_fruits[0]) #Apple
print(arr_fruits[1]) #banana
print(arr_fruits[2]) #kiwi
#find the length of an array
print(len(arr_fruits))
#Adding array element:
arr_fruits.append("grapes")
print(arr_fruits)
#deleting array element using index:
arr_fruits.pop(1)
print(arr_fruits)
#deleting array element using value:
arr_fruits.remove("kiwi")
print(arr_fruits)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\elsaw\AppData\Local\Programs\Microsoft VS Code> & C:\Users\els
y
Apple
banana
kiwi
3
['Apple', 'banana', 'kiwi', 'grapes']
['Apple', 'kiwi', 'grapes']
['Apple', 'grapes']
```

8) using the array module

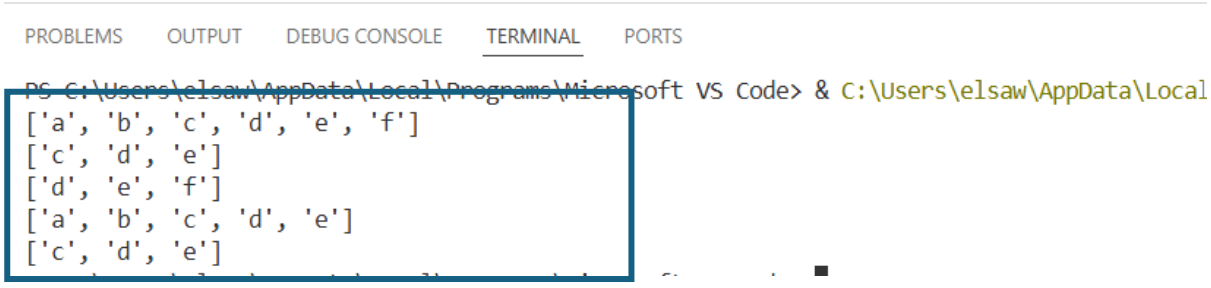
```python
import array as arr
# Create an array of signed integers ('i' type code)
my_array = arr.array('i', [10, 20, 30, 40, 50])

# Accessing elements
print(f"First element: {my_array[0]}") #f refers to formating string => appling {}
print(f"Third element: {my_array[2]}")

# Modifying an element
my_array[1] = 25
print(f"Modified array: {my_array}")

# Adding elements
my_array.append(60)
print(f"Array after appending: {my_array}")

my_array.insert(2, 35) # Insert 35 at index 2
print(f"Array after inserting: {my_array}")

# Removing elements
my_array.pop() # Removes the last element
print(f"Array after popping: {my_array}")

my_array.remove(30) # Removes the first occurrence of 30
print(f"Array after removing 30: {my_array}")

# Iterating through the array
print("Elements in the array:")
for x in my_array:
    print(x)
```

Output:

```
[Running] python -u "c:\Users\elsaw\Desktop\Ai\sec2_arr.py"
First element: 10
Third element: 30
Modified array: array('i', [10, 25, 30, 40, 50])
Array after appending: array('i', [10, 25, 30, 40, 50, 60])
Array after inserting: array('i', [10, 25, 35, 30, 40, 50, 60])
Array after popping: array('i', [10, 25, 35, 30, 40, 50])
Array after removing 30: array('i', [10, 25, 35, 40, 50])
Elements in the array:
10
25
35
40
50
```

9) slicing of an array:

```
1   x = ['a', 'b' ,'c','d','e', 'f']
2   #print all elements
3   print(x[:])
4   #slice elements from index 2 to index 5
5   print(x[2:5])
6   #slice from index 3 to the end of array
7   print(x[3:])
8   #slice from the begining to the index 5
9   print(x[:5])
10  #slice from index 4 from the end to the endex 1 from the end
11  print(x[-4:-1])
```



10) Stack and Queue in Python:

1) Stack:

- Stack works on principle of **"Last in, first out"**
- To add the new item in the stack we use **append ()** inbuilt function
- To remove the element from the stack use **pop ()** inbuilt function

2) Queue:

- Queue works on principle of **" first in, first out "**
- We use **pop (0)** to remove the first element
- To add the new item in the queue we use **append ()** inbuilt function

```python
#Stack using list
stack = ["A","B","C","D"]
#Add item in list
stack.append("F")
print(stack)

#remove item from list
print(stack.pop())
print(stack.pop())

print(stack)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SPI

PS C:\Users\elsaw\AppData\Local\Programs\Microsoft VS C
Ai/stack_queue.py
['A', 'B', 'C', 'D', 'F']
F
D
['A', 'B', 'C']
```

- **Queue**

```python
#Queue using list
Queue = ["A","B","C"]
#Add item in list
Queue.append("F")
print(Queue)

#remove item from list
print(Queue.pop(0))
print(Queue.pop(0))

print(Queue)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\elsaw\AppData\Local\Programs\Microsoft VS
Ai/stack_queue.py
['A', 'B', 'C', 'F']
A
B
['C', 'F']
```