Part 2: Python

Week:4

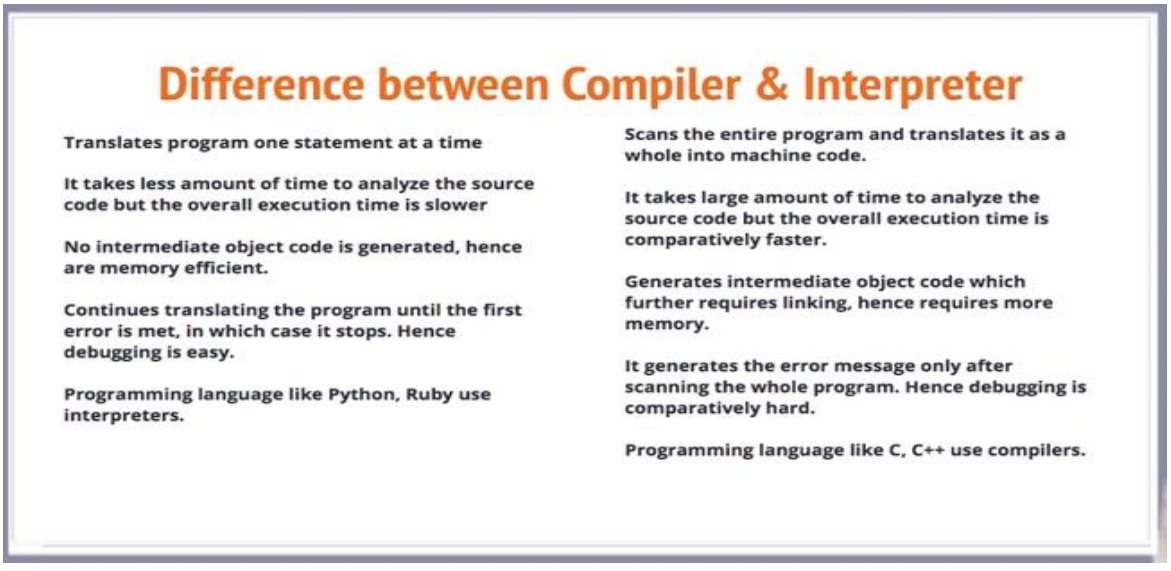
1. **Programming with python:**
   First, we have to differentiate between hardware and software. Basic difference is that all the tangible or touchable parts of computer/mobile is hardware of system and software is the program that runs and operate this hardware. Software is nit touchable like hardware. Software is very important fir the functioning if hardware without software, computer in just a box.
   Software like **operating system** are necessary for use hardware. **Operating System** controls and runs all hardware.

   Software/Program is a bunch of instructions that runs the hardware and system. It executes different things.

   Programming language is something in which we develop software. Python is one of language used to write/develop these programs.

2. **Compiler and Interpreter:**
   There is a strong system in backend that checks the error in our written program. Compiler and interpreter check the errors in the program and convert the source code (written by us) into object file (if it is error free) firstly and then convert it to computer language or machine code.

   Basic difference between compiler and interpreter is that **interpreter** actually checks the code line by line and convert it and if the line has the error, it will not go forward until error is removed. Languages like python, Ruby uses interpreter. **Compiler** checks the whole code first and then convert it to the machine code. All the errors are shown together. C, C++ use compilers.

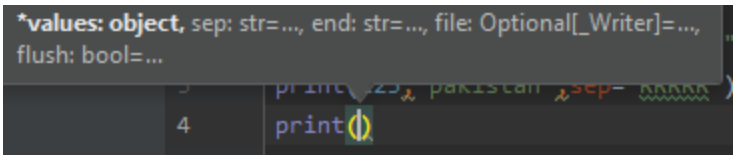


3. **Let's Learn Python:**
   We will follow text book **Smart way to learn python** and our reference book **Python Crash Course.** Firstly, we will study about print function.
   Print is a function used to output or display to you console window whatever you print out. It is one of the widely used functions.

4. **Version of Python:**
   For Artificial Intelligence best languages Python. It supports a lot of things that makes our task easy. It is working best for AI, ML, Deep learning and Data Science. It has a lot of libraries that makes the work the easy. It has simple syntax similar to English. It is lot easier to write program in Python instead of any other language. Python can be treated **Procedural way, an object-oriented, or a functional way.**
   Using python, we can deploy are mode. We can write backend of our website. One of the most famous ML libraries produced by **Google**, **Tensor Flow** is python based.

5. **Version of Python:**
   There are two versions of Python **version 2** and **version 3 and** latest is **version 3.7.** Its version keeps on increasing with improvement.

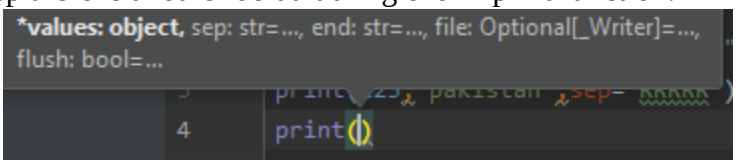
6. **Print:**
   The statement for print is **print().** This must be written like this for proper functionality. It is the case sensitive language.
   Print function will display the value you want it to show**. For example**
   We write a statement **print(100)**. It will display **100** in the output dialogue.
   Second this that can it print more than 1 value at a time so answer is **yes**. If we write **print(100,200)**.
   The output will be **100 200.** If you do not give any value to print function it will run but will not print any value.
   We can say these vales as **Data.** Data can be of different types. 100 and 200 are **Integer** type data. Another type of number data is **Float numbers data.** This the data type of those numbers that have point or decimal number like 12.1, 1.1, 6.5 etc. It will be print as it is too.
   Another data type is **String Data type.** That is used to output any text. Like if we want to print **Pakistan,** we have to write the print statement like that **print("Pakistan").** Any thing that will be written in single quotations or double quotations will become string we can also print numbers using quotations. Like if we want to print 123 in this case, we will write **print("123").**
   We can also print 2 different type of data together in a string. Like **print("123Pakistan").**
   Comma "," will separate the value. Interpreter will know that it's another value.
   If we will write **print("Pakistan zindabad", "123").** It will be print like **Pakistan zindabad 123**. It will have gap between to values because python use sep= " " which separate function and by default it has gap of 1 character.



If we don't want space in between two values we can write **print("Pakistan zindabad", "123", sep= "").** Here we created our own sep and don't give it space. It will print **Pakistan zindabad123.**

And if we write any number, character or sign in sep="" it will be printed instead of space. Like **print("Pakistan zindabad", "123", sep= "###")** it will give result **Pakistan zindabad###123.**

With sep there is another default thing end in print function.



It shows output on another line if we write it in separate print statement because it use **\n** (next line) function in default. So, if we don't want to go on next line we will not write **\n.**

```
print("Pakistan", end="")
print("Zindabad")
```

It will print Pakistan Zindabad.

You can use other escape sequence like **\t.** It will give tab space between two words. And also:

```
print("Pakistan", end="\n\n")
print("Zindabad")
```

It will Pakistan and Zindabad with one-line space.

7. Variables:

Variable is something that holds the value that may change. It is just a place holder or we can say box that you can put stuff in. variable store different type of values but for now we will just talk about strings. Whenever we store value in the variable it gets stored in the memory so when we needed, we can use that variable. Like other programming languages we don't need to define data type with variable because it infer/know by itself when the data is entered.

For example,
```
a = 10
print(a)
print(type(a))
a = 10.5
print(a)
print(type(a))
```

Now when we will execute this result will be:

10

<class 'int'>

10.5

<class 'float'>

So, first when the value was 10 it shows **int** when we tried to find data type with **type() function**. When the value was changes to 10.5, we call this overwrite. Then it shows 10.5 in output box and when we again checked its data type it was **float** so changes automatically with change in the data.

Now we make a string variable and assigned it a **string**. Code will be:

```
city = "Abbottabad"
print(city)
print(type(city))
```

Now the result is like:

Abbottabad

<class 'str'>

So, it displays the string stored in it and also the data type it shows was **str** which **string**. If we again change the value in the city and put any integer value it will change its data type to integer.

```
city = 12
print(city)
print(type(city))
```

Result:

12

<class 'int'>

Now we will do some other things with strings stored in the variables:

```
Student_name = "Usman"
Father_name = "Yaqoob"
print(Student_name + Father_name)
```

So, we made two variables and assigned them a string each. Then we use print function and between two variable we gave sign +. We did not give it for sum we gave it **Concatenation** or **joining** of two strings.

Result: **UsmanYaqoob**

For the sum two numbers we will write the following code:

```
num1 = 100
num2 = 200
```

```
sum = num1+num2
print(sum)
```

So, the result will be **300.** We made **3 variables** two for numbers and 1 for **result** we **added** two numbers and then stored its result in the 3rd variable and displayed its value.

Important thing we use = to between variable name and value. In programming we actually call it ***Assignment Operator.***

Now we will Study about some **Operators** like, +, -, *, /.

We will use all the operator in the way as we use + operator for addition of two numbers.

Subtraction:

```
num1 = 100
num2 = 200
subtraction = num1-num2
print(subtraction)
```

Result: **-100**

Now if we subtract float from integer like:

```
num1 = 100
num2 = 200.5
subtraction = num1-num2
print(subtraction)
```

Result: **-100.5**

Always when one of value in subtraction is float result will be float.

Multiplication:

```
num1 = 100
num2 = 200
multiplication = num1 * num2
print(multiplication)
```

Result: **20000**

Division:

```
num1 = 100
num2 = 200
div = num2 / num1
print(div)
```

Result: **2.0**

It gave result in float because python by default do division in float. So, for integer division we have to use //.

```
num1 = 100
num2 = 200
div = num2 // num1
print(div)
```

Result: **2**

// actually, cuts down the decimal part so if answer will be 0.5 only 0 will appear.

8. **If Else Statement:**

It is the logical expression used with condition. If the statement will be true the block of **If** statement will be executed.

Syntax:

If *condition*:

Things to be executed

Unlike C++ we don't need to use the curly brackets to start the block of statements but the cursor will be indented (lit bit forward) which shows that block of statement is started. It improves readability.

Example:

```
num1 = 100
num2 = 200
if num1<num2:
    print("num1 is less than num2 ")
    print("num2 is greater than num1 ")
```

Result:

num1 is less than num2

num2 is greater than num1

Explanation:

As the condition we gave was true so block of statement executed. Now if we make condition false like:

```
num1 = 100
num2 = 200
if num1>num2:
    print(“num1 is less than num2 ")
    print("num2 is greater than num1 ")
```

Result:

Result will be nothing. Because condition is false. Block of statements will not be executed.

With else statement we have else if and else statements too, their syntax and function is shown in the following code:

```
a = 10
b = 20
c = 30
if a>b and a>c:
    print("a is greater than b and c")
elif b>a and b>c:
    print('b is greater than a and c')
elif c>a and c>b:
    print("c is greater than a and b")
else:
    print("Numbers are not valid")
```

Result: c is greater than a and b.

9. **Comments:**

When we code, we also have to explain some lines that we have done so that person who is reading can understand what we did. Or if we come after months and see our code, we can know what we did. We use **comments** for that purpose.

Comments are lines of text ignores by python while execution of code. It means that it will be not readable and executable by interpreter while execution.

Types of comment:

- *Single line comment:*
  We use # to comment a single line.

```
# I am a comment
```

- *Multi-line comment:*
  For this we use triple quotes.

```
"""I am a
  comment
please
 """
```

10. **User Input:**

We use function input() to get input from the user it will prompt/ask the user to input.

Syntax:

```
a = input("Enter any number")
print(a)
```

It will firstly be asked for the number and it will print it.

Result:

Enter any number12

12

So, in this question one tricky thing is when we will ask for input and if person put number as input python will take it as string so if we then perform addition function to number it will concatenate it.

Like:

```
a = input("Enter any number")
b = input("Enter any number")
c = a+b
print(c)
```

Its result will be:

Enter any number12

Enter any number12

1212

So, to solve this we have to use **typecasting**. Type casting is changing one data type to other. Simply we have to mention that a and b, are of **int** value.

```
a = int(input("Enter any number"))
b = int(input("Enter any number"))
c = a+b
print(c)
```

Result:

Enter any number12

Enter any number12

24

11. **Lists:**

List is the thing that helps us to store more than one value in the single variable. Usually when we give new value to a variable, it over writes its older value means its older value vanished away. This concept is similar to the array.

It is data structure that is mutable, or changeable, ordered sequence of elements.

Each element in the list is known as ***Item.*** List is defined by having its items in ***square brackets*** **[]** and items are separated b ***comma*** ‘**,**”.

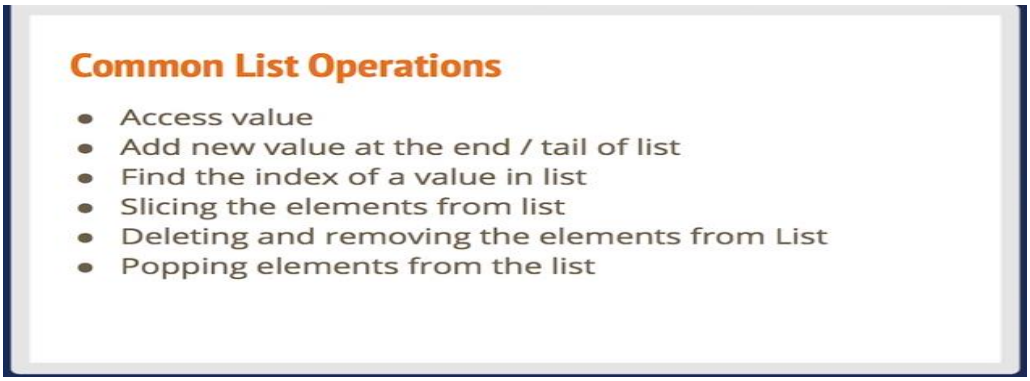
Every item in the list have special identity through which we can access to it, which is actually a number known as ***Index.*** Index of 1st item is always **0** and increase with number of items. First item will have index 0, second will have index 1 and so on.

Syntax and different examples of list are shown in following ***Code:***

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan"]
print(friends)
numbers = [1,2,3,4,5,6]
print(numbers)
float = [2.3,1.3,12.2,1.2,12.1]
print(float)
mix = ["Usman", 12, 4.3, True, False]
print(mix)
```

Result:

['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan']
[1, 2, 3, 4, 5, 6]
[2.3, 1.3, 12.2, 1.2, 12.1]
['Usman', 12, 4.3, True, False]



12. List - In Practice:

In last topic we discovered how to make list and discussed its syntax. Now we are going to know ***how to access the members/items in the list.***

- To ***access the item*** in the list we have to write name of the list and then in square brackets we have to **index** of the element or item we want.
  This shown in the following code in which we print each item of list:

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan"]
print(friends[0])
print(friends[1])
print(friends[3])
print(friends[4])
print(friends[5])
```

Result:
Saad
Huzaifa
Danial
Umar
Furaqan

- To ***check number of members*** in the list we use ***len()*** function. We have to write name of list in ().
  Code:

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan"]
print(len(friends))
```

Result:
6

- Next thing is ***Adding item to the list*** through ***append()*** function. Same as earlier we have to write name of list in (). But first we have to write name of list and after

it put dot '.'. And then write append function. The thing is it will always add that item that we want to enter in the end of the list.

Syntax:

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan"]
print(friends)
friends.append("Ali")
print(friends)
```

Result:

['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan']
['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan', 'Ali']

As you can see Ali is added at the end of the list.

- Next is ***Insert().*** Which is similar to append but the main difference is it is used to add **item where ever we want in the list.** Syntax is similar to append one like write name of list then dot and write **insert()** but in brackets instead of directly writing item **we have to first write the index** where we want this item.

Syntax:

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan"]
print(friends)
friends.insert(3,"Ali")
print(friends)
```

Result:

['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan']
['Saad', 'Huzaifa', 'Hassaan', 'Ali', 'Danial', 'Umar', 'Furaqan']

Clearly Ali is added at fourth place which means at index 3.

- ***Extend()*** is similar to the append function that adds item in the end. But the difference is that when we want to add more than one item in the end of list we use **Extend()** function. Generally, we add list at the end of list with extend.
It can be used instead of append because it can add one item too in the list.

Syntax:

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan"]
print(friends)
friends.extend(["Ali","Abdul Raheem"])
print(friends)
```

Result:

['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan']
['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan', 'Ali', 'Abdul Raheem']

- ***count()*** is another useful function related to lists. It is used to count the number of particular items in the list.
  Syntax:

```
numbers = [1,2,3,3,4,5,6,7,8,9]
print(numbers.Count(3))
```

  Result:
  2

- ***Index()*** function is also a function used in list. It is obvious from the name that it is related with index. Yes, it is actually used to find the index of item that at what place it exists in the list.
  Syntax:

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan"]
print(friends)
print(friends.index("Hassaan"))
```

  Result:
  ['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan']
  2

- Next function that can be perform on the list is ***clear().*** It is used to clear the list. It removes all the items from the list.
  Syntax:

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan"]
print(friends)
friends.clear()
print(friends)
```

  Result:
  ['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan']
  []

- ***Copy()*** function show it work from the name that it makes copy of the list. And all item can be store in the other list.
  Syntax:

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan"]
freinds_foreever=friends.copy()
print(freinds_foreever)
```

  Result:
  ['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan']

It copies the list by **values**. There is another copying which is called as copying by reference. In which we assign **reference** of one list to other.

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan"]
freinds_foreever = friends
print(freinds_foreever
```

It will give the same result:
['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan']
So, here there is a difference if I change or add item in friends list it will be directly added in friends_foreever list too.

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan"]
freinds_foreever = friends
friends.append("Ali")
print(freinds_foreever)
```

Its result will be:
['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan', 'Ali']

On the other hand, in by value if change happens in any list it will not be shown in other list. This can be seen by the following lines of code:

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan"]
freinds_foreever=friends.copy()
friends.append("Ali")
print(friends)
print(freinds_foreever)
```

Result will be:
['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan', 'Ali']
['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan']

- One of the important functions of list is **remove()** function. It is used to remove a specific item from the list.
  Syntax:

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan","Snake"]
print(friends)
friends.remove("Snake")
print(friends)
```

Result:
['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan', 'Snake']
['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan']

Second method of removing any element from the list is through **del Statement** , we use keywork **del** *name of list[index of item to be removed]*. It is the syntax which can be more cleared by following set of code:

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan","Snake"]
print(friends)
del friends[6]
print(friends)
```

Result:
['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan', 'Snake']
['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan']

- **pop()** is another function through which we can remove last item or any item from the list and can store it in other list. It is the specialty that it stores removed value too
  This will get clear if you see its ***Syntax:***

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan","Snake"]
popped = friends.pop()
print(f"This friend is popped from the list {popped}.")
print(f"The remaining friends are {friends}")
```

Result will be:
This friend is popped from the list Snake.
The remaining friends are ['Saad', 'Huzaifa', 'Hassaan', 'Danial', 'Umar', 'Furaqan']

Pop through index:

```
friends = ["Saad","Huzaifa","Hassaan","Danial","Umar","Furaqan","Snake"]
popped = friends.pop(2)
print(f"This friend is popped from the list {popped}")
print(f"The remaining friends are {friends}")
```

Result:
This friend is popped from the list Hassaan
The remaining friends are ['Saad', 'Huzaifa', 'Danial', 'Umar', 'Furaqan', 'Snake']

- ***Sort()*** is the function related to list used to sort the list either in the ascending order or in descending order. When we use sort() function with default it organizes the list in the ascending order because the reverse in the parameters of function is given false value. So, for descending order we have to write **reverse=True** in brackets or we say parameter of this function.

List Sorted in Ascending order Syntax:

```
numbers = [1,23,4,5,6,7]
numbers.sort()
print(numbers)

Alphabets= ["a","f","d","c","e","e","h","g"]
Alphabets.sort()
print(Alphabets)
```

Result:
[1, 4, 5, 6, 7, 23]
['a', 'c', 'd', 'e', 'e', 'f', 'g', 'h']
It sorts the first and second list in ascending order with respect to numbers and alphabets.

List Sorted in Descending order Syntax:

```
numbers = [1,23,4,5,6,7]
numbers.sort(reverse = True)
print(numbers)

Alphabets= ["a","f","d","c","e","e","h","g"]
Alphabets.sort(reverse = True)
print(Alphabets)
```

Result:
[23, 7, 6, 5, 4, 1]
['h', 'g', 'f', 'e', 'e', 'd', 'c', 'a']

- ***Reverse()*** is the separate function related to list that reverse the items of list. Don't consider any order just reverse all items.
  Syntax:

```
numbers = [1,23,4,5,6,7]
numbers.reverse()
print(numbers)
Alphabets= ["a","f","d","c","e","e","h","g"]
Alphabets.reverse()
print(Alphabets)
```

Result:
[7, 6, 5, 4, 23, 1]
['g', 'h', 'e', 'e', 'c', 'd', 'f', 'a']

13. Accessing List Elements:
Already done in detail in previous topic.

14. Adding Values in Lists:
Already done in detail in previous topic.

15. Adding Values in List – In practice:

Already done in detail in previous topic.

16. Finding Values in Lists:

There are **two** ways to find a specific value/item in the list.

- First one is through using index if you. **Index()** can be used, just give name of item in the brackets and it will tell you that at what index that item exists.
  Syntax:

```
Alphabets= ["a","f","d","c","e","e","h","g"]
print(Alphabets.index("h"))
```

Its *Result* will be:
6

- **In operator** provides the second way to find any item in the list easily. It will return true value if that item is present in it and false if it is not. *REMEMBER PYTHON IS CASE SENSITIVE.*
  Syntax:

```
Alphabets= ["a","f","d","c","e","e","h","g"]
print("h" in Alphabets)
print("u" in Alphabets)
```

Result:
True
False

17. Slicing Elements from Lists:

Slicing is the method used to get more than one items from the list. Its syntax is like:
Name of list[Starting element index: index of element required+1]
Code:

```
Alphabets= ["a","f","d","c","e","e","h","g"]
print(Alphabets[1:5])
Result:
```

['f', 'd', 'c', 'e']

18. Slicing - In Practice:

As we are slicing elements from the list so it will return a list too. Which will have more than 1 elements.

- You can get elements with second method of using index. For that you to write "-" with index. Last element has index -1, second last -2, third last -3 and so on.

```
In [56]:
#index     -5      -4      -3      -2      -1
students=["Ali", "Faisal", "Saleem", "Hamza", "Kashif"]

#index      0       1       2       3       4
```

Code:

```
Alphabets= ["a","f","d","c","e","e","h","g"]
print(Alphabets[-4:-1])
```

Result:
['e', 'e', 'h']
You can get elements only in forward direction.

- If you don't give left and right index in brackets, it will return the whole list as it.

```
Alphabets= ["a","f","d","c","e","e","h","g","f","d","r"]
print(Alphabets[:])
```

Result:
['a', 'f', 'd', 'c', 'e', 'e', 'h', 'g', 'f', 'd', 'r']

- If you don't give the ending index it will print all elements from starting index which is specified to last element's index.

```
Alphabets= ["a","f","d","c","e","e","h","g","f","d","r"]
print(Alphabets[2:])
```

Result:
['d', 'c', 'e', 'e', 'h', 'g', 'f', 'd', 'r']

- Similarly, if you don't give starting index it will by default starts form index 0.

```
Alphabets= ["a","f","d","c","e","e","h","g","f","d","r"]
print(Alphabets[:4])
```

Result:
['a', 'f', 'd', 'c']

-
```
Alphabets= ["a","f","d","c","e","e","h","g","f","d","r"]
print(Alphabets[3:-1])
```

Result:
['c', 'e', 'e', 'h', 'g', 'f', 'd']

- We can skip the elements in between getting elements .So, here is the method used:

```
numbers= [1, 2, 3, 4,5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
print(numbers[1:14:2])
```

Result:
[2, 4, 6, 8, 10, 12, 14]
First value in last most is starting index, in middle we have ending element **index+1 and in the very right ,most we have stepping numbers as we gave 2 so, it** skips 1 element if we give 3, it will skip 2 elements and so on.

```
numbers= [1, 2, 3, 4,5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
print(numbers[::2])
```

Here we don't give starting and ending index so it will start from first element and will till last element.
So, the result will be:
[1, 3, 5, 7, 9, 11, 13, 15]

19. Removing …...:
Done.

20. Popping . …..:
Done.

21. Tuples:
Tuples is like a list in which multiple element are stored in single variable There is only one difference between Tuples and list which is:
We can change the elements in the and can remove them or can add elements in it but in tuple we can not do all of that, they are immutable/unable to change.
Syntax of tuple:
Name of tuple = (elements)

22. Tuple- In Practice:
We can check length of tuple, can count number of specific elements, can find index of any element and can access to elements of tuple but we cannot delete or change the elements from tuple.

Following lines of code show all of function that can be performed on tuple:

```python
tuple = (1,2,4,2,3,5)
print(tuple)

count =  (tuple.count(2))
index = (tuple.index(4))
length = (len(tuple))

print(f"Number of 2 in tuple = {count}")
print(f"Index of 4 is {index}")
print(f"Total elements in tuple are {length}")
```

Result:

(1, 2, 4, 2, 3, 5)
Number of 2 in tuple = 2
Index of 4 is 2
Total elements in tuple are 6