WhiteHat Jr
Live online Coding for Kids

<table>
<tr><th>Topic</th><th colspan="2">SEQUENCE DATA TYPE &amp; LOOPS</th></tr>
<tr><td><b>Class Description</b></td><td colspan="2"><b>The student will learn about the lists and range in sequence data type in Python. The student will also understand for loops in Python and how to access lists using for loops.</b></td></tr>
<tr><td><b>Class</b></td><td colspan="2"><b>PRO C98</b></td></tr>
<tr><td><b>Class time</b></td><td colspan="2"><b>45 mins</b></td></tr>
<tr><td><b>Goal</b></td><td colspan="2">● Learn about the sequence data type.
● Learn about for loops in Python.
● Learn to use for loops to access lists elements.</td></tr>
<tr><td><b>Resources Required</b></td><td colspan="2">● Teacher Resources:
○ Laptop with internet connectivity
○ Earphones with mic
○ Notebook and pen
○ Smartphone
● Student Resources:
○ Laptop with internet connectivity
○ Earphones with mic
○ Notebook and pen</td></tr>
<tr><td><b>Class structure</b></td><td><b>Warm-Up</b>
<b>Teacher-led Activity 1</b>
<b>Student-led Activity 1</b>
<b>Wrap-Up</b></td><td><b>10 mins</b>
<b>10 mins</b>
<b>20 mins</b>
<b>5 mins</b></td></tr>
<tr><td colspan="3"><b>WARM-UP SESSION - 10 mins</b></td></tr>
<tr><td colspan="3"><b>Teacher Starts Slideshow</b>
<b>Slide 1 to 3</b>
Refer to speaker notes and follow the instructions on each slide.</td></tr>
</table>

Hey <student's name>. How are you? It’s great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** ● Greet the student. ● Revision of previous class activities. ● Quizzes.	**ESR**: Hi, thanks! Yes I am excited about it! Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
Continue WARM-UP Session **Slide 4 to 18**	
Following are the session deliverables: ● Appreciate the student. ● Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	
Teacher Action	**Student Action**
In today’s class, we are going to learn about Sequence Data Types in Python. Before we start learning, can you tell me what’s a sequence? Amazing! Can you give me some examples where you see a list of things? ***Note**: Encourage the student to give answers and be more involved in the discussion.*	**ESR**: A sequence is a list of things. **ESR**: Varied.

<table>
<tr><td>There could be so many examples…
A list of TV Shows you like.
A list of colors.
A list of birthday celebration items.
A list of marks.
A list of images and so on.

All these examples have multiple objects, right?

To summarize, a sequence is basically a collection of objects.

We are going to learn some sequence data types in Python today, which we mostly use.

Are you excited?

Let’s get started.</td><td>

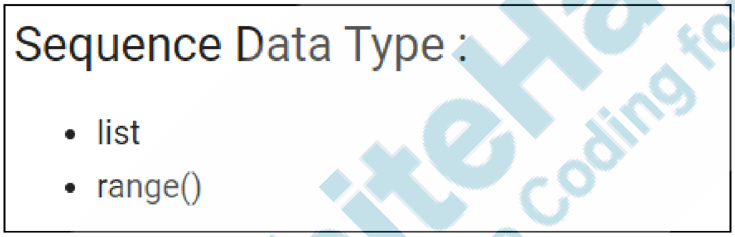
<b>ESR</b>: Yes.</td></tr>
<tr><td colspan="2"><b>Teacher Ends Slideshow</b></td></tr>
<tr><td colspan="2"><b>TEACHER-LED ACTIVITY - 10 mins</b></td></tr>
<tr><td colspan="2"><b>Teacher Initiates Screen Share</b></td></tr>
<tr><td colspan="2"><b><u>ACTIVITY</u></b>
● <b>Python Sequence Data Types</b>
● <b>For Loops</b></td></tr>
<tr><td><b>Teacher Action</b></td><td><b>Student Action</b></td></tr>
<tr><td>We are going to learn about three sequence data types in Python.

There are 3 types of sequence data types in Python:</td><td></td></tr>
</table>

- List
- Range
- Tuples

In today's class, we will understand lists and range in Python.

***Note**: The tuples data type is covered as part of ADDITIONAL ACTIVITY. If the student is able to cover the list and range, please take out time to explain tuples. Tuples, however, will be discussed wherever used in the projects in the upcoming classes.*



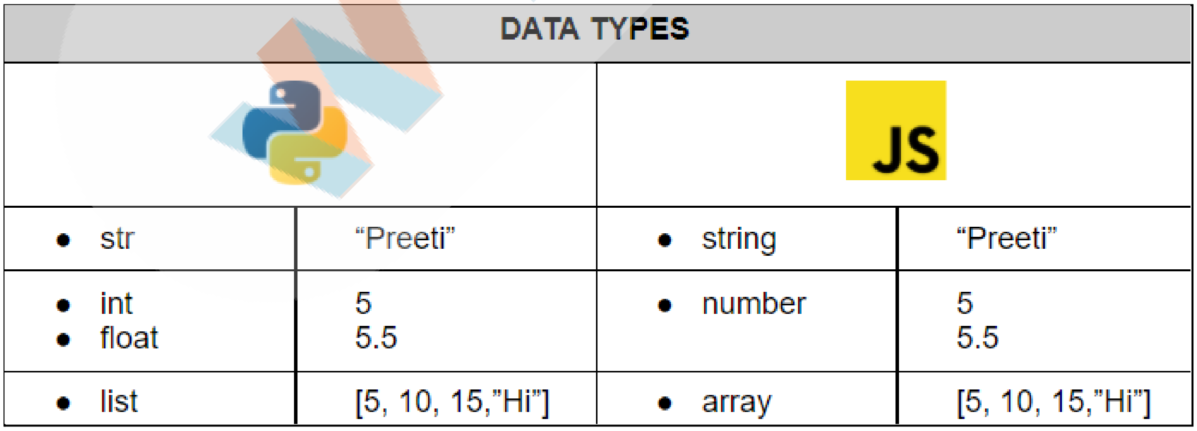
Sequence Data Type :

- list
- range()

Can you tell how we represent multiple objects using one variable in JavaScript?

Superb!

ESR: We use "**array**".



DATA TYPES			
		JS	
● str	"Preeti"	● string	"Preeti"
● int ● float	5 5.5	● number	5 5.5
● list	[5, 10, 15,"Hi"]	● array	[5, 10, 15,"Hi"]

In Python, we use “**list**” for multiple objects.

There are many similarities and differences between lists in Python and arrays in JavaScript.

Let’s start by taking an empty list.

We can take a variable **list1** and use a pair of square brackets, **[ ]**, to declare the empty list.

To define a list with values, we can have elements separated by a comma within the square brackets.

Elements of the list can be of any type. We can have **int**, **float**, **str**, **list**, **dict,** etc.

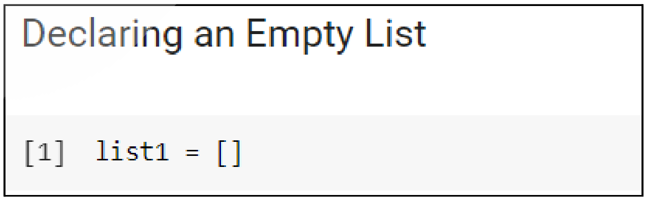
We do not have to worry about other data types for now. We will learn about those in the upcoming classes. We will use only **int** and **str** data types as the values in the list.

The teacher opens [Teacher Activity 1].

Let's take **list1** and **list2** as variable names of the list.

To define an empty list, we can just use empty square brackets.

The teacher writes the code.



Declaring an Empty List

```
[1]  list1 = []
```

Now let’s take five int elements in the **list1** and three string elements in the **list2** and print the list.

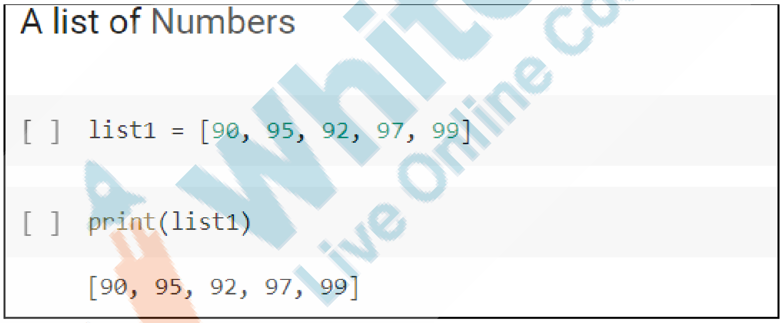
Printing a list:

1. We can use the **print()** method **OR**

2. To check how many elements are there in the list, we use a method called **len()**.

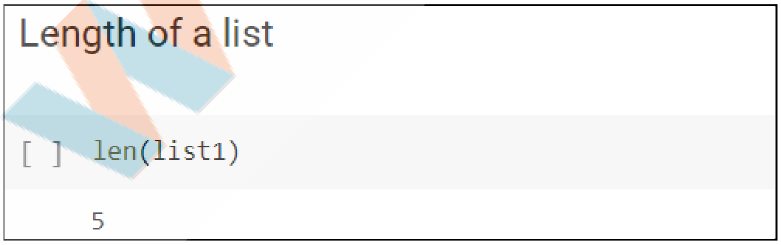
3. We can directly write the list variable name in the cell to print the complete list.
   ***Note**: This only works while working with Notebooks.*

The teacher writes the code.



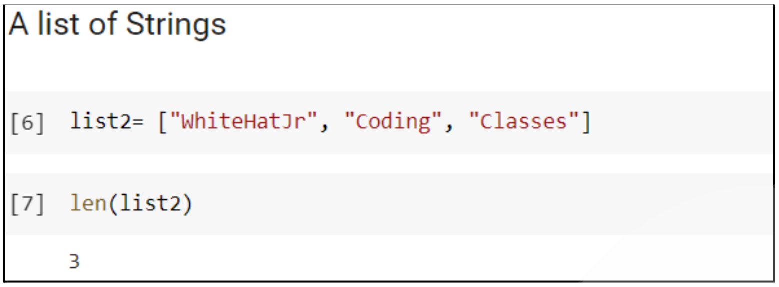
A list of Numbers

```
[ ]  list1 = [90, 95, 92, 97, 99]

[ ]  print(list1)

     [90, 95, 92, 97, 99]
```



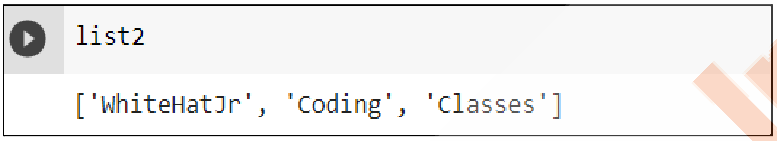
Length of a list

```
[ ]  len(list1)

     5
```



A list of Strings

```
[6]  list2= ["WhiteHatJr", "Coding", "Classes"]

[7]  len(list2)

     3
```



```
list2

['WhiteHatJr', 'Coding', 'Classes']
```

Once we assign the elements in the list, we should be able to access those elements.

Do you remember how we access elements of an array in JavaScript?
Great!

Before we can understand how to access the elements of the list using the index, Can you tell me what you understand by the index of an element?

Indexing:

- In very simple terms, the index of the element tells us the position of the element in that array if we start counting from the beginning of the array.

Why do you think we need indexes?

Let's take an example to understand that.

Let's say your class has 30 kids at school. How do you think the teacher is able to maintain the attendance of every kid in your class?

If we look at the names of each kid in the class, it's quite possible that two kids have the same name, and it will be

ESR: We used an index that starts from zero.

ESR: Varied.

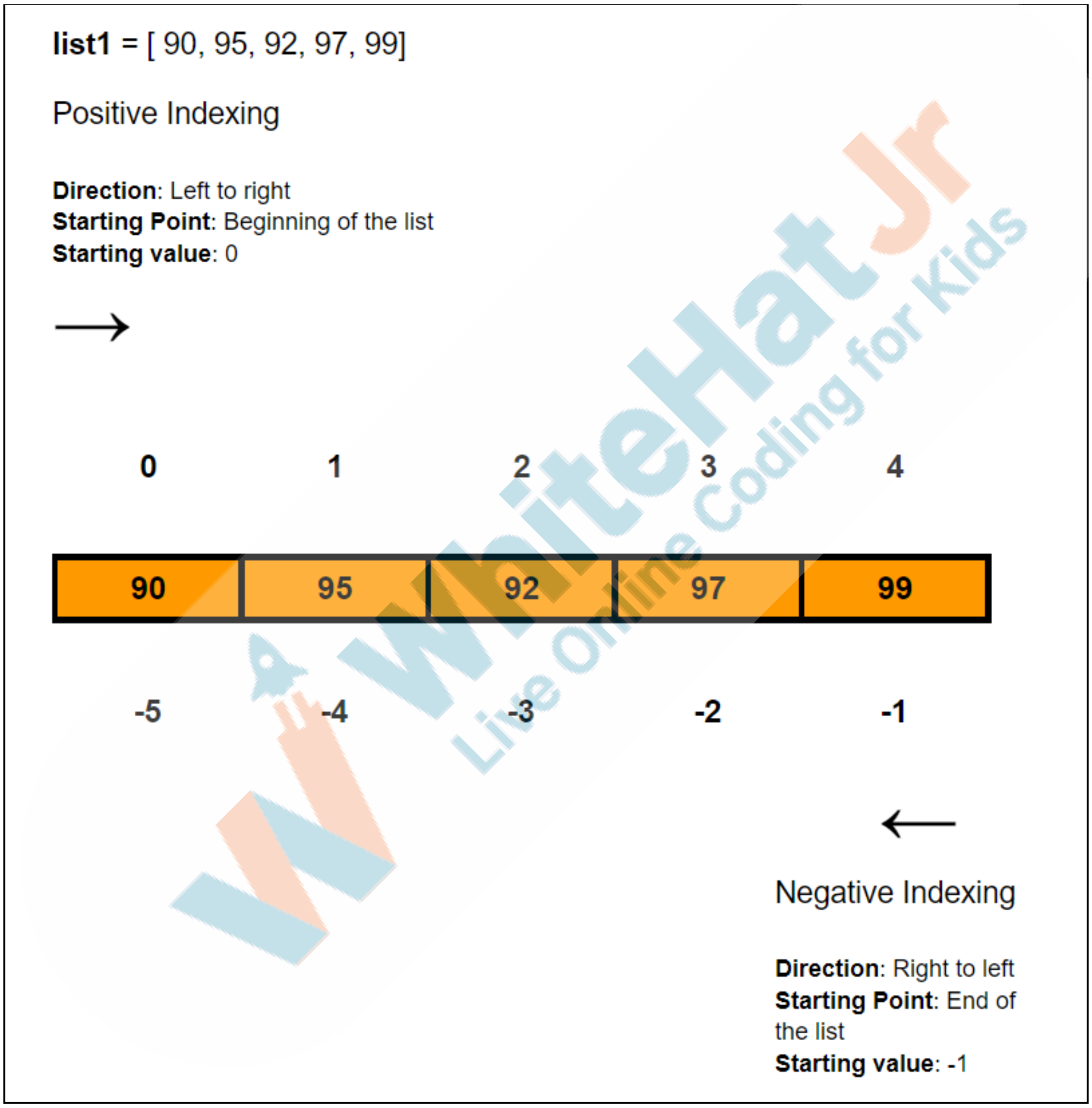
ESR: Varied.

ESR: Varied.

difficult for the teacher to keep a track of the student's attendance, and that's why every student is given a unique number as roll numbers. This makes it easy for the teachers to keep a track of the attendance.	
So indexes are basically the numbers assigned to the elements of a list that help the computer to remember the element.	
In our day-to-day life, we start numbering from one, but the computer starts counting from zero.	
To access elements of the list in Python, we use the index but in Python, we have two types of indexing: ● Positive indexing ● Negative indexing	
The positive indexing in the list is similar to the array indexing in JavaScript. It starts from 0 at the beginning of the list, and it keeps on going till the last element at the end of the list.	
The negative indexing in the list starts from -1 at the end of the list and keeps on going till the beginning of the list.	**ESR**: Yes.
***Note**: Ask the student to open the image in Student Activity 1. Let the student observe and ask questions.*	
You must be wondering why we need two types of indexing, right?	
We can easily access and manage all the elements with one type of index but let us say, we have a thousand elements in the list, and we just want to know the last five elements.	**ESR**: Yes.
Well if you have only one kind of indexing, you will start counting from zero, and then you will go till the end, and you will then pick up the last five elements, right?	

But with the help of a negative index, we can quickly pick up 5 elements just by using an index from - 1 to - 5.

Python Developers have made it so easy to access elements of the list in any direction.



list1 = [90, 95, 92, 97, 99]

Positive Indexing

Direction: Left to right
Starting Point: Beginning of the list
Starting value: 0

0	1	2	3	4
90	95	92	97	99
-5	-4	-3	-2	-1

Negative Indexing

Direction: Right to left
Starting Point: End of the list
Starting value: -1

Accessing the elements of the list:

- To access elements using the index, we specify the index inside the square brackets.

In Python, we can access elements in two ways:

- One element at a time.
- Multiple elements.

We will understand each of these. We will begin with accessing one element at a time.

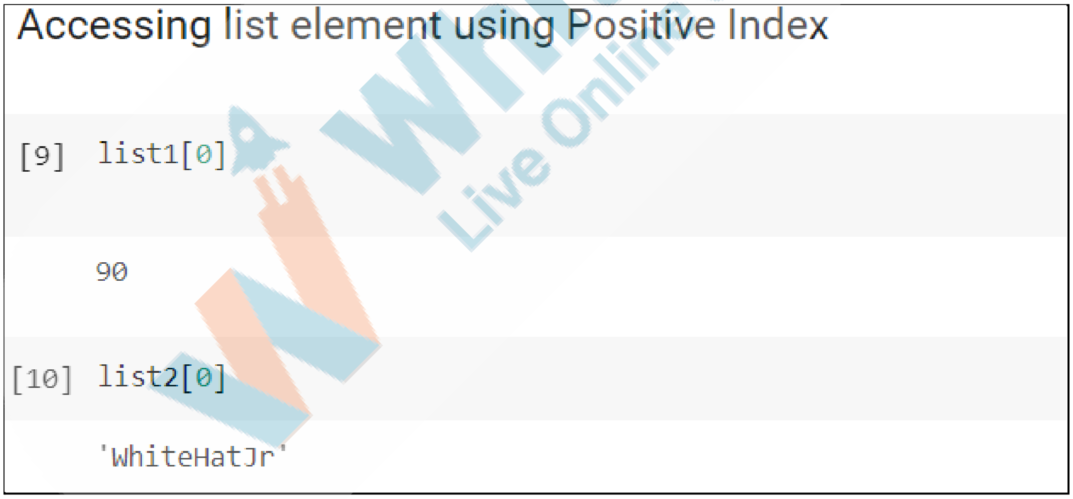
Let's try to access the first element of **list1** and the first element of **list2**

Which indexing would you use, positive or negative?

Since it is the first element, we should definitely use positive indexing and for the first element, the index is 0.

The teacher writes the code.

ESR: Positive.



Accessing list element using Positive Index

```
[9]  list1[0]

     90

[10] list2[0]

     'WhiteHatJr'
```

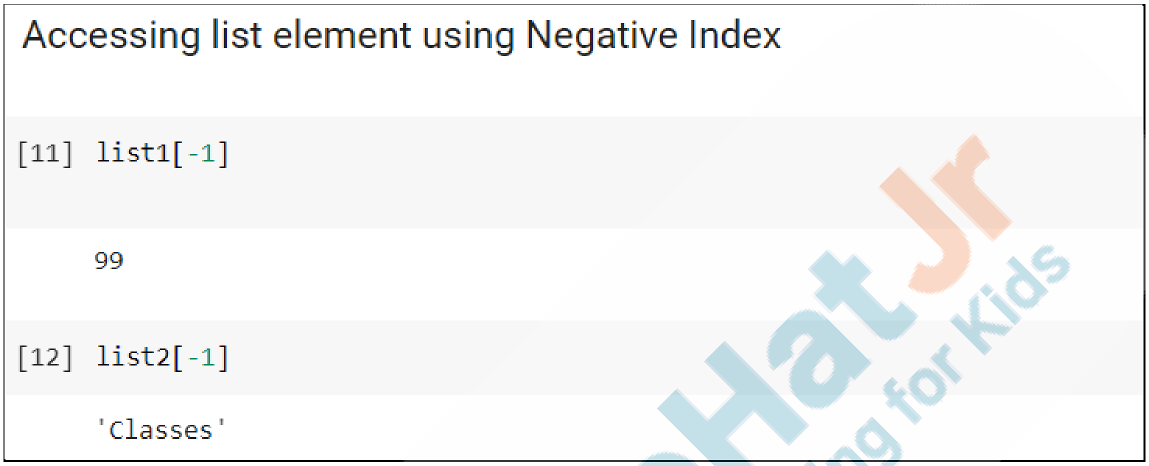
Let's try to access the last element of **list1** and the first element of **list2**.

Which indexing would you use, positive and negative?

ESR: Negative.

Since it is the last element, we should not start from zero, but we will start from -1.

The teacher writes the code.



Accessing list element using Negative Index

```
[11] list1[-1]

     99

[12] list2[-1]

     'Classes'
```

Now to access multiple elements in one go we can use the slice operator.

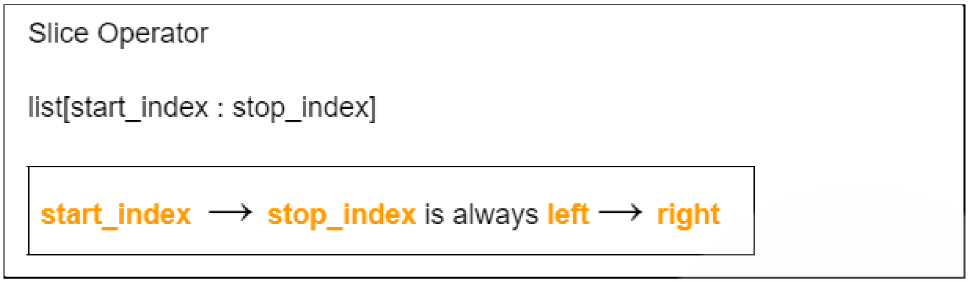
The **slice operator** is represented by the symbol colon (**:**).

To access multiple elements, we need to tell the computer from where the computer needs to start picking up the element and where the computer needs to stop taking up the elements of the list.

To access multiple elements of the list, the **start index** and **stop index** are mentioned, **separated by the colon** in the square brackets.

ESR: Yes.

***Note**: The following is only for the teacher's understanding.*



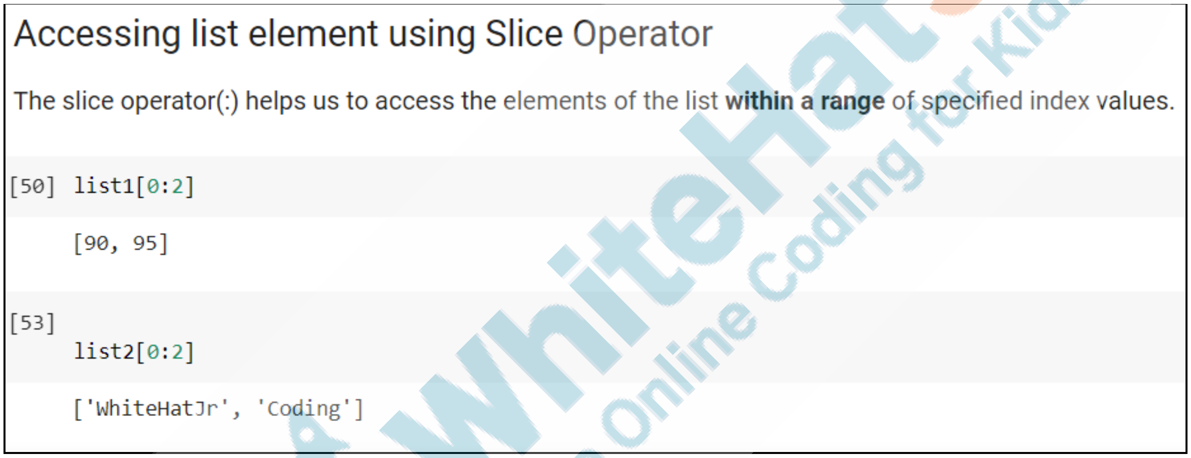
Slice Operator

list[start_index : stop_index]

start_index ⟶ stop_index is always left ⟶ right

Let's understand this by an example.

The teacher writes the code.

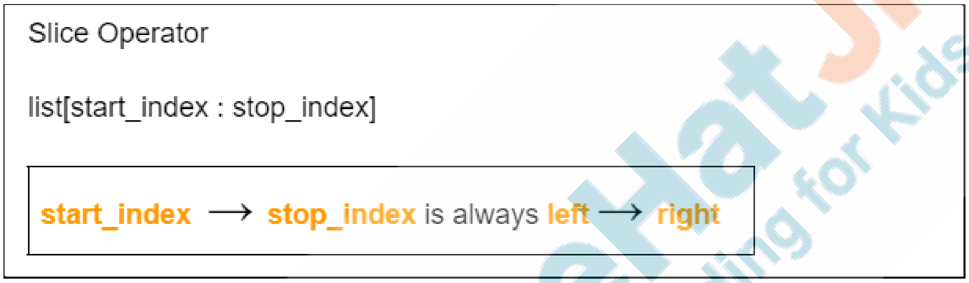


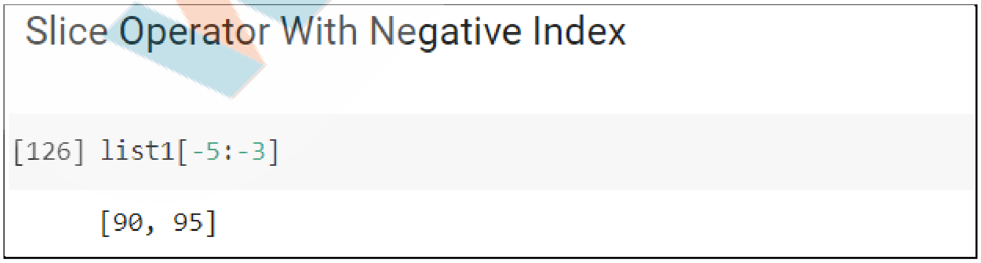
Accessing list element using Slice Operator

The slice operator(:) helps us to access the elements of the list **within a range** of specified index values.

```
[50]  list1[0:2]

      [90, 95]

[53]
      list2[0:2]

      ['WhiteHatJr', 'Coding']
```

So we mentioned the start index as 0 and the stop index as 2. How many elements should we have? ***Note**: Encourage the student to give answers and be more involved in the discussion.* If we include, both the start index and the stop index we have three indices 0, 1, and 2 hence, we should get three elements but in Python, this is not correct! The **stop index is NOT included while accessing the elements**. Only elements before the stop index are	**ESR:** We should get 3 elements for indexes 0, 1, and 2.

returned from the start index. Similarly, we can also use negative indexes with the slice operator. To use the negative index, we have to start from the indexes from the left side only (which is opposite to assigning negative indexes to the elements).	
***Note**: The following is only for the teacher's understanding.*  Slice Operator list[start_index : stop_index] start_index ⟶ stop_index is always left ⟶ right	
Let's understand that by an example. Here we will get the values from -5 to -4. Can you tell me why? Great! *The teacher writes the code.*	**ESR:** We do not get the value at the stop index while accessing the elements using the slice operator.


Slice Operator With Negative Index
`[126] list1[-5:-3]`
`[90, 95]`	

This result is the same as list1[0:2]	

list1 = [90, 95, 92, 97, 99]

Example: list1[0 : 2] OR list1[-5 : -3]

→

start_index		stop_index		
0	1	2	3	4
90	95	92	97	99
-5	-4	-3	-2	-1

→

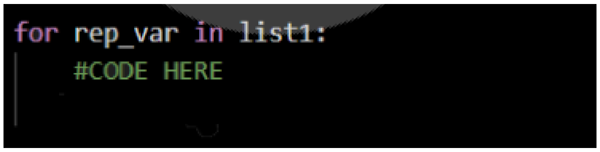
We have learned about the lists and how to access its element using the slice operator.

We can also access the elements of a list using a **for** loop.

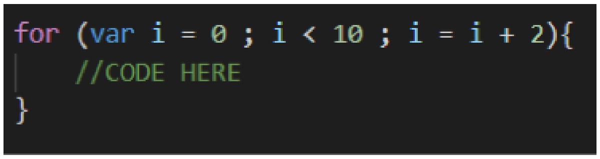
Let's understand how a for loop is written in Python programming language syntax.

FOR LOOPS

JS

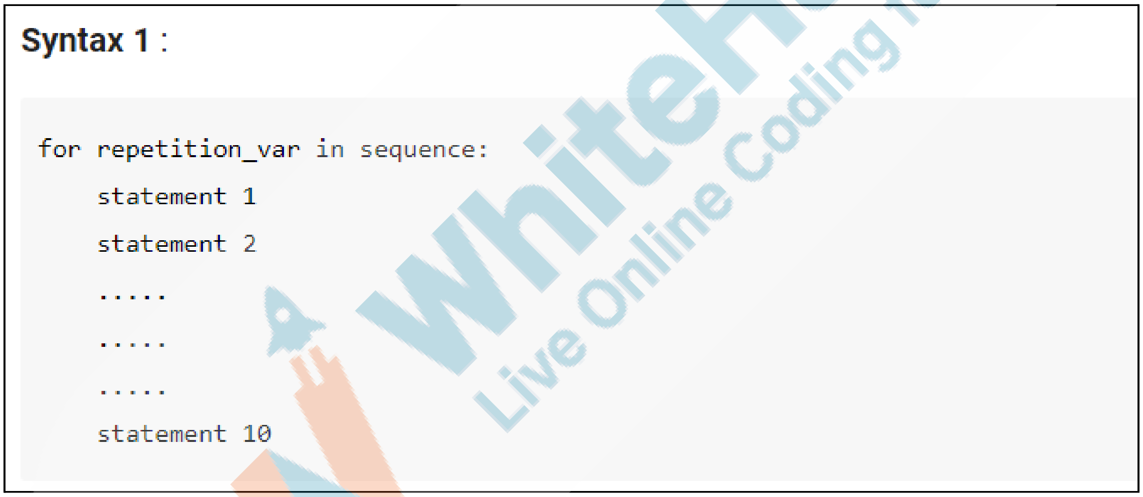


```
for rep_var in list1:
    #CODE HERE
```

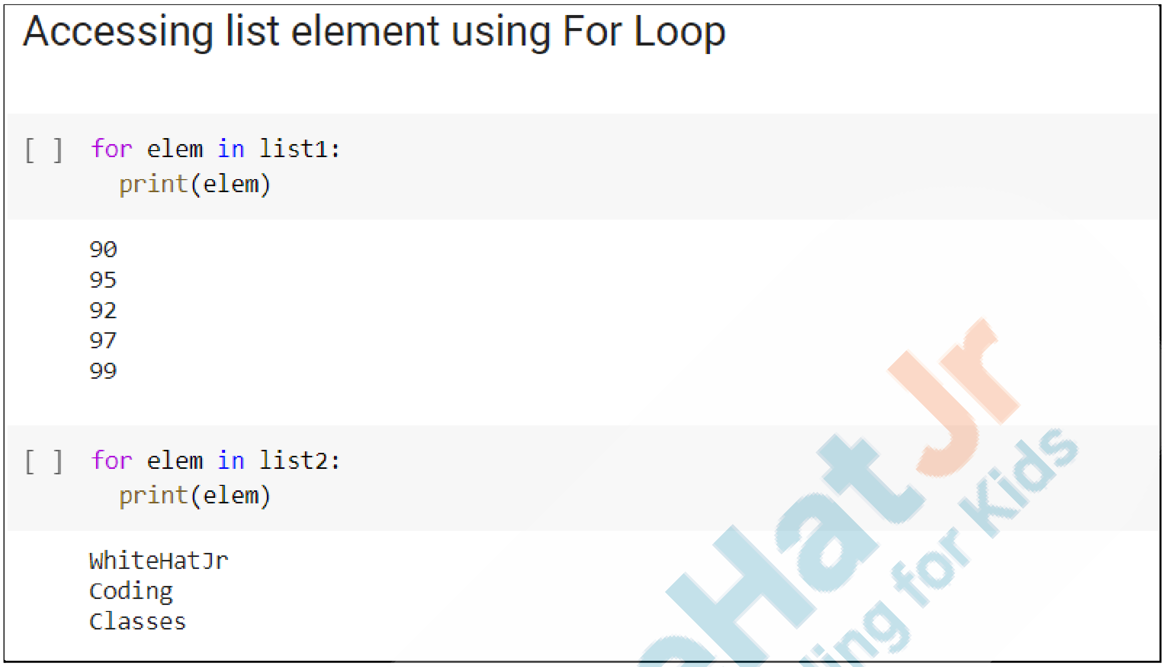


```
for (var i = 0 ; i < 10 ; i = i + 2){
    //CODE HERE
}
```

Do you remember how we used to write a **for** loop in JavaScript? In Python, we also use the **for** keyword, **loop repetition variable** along with **in** keyword. The **in** operator helps to trace through all the elements of the sequence without using the element's index. **Syntax:**	**ESR**: We use: ● The **for** keyword ● And a **loop repetition** variable.

Open Visual Aid slides to discuss the syntax.

Syntax 1 :
`for repetition_var in sequence:`
`    statement 1`
`    statement 2`
`    .....`
`    .....`
`    .....`
`    statement 10`	

Let's try to print the elements of the **list1** and **list2.** *The teacher writes the code.*	



Accessing list element using For Loop

```
[ ]  for elem in list1:
       print(elem)

     90
     95
     92
     97
     99
```

```
[ ]  for elem in list2:
       print(elem)

     WhiteHatJr
     Coding
     Classes
```

We have learned a few things about the list data types in Python. We will keep learning more about lists in the upcoming classes.

Now let’s talk about another sequence data type in Python.

The **range()** method. This method gives a sequence of numbers within a given range.

Syntax: **range(start, end, steps)**

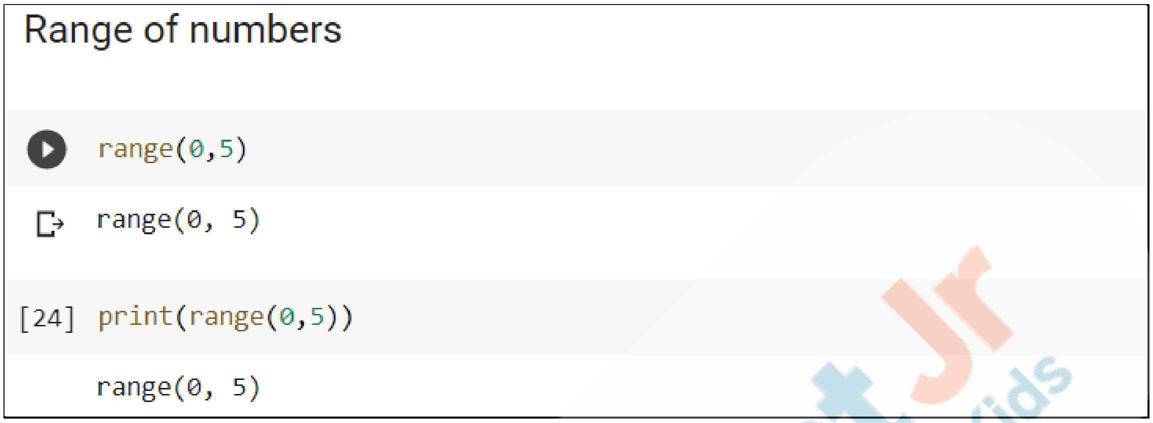
start: The number from where we want to start the sequence

end: The number where we want to end the sequence.

steps: By how much we want to increase(positive steps) or decrease(negative steps) the numbers in the range. By default, the value is positive 1 step.

Let’s try to print a range of numbers from **0** to **5**.

The teacher writes the code.



Range of numbers

```
range(0,5)
```

```
range(0, 5)
```

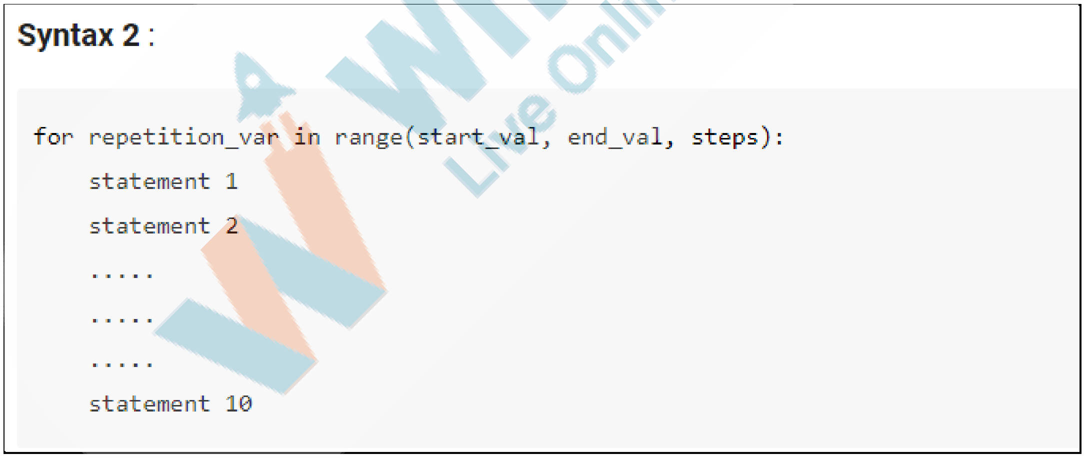
```
[24] print(range(0,5))
```

```
range(0, 5)
```

We can see that the range of numbers cannot be printed directly using the print() method. For this, we can use **for** loops.

Syntax:

Open Visual Aid Slides to discuss the syntax.

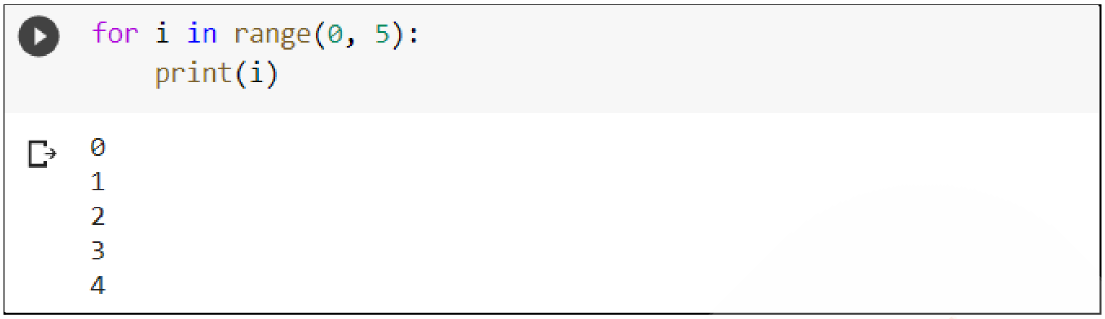


Syntax 2 :

```
for repetition_var in range(start_val, end_val, steps):
    statement 1
    statement 2
    .....
    .....
    .....
    statement 10
```

Now let’s print the numbers from 0 to 5.

The teacher writes the code.



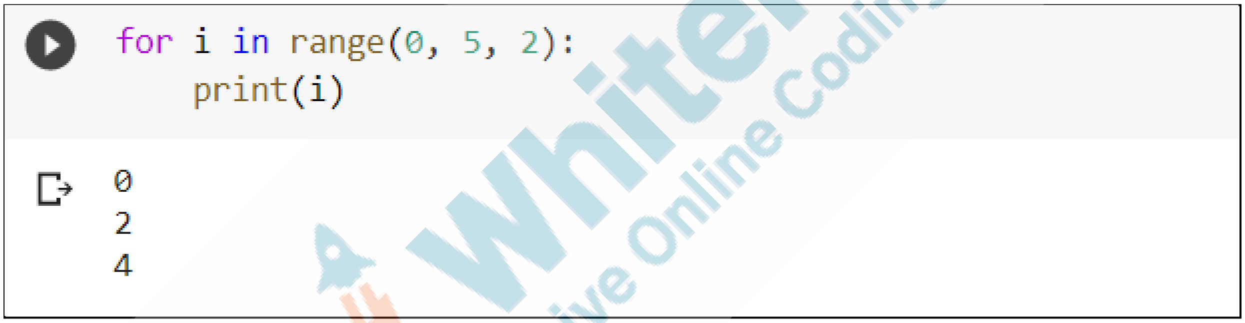
```
for i in range(0, 5):
    print(i)
```

```
0
1
2
3
4
```

Let’s print the numbers from 0 to 5 by step 2.

Note: If the steps are positive, **start** < **end**. This means we start from a lower number and keep increasing till the highest number.

The teacher writes the code.

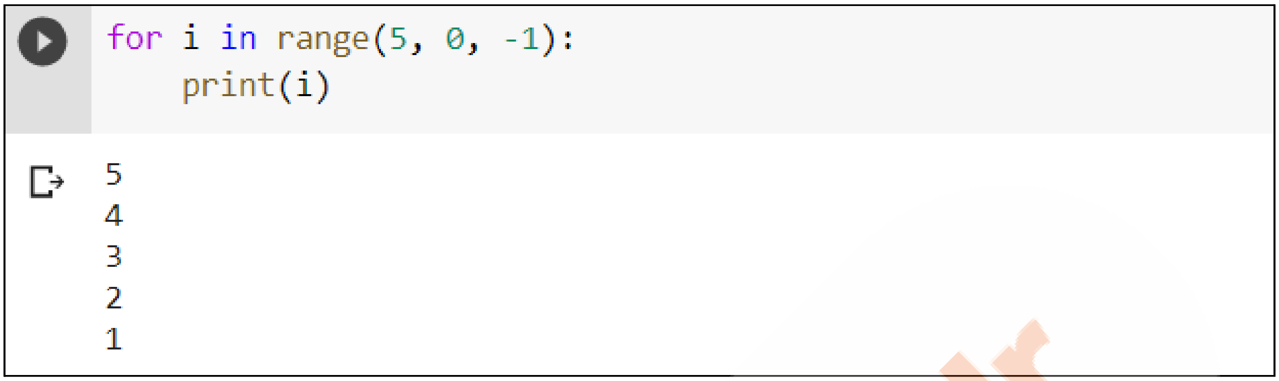


```
for i in range(0, 5, 2):
    print(i)
```

```
0
2
4
```

Let’s print the numbers from 0 to 5 by step -1.

Note: If the steps are negative, **start** > **end.** This means we start from a higher number and keep decreasing till the lowest number.



```
for i in range(5, 0, -1):
    print(i)
```

```
5
4
3
2
1
```

That was interesting!

We learned about the **lists** and **range()** method in Python.

Now you will write a function to count one element in the list.

Are you excited?

ESR: Yes.

Teacher Stops Screen Share

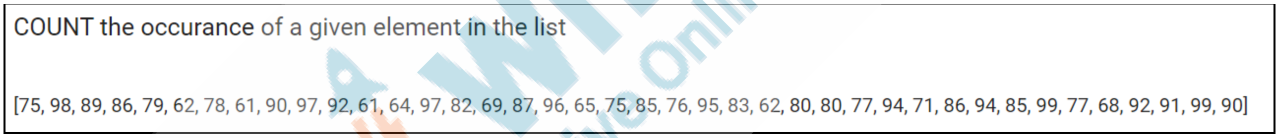
So now it's your turn.
Please share your screen with me.

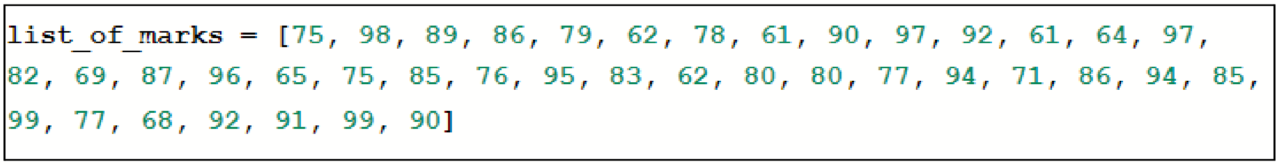
Teacher Starts Slideshow
Slide 19 to 20
Refer to speaker notes and follow the instructions on each slide.

We have one more class challenge for you.
Can you solve it?

Let's try. I will guide you through it.

Teacher Ends Slideshow

STUDENT-LED ACTIVITY - 20 mins	
- **Ask the student to press the ESC key to come back to the panel.** - **Guide the student to start Screen Share.** - **The teacher gets into Fullscreen.**	
ACTIVITY - **Write a function to count an element in the list.**	
Teacher Action	**Student Action**
Guide the student to open a Google Colab Notebook. *Student Activity 1*.	
Let’s say we want to find how many times the number 62 appears in the list of marks that all students in your class got at school. ***Note****: This is a random list of numbers. The list of numbers can vary. The student can pick any number on their own.*	
 COUNT the occurance of a given element in the list [75, 98, 89, 86, 79, 62, 78, 61, 90, 97, 92, 61, 64, 97, 82, 69, 87, 96, 65, 75, 85, 76, 95, 83, 62, 80, 80, 77, 94, 71, 86, 94, 85, 99, 77, 68, 92, 91, 99, 90]	
Guide the student to write a simple ***function to count a particular element of the list.*** Count the occurrence of the number 62 in the list: 1. Take a list of numbers as **list_of_marks** with a few random numbers. ***Note****: The list of numbers can vary. The students can pick any number on their own.*	



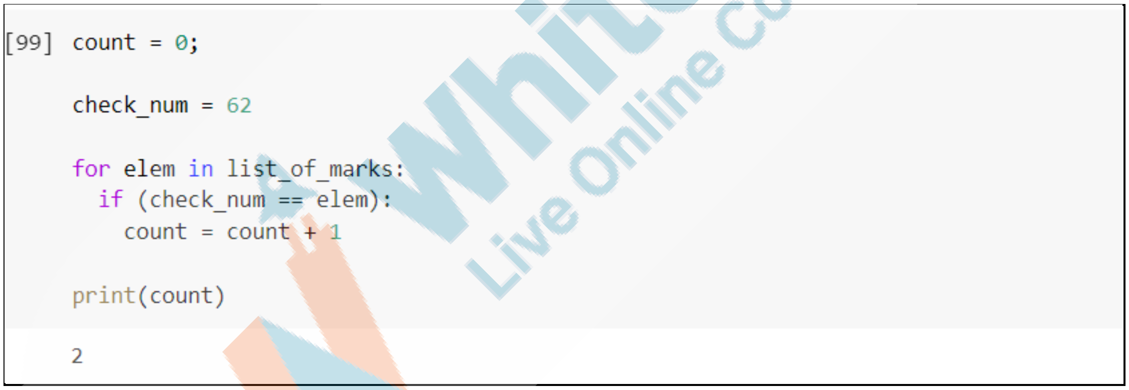
```
list_of_marks = [75, 98, 89, 86, 79, 62, 78, 61, 90, 97, 92, 61, 64, 97,
82, 69, 87, 96, 65, 75, 85, 76, 95, 83, 62, 80, 80, 77, 94, 71, 86, 94, 85,
99, 77, 68, 92, 91, 99, 90]
```

2. Take a variable, **count**, to keep a track of counting the occurrence of the number we want to count.

3. Take a variable, **check_num**, for the number that we want to count.

4. Loop through the **list_of_marks**:
    a. Check if the element of the list is the same as the **check_num**, if yes increase the **count** variable by **1**.
5. Print the count using the **print()** method.

Note: *Make sure the code is properly indented.*



```
[99] count = 0;

     check_num = 62

     for elem in list_of_marks:
       if (check_num == elem):
         count = count + 1

     print(count)

     2
```

You did really amazing work today.

We learned about the list and range() method in Python.

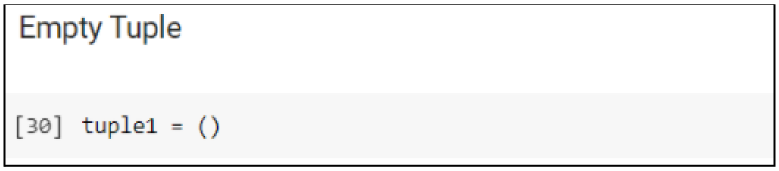
Refer to Student Activity 2 of List Indexing.

Teacher Guides Student to Stop Screen Share

WRAP UP SESSION - 5 mins

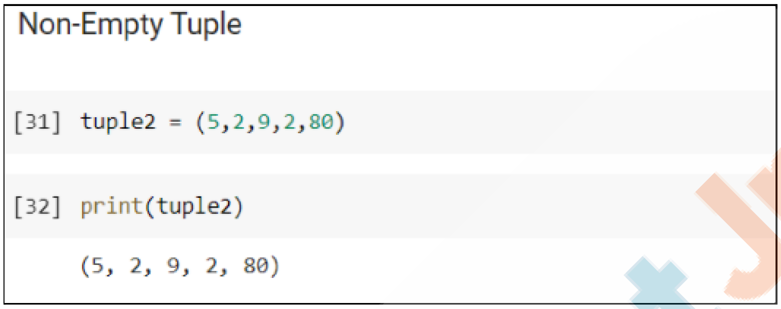
Teacher Starts Slideshow Slide 21 to 26	
Activity details **Following are the WRAP-UP session deliverables:** ● Appreciate the student. ● Revise the current class activities. ● Discuss the quizzes.	
WRAP-UP QUIZ Click on In-Class Quiz	
Continue WRAP-UP Session **Slide 27 to 32**	
Activity Details **Following are the session deliverables:** ● Explain the facts and trivia ● Next class challenge ● Project for the day ● Additional Activity (Optional)	
FEEDBACK ● **Appreciate and compliment the student for trying to learn a difficult concept.** ● **Get to know how they are feeling after the session.** ● **Review and check their understanding.**	
Teacher Action	**Student Action**
You get Hats off for your excellent work!	*Make sure you have given at least 2 Hats Off during the class for:*

In the next class, you will learn how to use nested for loops in Python. You will create star and number patterns using for loops in Python.	Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections	
Teacher Clicks ✖ End Class	
ADDITIONAL ACTIVITIES	
Additional Activities *Encourage the student to extend the student to explore the **tuple** sequence data type.* *Help the student to:* ● **Define a tuple**: We define a tuple using parenthesis(curved brackets) **Empty tuple**: Empty tuple is just a parenthesis, () without any elements. **Non-empty tuple**: Non-empty tuple has elements inside the parentheses separated by a comma.	



Empty Tuple

```
[30] tuple1 = ()
```



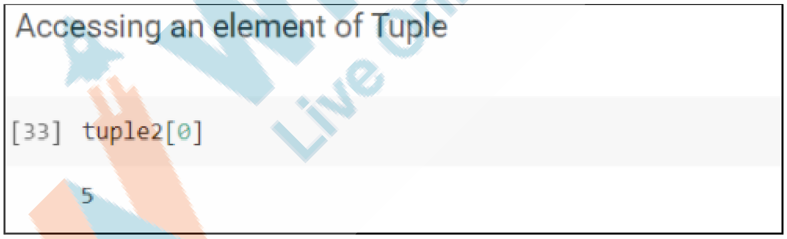
Non-Empty Tuple

```
[31] tuple2 = (5,2,9,2,80)

[32] print(tuple2)

     (5, 2, 9, 2, 80)
```

- **Access an element of the tuple using the index**:

  We can use the elements of the tuple using indexes in the same way as we do for lists.

  We provide the index number inside the square brackets to access elements.



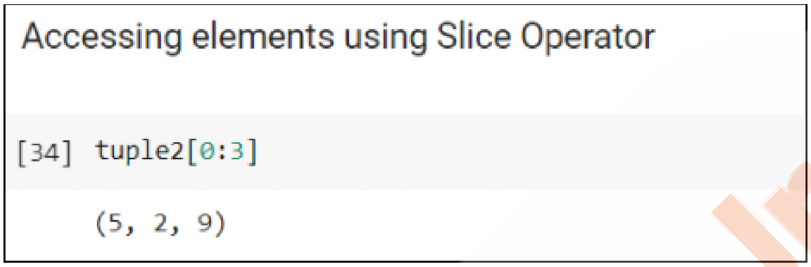
Accessing an element of Tuple

```
[33] tuple2[0]

     5
```

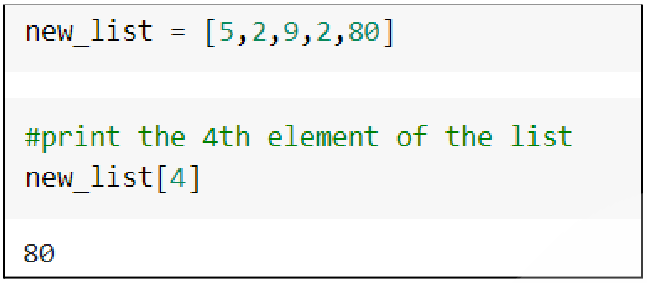
- **Access the elements using the slice operator**:

  We can access the elements of the tuple using the slice operator too in the same way as we do for lists.

We provide the start and stop index separated by the colon inside the square brackets to access elements for a range of indexes.	


Accessing elements using Slice Operator
`[34] tuple2[0:3]`
`(5, 2, 9)`	

These are a few of the properties of the tuple data type in Python. Don't you think what we discussed till now is similar to lists? Then why do you think we need a different data type if we are doing the same things? Well yeah, **tuples are similar to list data types in every sense, except** the **value of a particular element cannot be reassigned**! Let's understand this by an example. Take a list variable, **new_list** with few elements in it. Now, print the 5th element using the index value 4.	 *ESR: Yes* *ESR: There must be some difference.*

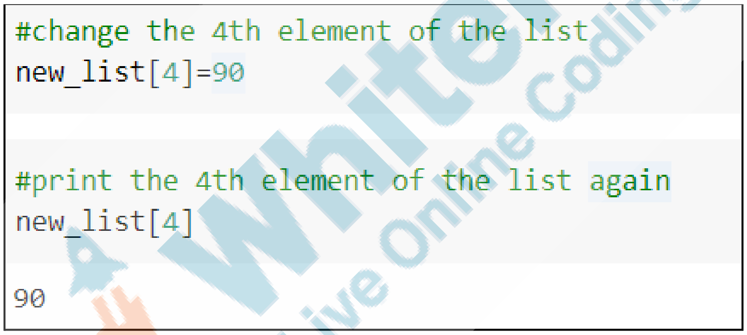


```
new_list = [5,2,9,2,80]

#print the 4th element of the list
new_list[4]

80
```

Now, let’s update the element at the 5th position.

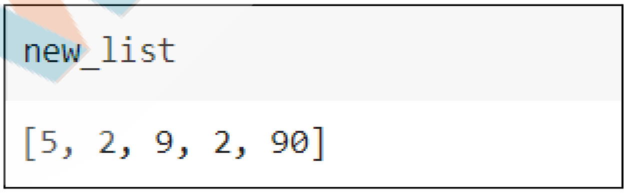
We can do this by accessing the element using the index position and assigning a new value, and again print the 5th element.

We can see that the element at 5th gives the new value.



```
#change the 4th element of the list
new_list[4]=90

#print the 4th element of the list again
new_list[4]

90
```

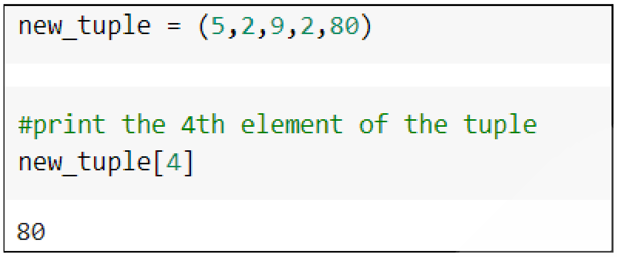
We can verify by printing the whole list again. We can see the 5th element in the list is updated.



```
new_list

[5, 2, 9, 2, 90]
```

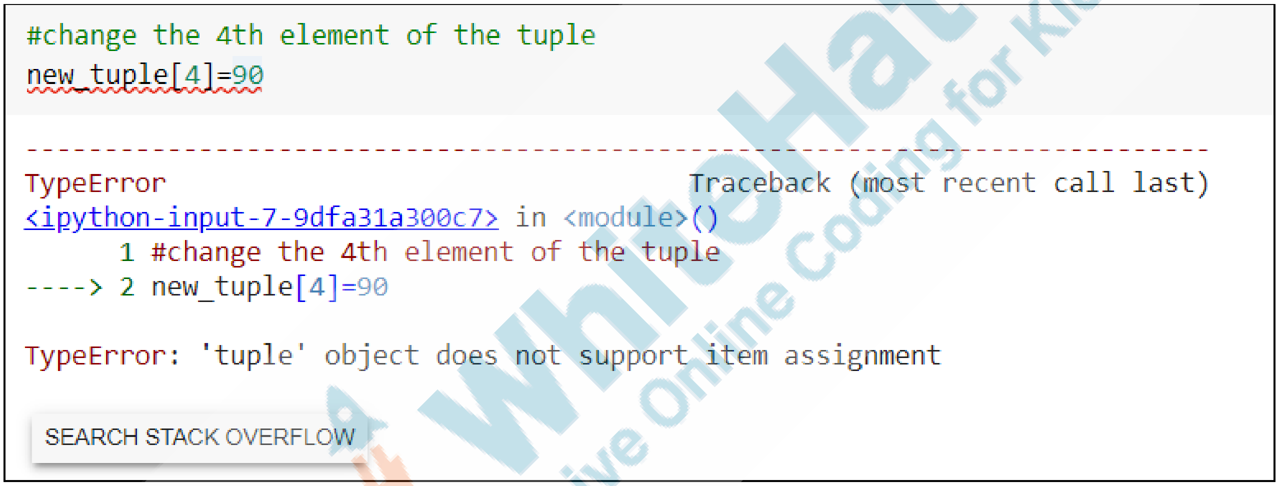
Now, let’s take a tuple variable, **new_tuple** with few elements in it.

Print the 5th element using the index value 4.



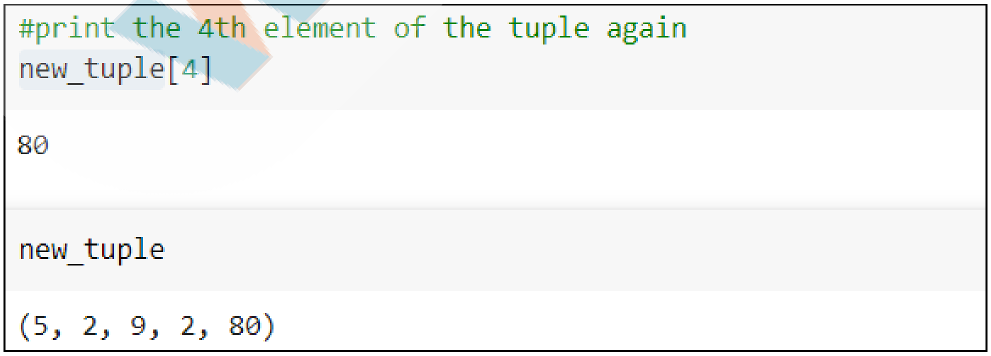
```
new_tuple = (5,2,9,2,80)

#print the 4th element of the tuple
new_tuple[4]

80
```

But if you try to update the 5th element in the tuple, we will get an error.



```
#change the 4th element of the tuple
new_tuple[4]=90

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-7-9dfa31a300c7> in <module>()
      1 #change the 4th element of the tuple
----> 2 new_tuple[4]=90

TypeError: 'tuple' object does not support item assignment
```

If we try to print the tuple again, we will get the same element at the 5th position.



```
#print the 4th element of the tuple again
new_tuple[4]

80

new_tuple

(5, 2, 9, 2, 80)
```

We understood how tuples are similar and different from lists in Python.	

ACTIVITY LINKS		
Activity Name	**Description**	**Link**
Teacher Activity 1	Google Colab Notebook	Google Colaboratory
Teacher Activity 2	Reference Code	https://colab.research.google.com/drive/1mT1H9-j5fkNuEdX9D1kD1U2H1_rJ5-PU?usp=sharing
Student Activity 1	Google Colab Notebook	Google Colaboratory
Student Activity 2	List Indexing Reference	https://s3-whjr-v2-prod-bucket.whjr.online/ca302d6f-0ddb-410d-931d-ec2bf13f9946.png
Teacher Reference 1	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/724e4b07-7c21-4044-84be-169c5704b2cf.pdf
Teacher Reference 2	Project Solution	https://colab.research.google.com/drive/1EgFJOrSQCbK70F-LFo5SwbAHSGx_rmng?usp=sharing
Teacher Reference 3	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/b32d1abe-27fc-4adb-8380-1ad5d447357c.html
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/688696ff-51f7-43e6-a863-b8b4440a1cde.pdf