ShiP.py

Learn to Py while Shelter-in-Place

L4A: Lists and Tuples

A volunteering educational initiative during COVID-19

ShiP Crew

JD

Teddy

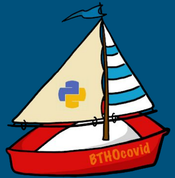
Chinmay

Pratik

Siddharth

Umang

Waseem



A volunteering educational initiative during COVID-19

Topics

PHASE I: Foundations

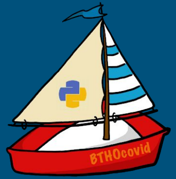
All times are in CDT (GMT-5)

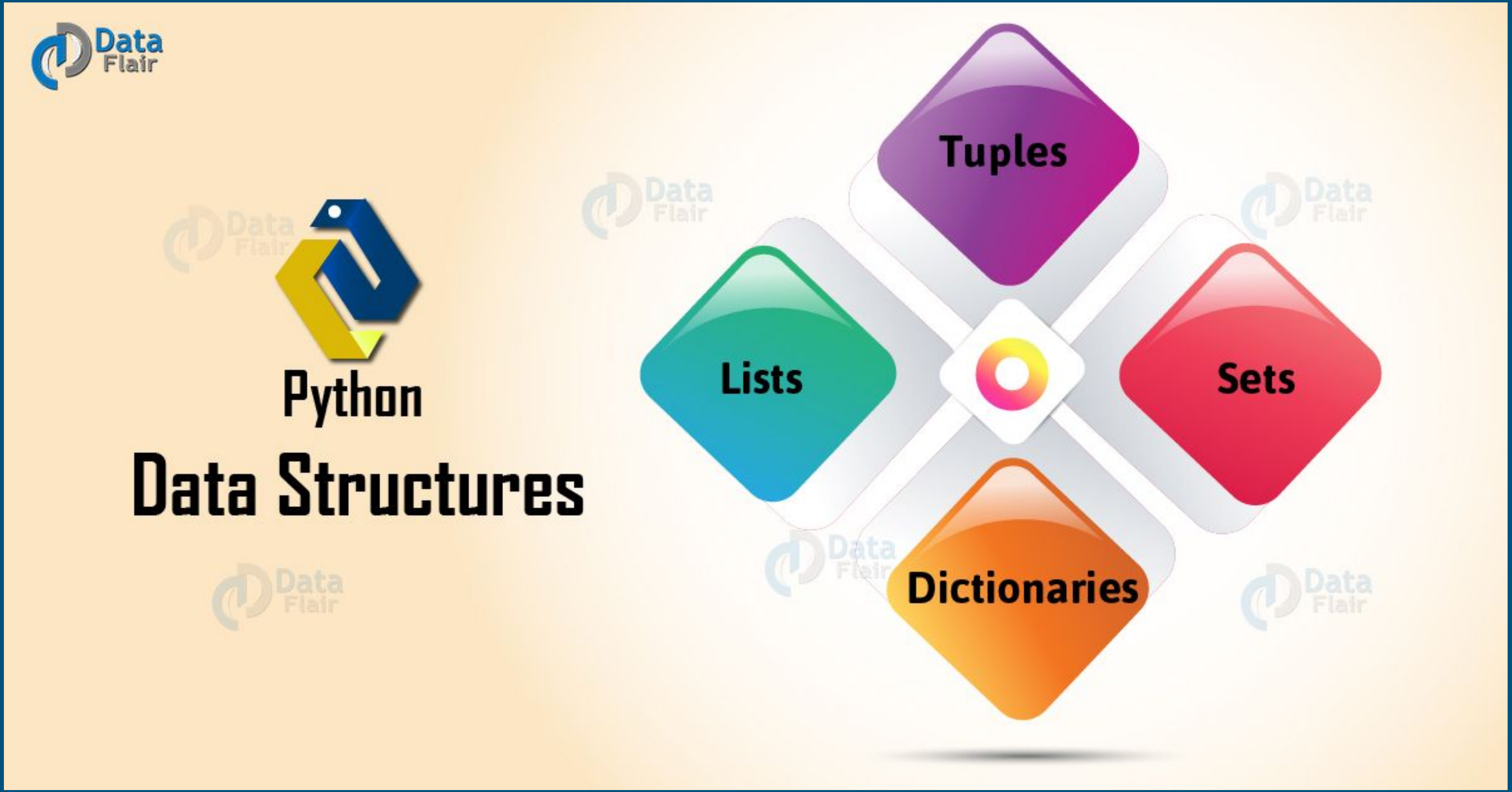
1.	Variables, Expressions, Simple I/O	Sat, April 18 (11 am-12 noon)	☀
2.	Boolean Decisions (branching)	Wed, April 22 (9 pm-10 pm)	🌙
3.	Repetitions (loops)	Sat, April 25 (11 am-12 noon)	☀
4A.	Collective Data Structures (Lists and Tuples)	Wed, April 29 (9 pm-10 pm)	🌙
4B.	CDS (Dictionaries and Sets)	Sat, May 02 (11 am-12 noon)	☀
5.	Functions	Wed, May 06 (9 pm-10 pm)	🌙
6.	File I/O	Sat, May 09 (11 am-12 noon)	☀

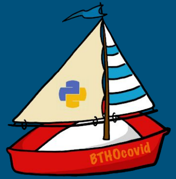
Lecture 4A

AGENDA

- Lists
- Nested Lists
- Indexing and Slicing
- List methods
- Tuples



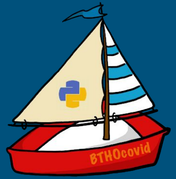

Data Flair
Python
Data Structures
Tuples
Lists
Sets
Dictionaries


BTHOcovid

A collection of items

- I have a collection games
- I want to keep them organized
- To do so, I store them in numbered boxes
- I can retrieve a particular game by knowing the box number

GTA5	DOTA	CS-GO	AOE	AOEII	COD	FIFA20	NFS	SIMS4
1	2	3	4	5	6	7	8	9

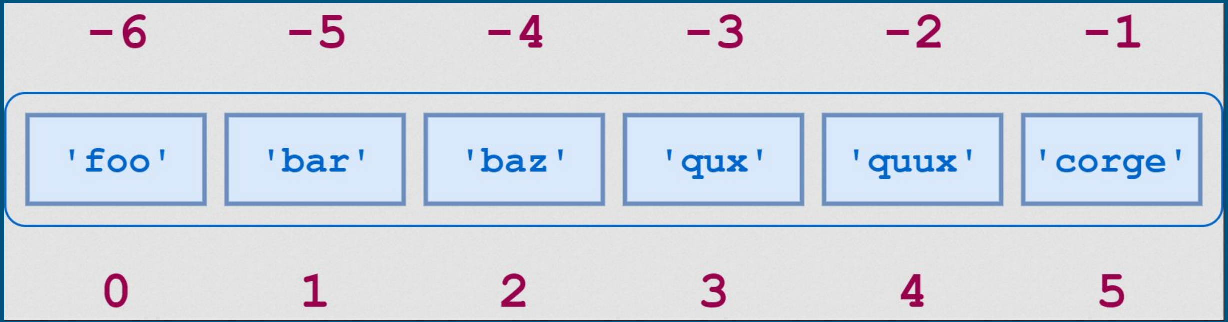


I have a structure to store my games, similarly python has a data structures to keep the stored data organized

Lists

A list is a sequence of values.

	‘H’	‘o’	‘w’	‘d’	‘y’
Indexing starts with 0	0	1	2	3	4

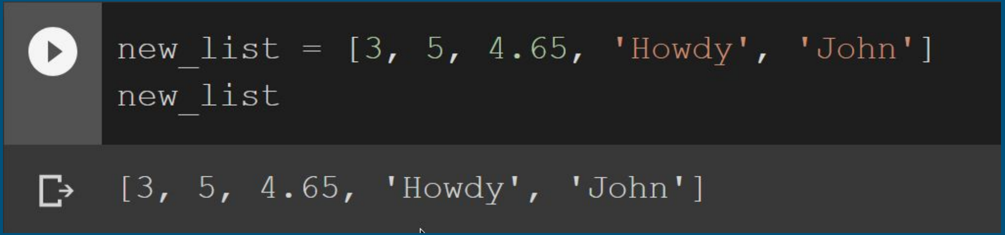


Negative Index

Positive Index

Creating a list in python

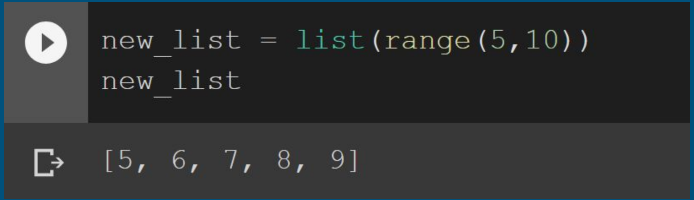
Using a square bracket notation [ ]



```
new_list = [3, 5, 4.65, 'Howdy', 'John']
new_list
```

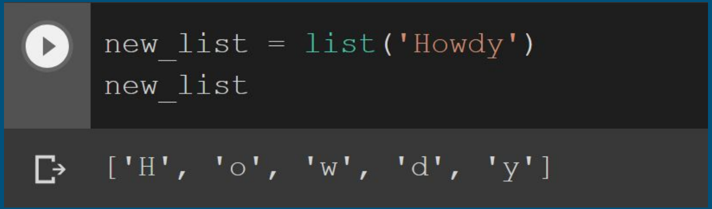
```
[3, 5, 4.65, 'Howdy', 'John']
```

Using `list()` function on an iterable



```
new_list = list(range(5,10))
new_list
```

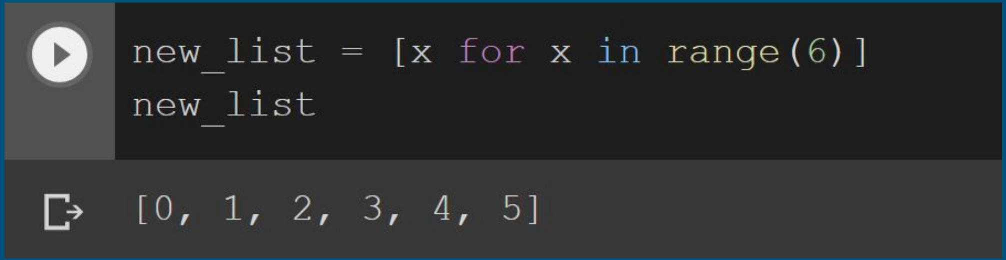
```
[5, 6, 7, 8, 9]
```



```
new_list = list('Howdy')
new_list
```

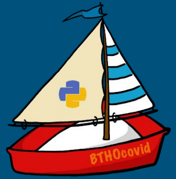
```
['H', 'o', 'w', 'd', 'y']
```

Using list comprehension - discussed later in lecture L4B



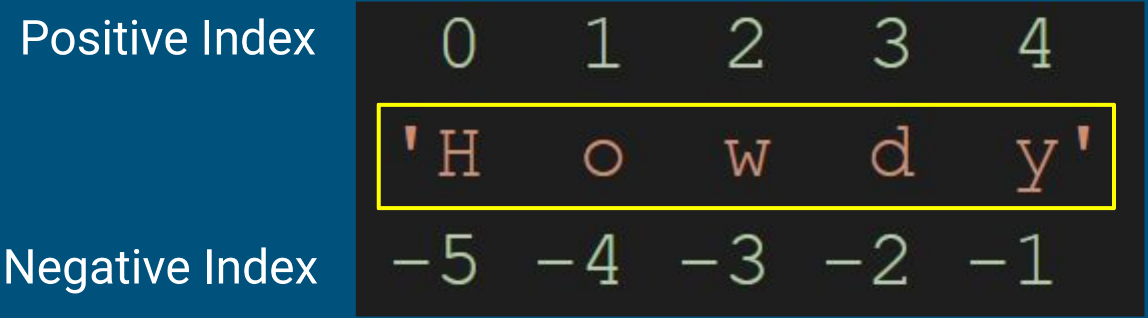
```
new_list = [x for x in range(6)]
new_list
```

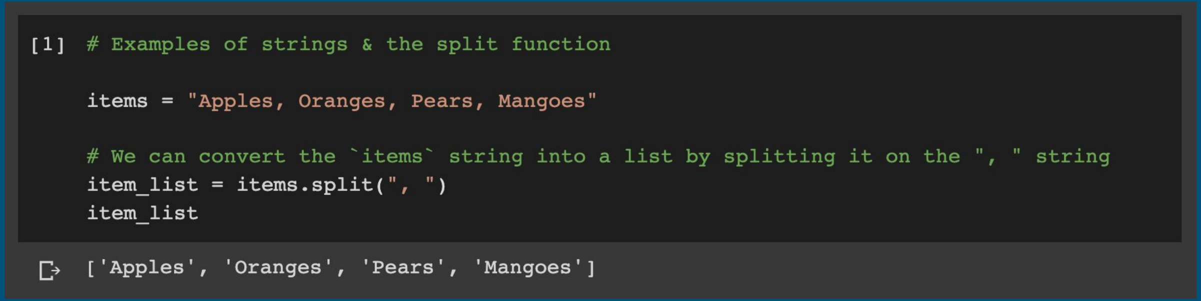
```
[0, 1, 2, 3, 4, 5]
```



Strings and Lists

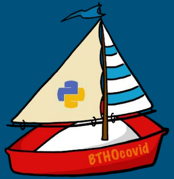
Strings can be visualized as a collection of characters





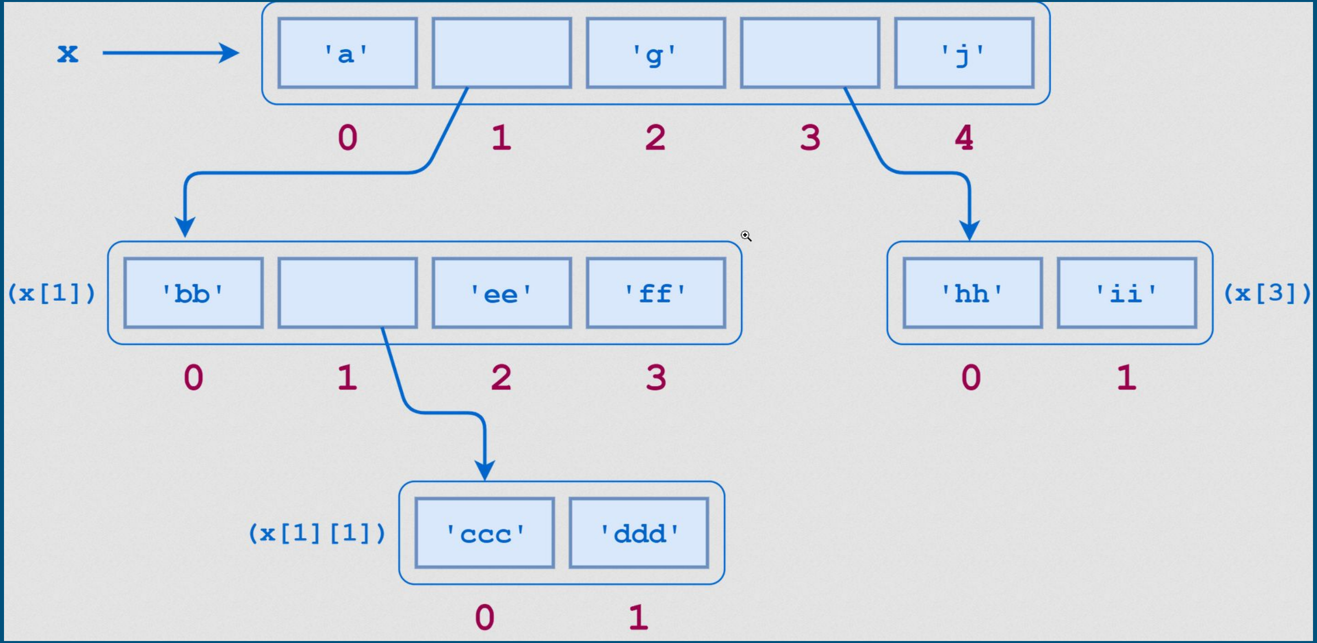
```
[1] # Examples of strings & the split function

    items = "Apples, Oranges, Pears, Mangoes"

    # We can convert the `items` string into a list by splitting it on the ", " string
    item_list = items.split(", ")
    item_list
```

```
['Apples', 'Oranges', 'Pears', 'Mangoes']
```



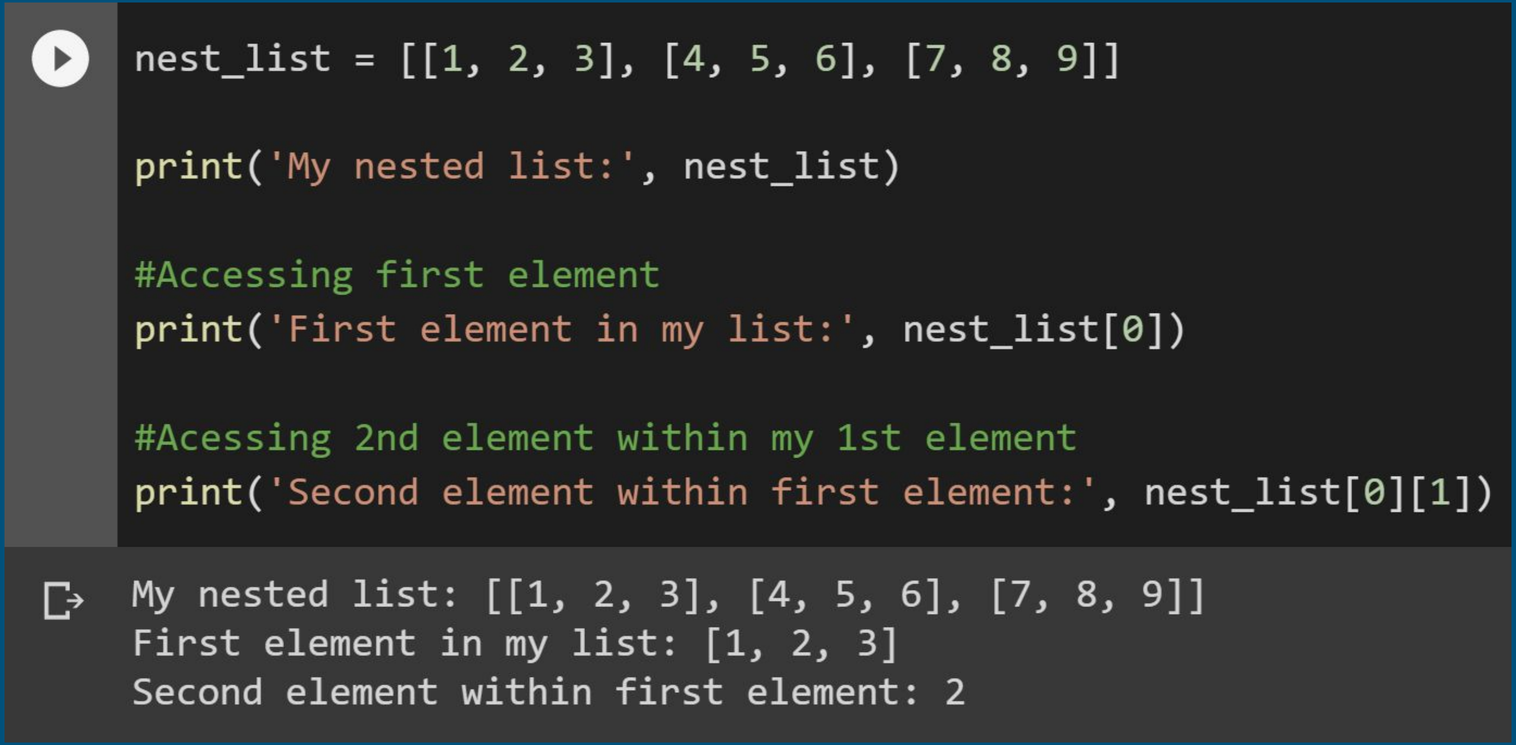
Nested lists

Each element of a list could be a list itself



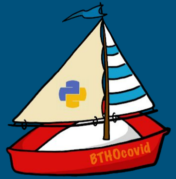
source

Nested list: Example

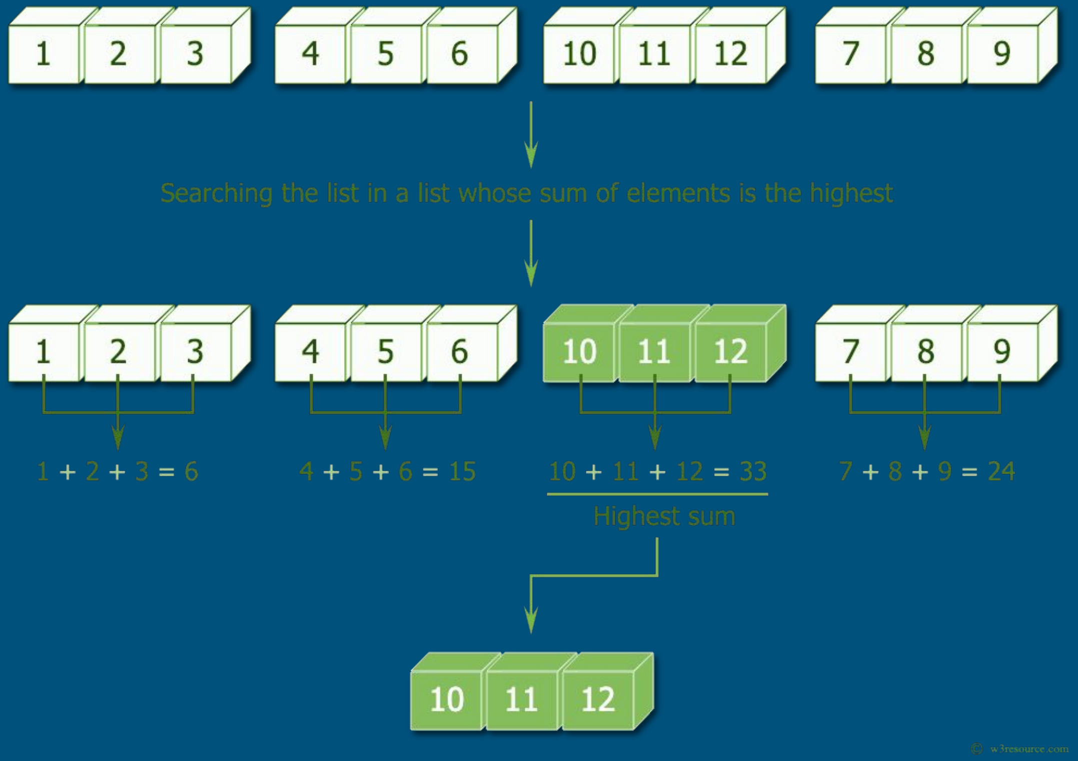


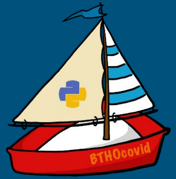
```
nest_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

print('My nested list:', nest_list)

#Accessing first element
print('First element in my list:', nest_list[0])

#Acessing 2nd element within my 1st element
print('Second element within first element:', nest_list[0][1])
```

```
My nested list: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
First element in my list: [1, 2, 3]
Second element within first element: 2
```



Nested list: Application





Indexing

A technique used select a particular element

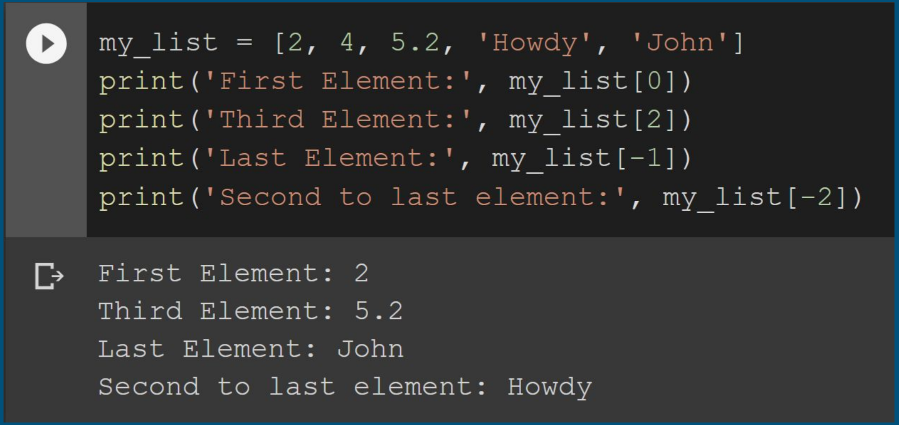
Name of the list

```
variable_name[index]
```

Index number
(either positive or negative)

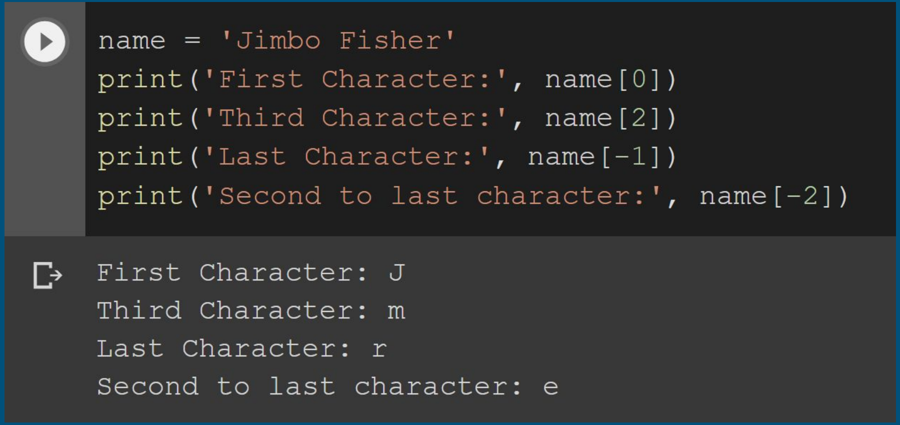
Indexing for a string works in a similar way

Examples of Indexing



```python
my_list = [2, 4, 5.2, 'Howdy', 'John']
print('First Element:', my_list[0])
print('Third Element:', my_list[2])
print('Last Element:', my_list[-1])
print('Second to last element:', my_list[-2])
```

```
First Element: 2
Third Element: 5.2
Last Element: John
Second to last element: Howdy
```



```python
name = 'Jimbo Fisher'
print('First Character:', name[0])
print('Third Character:', name[2])
print('Last Character:', name[-1])
print('Second to last character:', name[-2])
```

```
First Character: J
Third Character: m
Last Character: r
Second to last character: e
```

Slicing

A technique used to select a part of a the collection

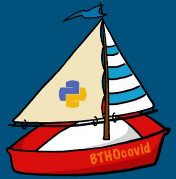
could be a list, tuple or a string

Stop index
Default = End of collection

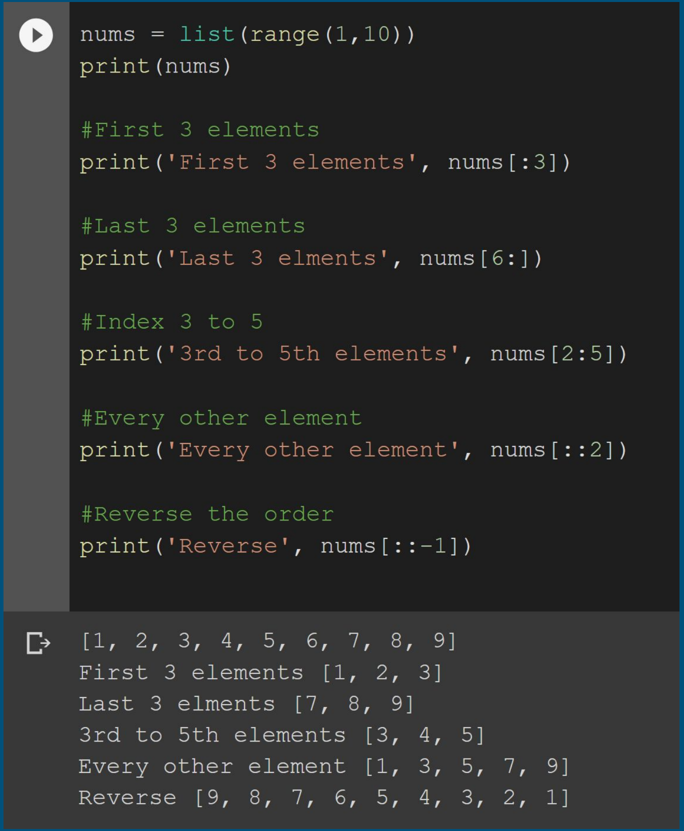
```
variable_name[start: stop : step]
```

Starting index
Default=0

Step size to count
Default=1

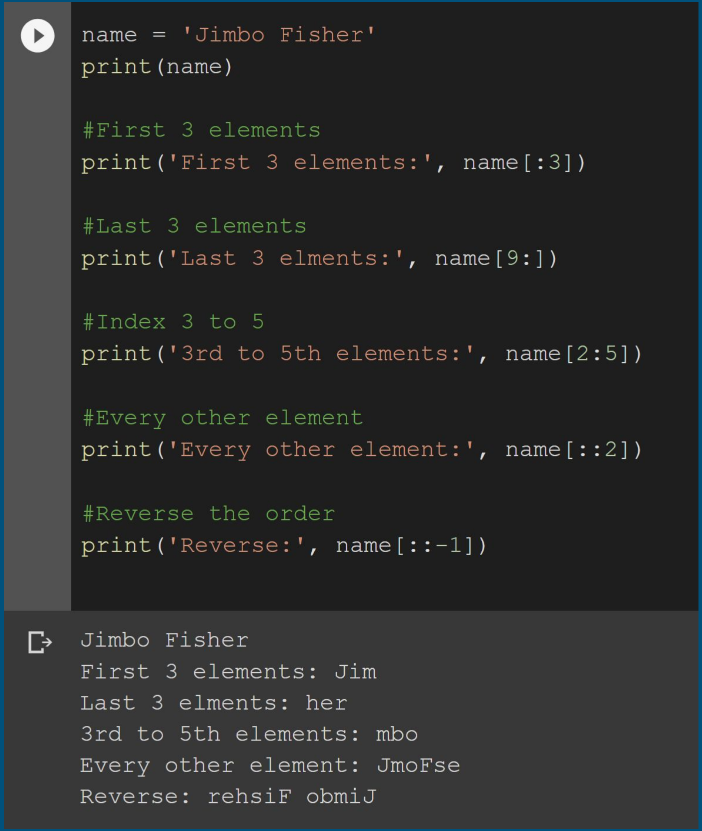


Examples of Slicing



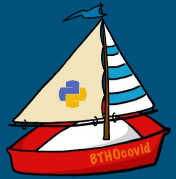
```
nums = list(range(1,10))
print(nums)

#First 3 elements
print('First 3 elements', nums[:3])

#Last 3 elements
print('Last 3 elments', nums[6:])

#Index 3 to 5
print('3rd to 5th elements', nums[2:5])

#Every other element
print('Every other element', nums[::2])

#Reverse the order
print('Reverse', nums[::-1])
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
First 3 elements [1, 2, 3]
Last 3 elments [7, 8, 9]
3rd to 5th elements [3, 4, 5]
Every other element [1, 3, 5, 7, 9]
Reverse [9, 8, 7, 6, 5, 4, 3, 2, 1]
```



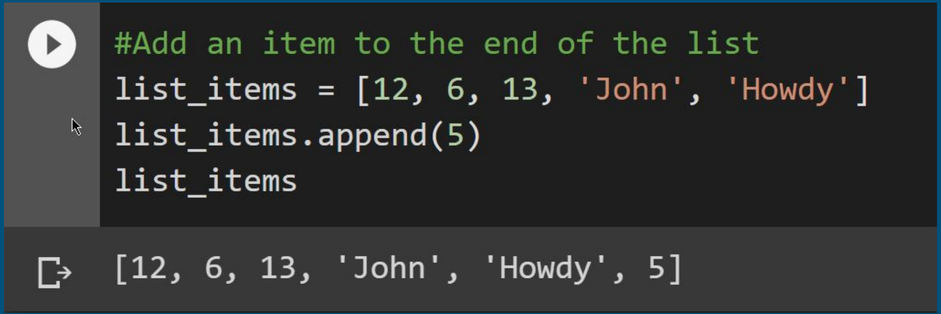
```
name = 'Jimbo Fisher'
print(name)

#First 3 elements
print('First 3 elements:', name[:3])

#Last 3 elements
print('Last 3 elments:', name[9:])

#Index 3 to 5
print('3rd to 5th elements:', name[2:5])

#Every other element
print('Every other element:', name[::2])

#Reverse the order
print('Reverse:', name[::-1])
```

```
Jimbo Fisher
First 3 elements: Jim
Last 3 elments: her
3rd to 5th elements: mbo
Every other element: JmoFse
Reverse: rehsiF obmiJ
```



Common list methods

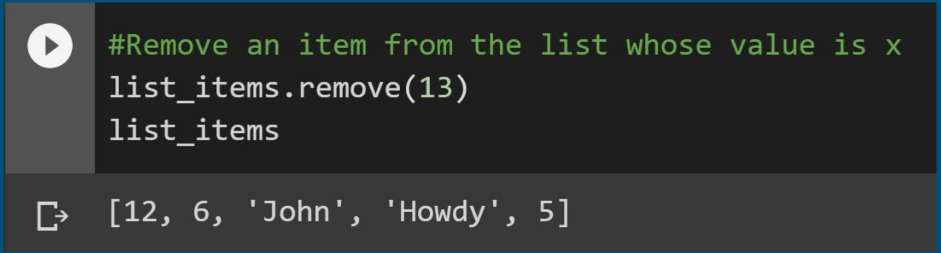
append( )



```
#Add an item to the end of the list
list_items = [12, 6, 13, 'John', 'Howdy']
list_items.append(5)
list_items
```

```
[12, 6, 13, 'John', 'Howdy', 5]
```

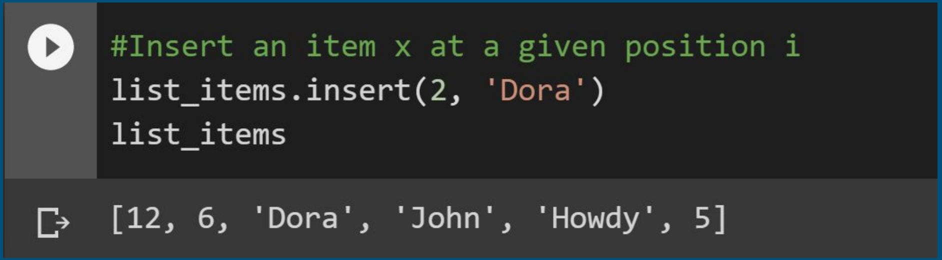
remove( )



```
#Remove an item from the list whose value is x
list_items.remove(13)
list_items
```

```
[12, 6, 'John', 'Howdy', 5]
```

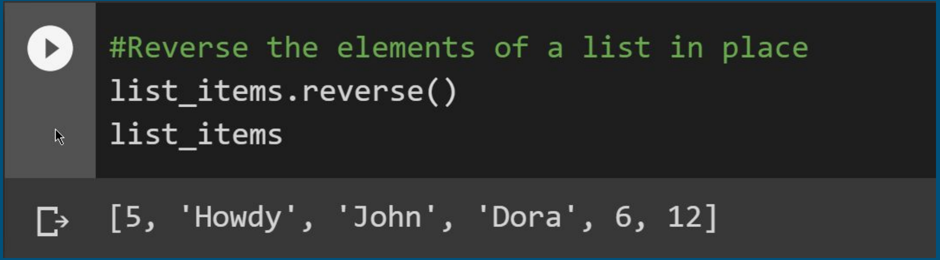
insert( )



```
#Insert an item x at a given position i
list_items.insert(2, 'Dora')
list_items
```

```
[12, 6, 'Dora', 'John', 'Howdy', 5]
```

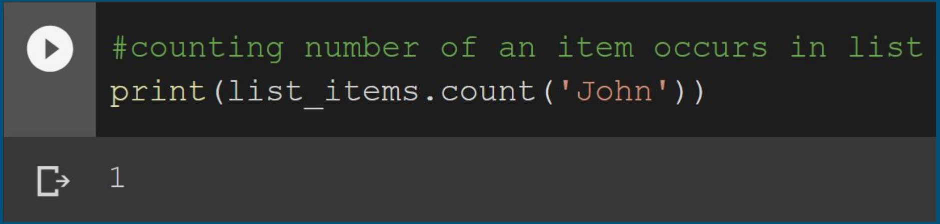
reverse( )



```
#Reverse the elements of a list in place
list_items.reverse()
list_items
```

```
[5, 'Howdy', 'John', 'Dora', 6, 12]
```

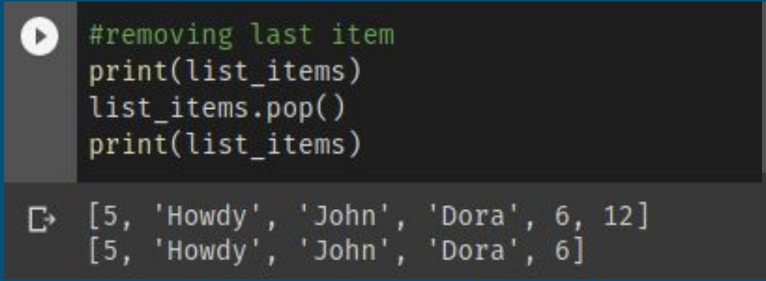
count( )



```
#counting number of an item occurs in list
print(list_items.count('John'))
```

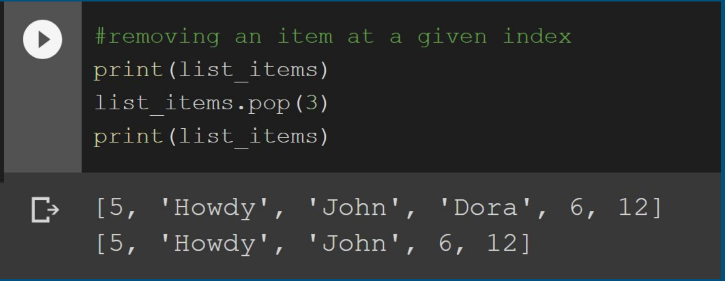
```
1
```

pop( )



```
#removing last item
print(list_items)
list_items.pop()
print(list_items)
```

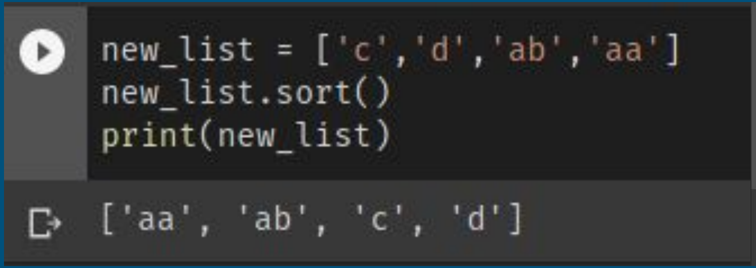
```
[5, 'Howdy', 'John', 'Dora', 6, 12]
[5, 'Howdy', 'John', 'Dora', 6]
```



```
#removing an item at a given index
print(list_items)
list_items.pop(3)
print(list_items)
```

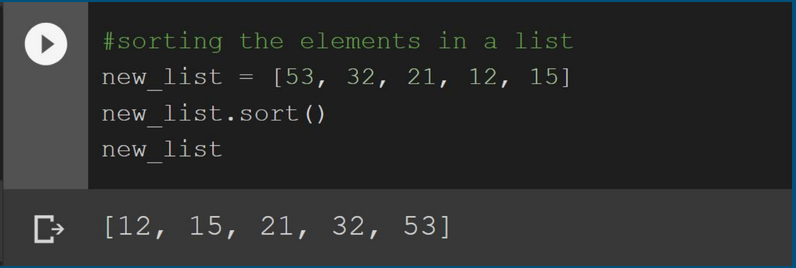
```
[5, 'Howdy', 'John', 'Dora', 6, 12]
[5, 'Howdy', 'John', 6, 12]
```

sort( )



```
new_list = ['c','d','ab','aa']
new_list.sort()
print(new_list)
```

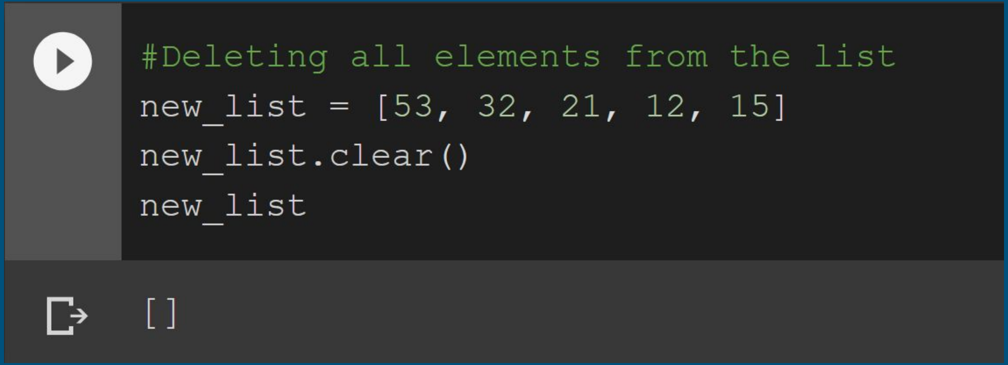
```
['aa', 'ab', 'c', 'd']
```



```
#sorting the elements in a list
new_list = [53, 32, 21, 12, 15]
new_list.sort()
new_list
```

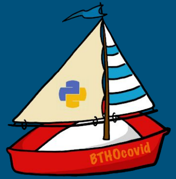
```
[12, 15, 21, 32, 53]
```

clear( )



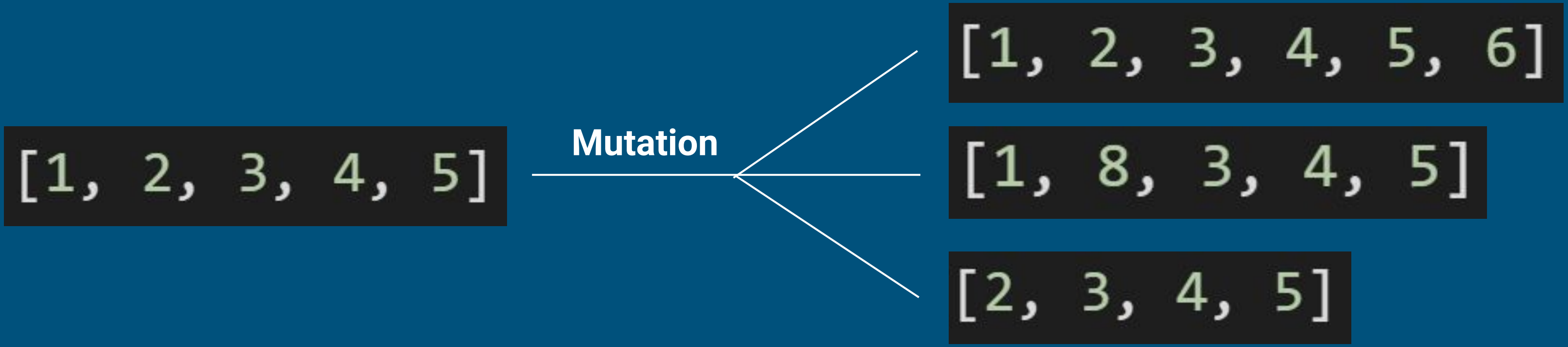
```
#Deleting all elements from the list
new_list = [53, 32, 21, 12, 15]
new_list.clear()
new_list
```

```
[]
```



For an exhaustive discussion, check out python's documentation
https://docs.python.org/3/tutorial/datastructures.html

Properties of lists

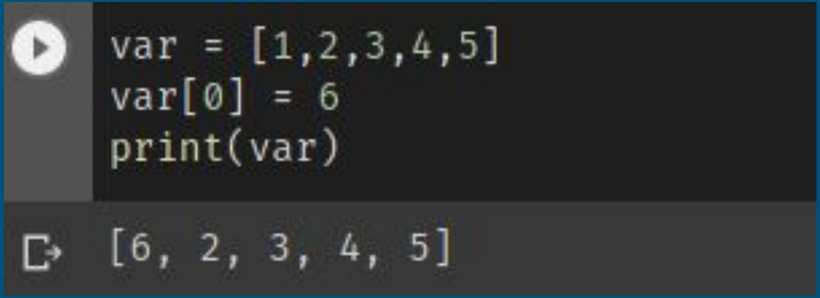


```
[1, 2, 3, 4, 5]
```

```
[1, 2, 3, 4, 5, 6]
[1, 8, 3, 4, 5]
[2, 3, 4, 5]
```

- Lists are **mutable** - we can change elements after creating it
- We can add and remove elements from a list
- Lists are dynamic - size of list changes as we add or remove elements

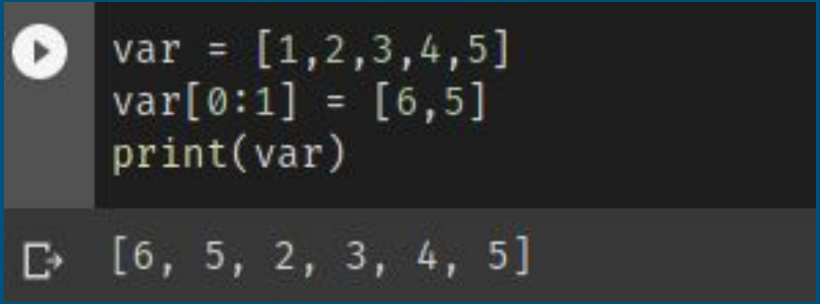
Changing value of the first element



```
var = [1,2,3,4,5]
var[0] = 6
print(var)
```

```
[6, 2, 3, 4, 5]
```

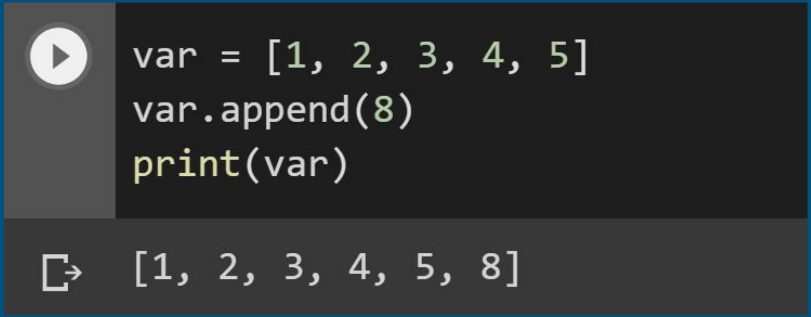
Changing value of the first two elements



```
var = [1,2,3,4,5]
var[0:1] = [6,5]
print(var)
```

```
[6, 5, 2, 3, 4, 5]
```

Adding an element to the list

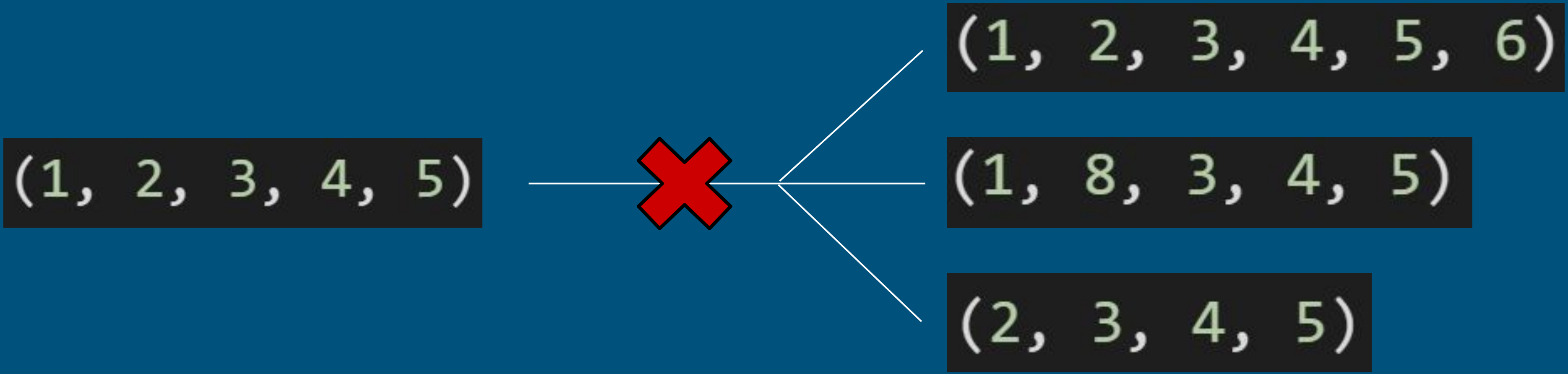


```
var = [1, 2, 3, 4, 5]
var.append(8)
print(var)
```

```
[1, 2, 3, 4, 5, 8]
```

Tuples

Same as lists (a collection of objects) but **immutable**

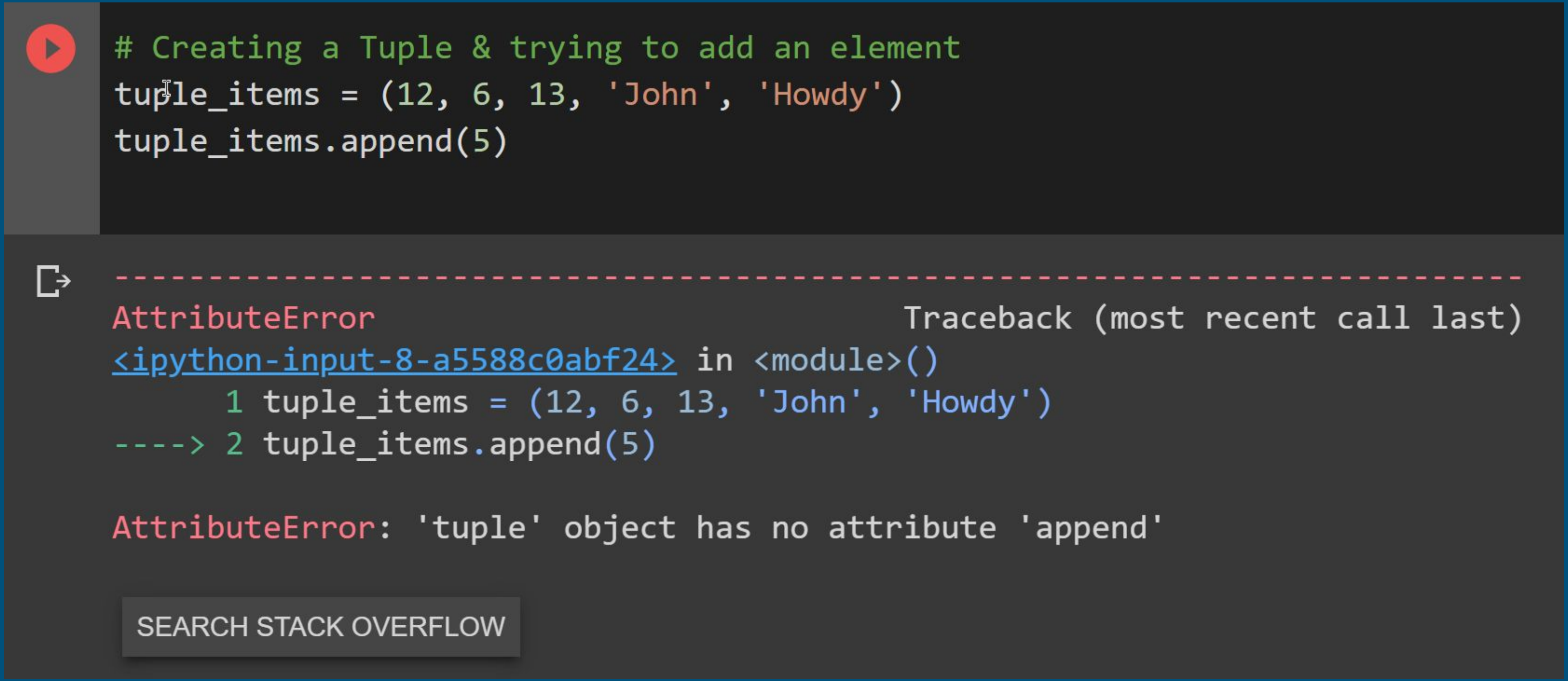


```
(1, 2, 3, 4, 5)
```

```
(1, 2, 3, 4, 5, 6)
```

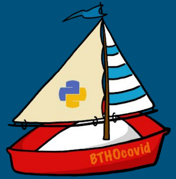
```
(1, 8, 3, 4, 5)
```

```
(2, 3, 4, 5)
```



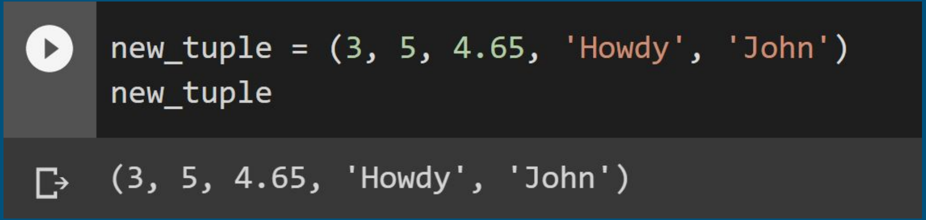
```
# Creating a Tuple & trying to add an element
tuple_items = (12, 6, 13, 'John', 'Howdy')
tuple_items.append(5)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-8-a5588c0abf24> in <module>()
      1 tuple_items = (12, 6, 13, 'John', 'Howdy')
----> 2 tuple_items.append(5)

AttributeError: 'tuple' object has no attribute 'append'
```



Creating a tuple

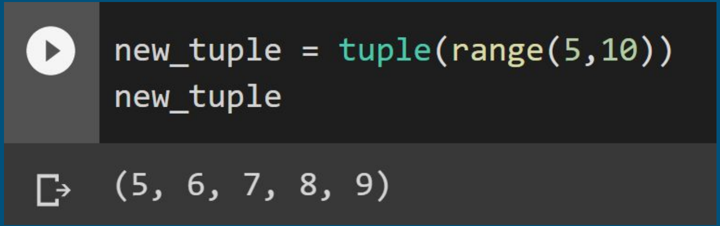
Using ( )



```
new_tuple = (3, 5, 4.65, 'Howdy', 'John')
new_tuple
```

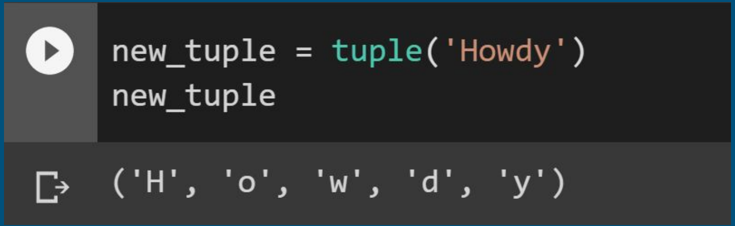
```
(3, 5, 4.65, 'Howdy', 'John')
```

Using `tuple()` function



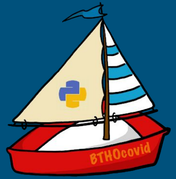
```
new_tuple = tuple(range(5,10))
new_tuple
```

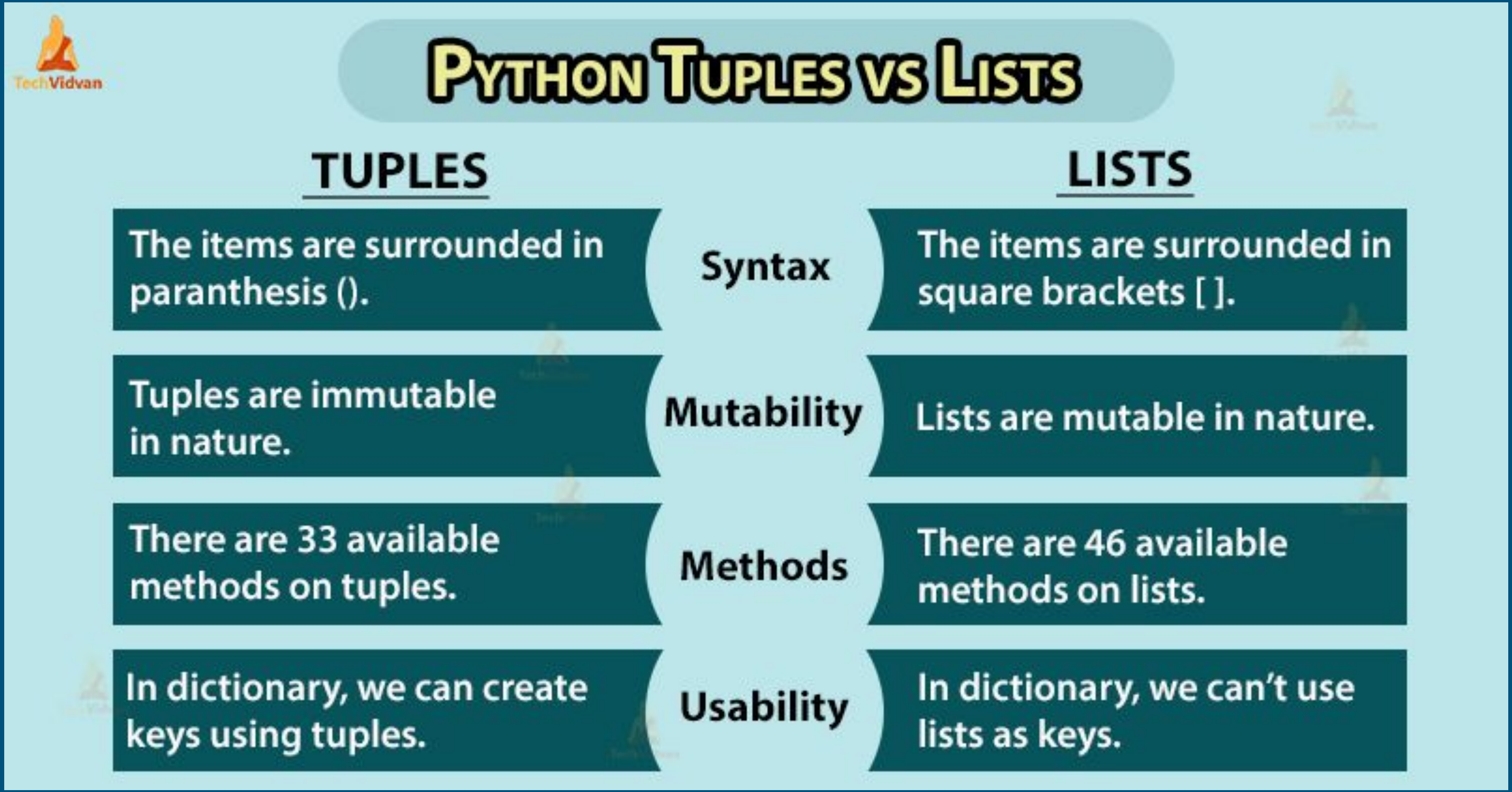
```
(5, 6, 7, 8, 9)
```



```
new_tuple = tuple('Howdy')
new_tuple
```

```
('H', 'o', 'w', 'd', 'y')
```





TechVidvan

PYTHON TUPLES VS LISTS

TUPLES		LISTS
The items are surrounded in paranthesis ().	Syntax	The items are surrounded in square brackets [ ].
Tuples are immutable in nature.	Mutability	Lists are mutable in nature.
There are 33 available methods on tuples.	Methods	There are 46 available methods on lists.
In dictionary, we can create keys using tuples.	Usability	In dictionary, we can't use lists as keys.

Next Lecture

Dictionaries and Sets

Sat, May 02 (11 am-12 noon CDT)