Welcome to Basic Python Workshop day 2



Monirul Islam
Executive at Nextzen Limited
B.Sc in CSE University of Global Village





Youtube.com/@techprosbd



Python Collection Type Array

- Python doesn't have a built-in array data structure like other programming languages. However, it has several data structures that can be used to mimic arrays, such as:
- List: A list is an ordered collection of elements, which can be of any data type, including other lists. Lists are mutable, meaning their elements can be changed.
- Tuple: A tuple is an ordered and immutable collection of elements, similar to a list but with less functionality. Once created, the elements of a tuple cannot be changed.
- Set: A set is an unordered collection of unique elements, meaning there can be no duplicate elements in a set. Sets are mutable, meaning their elements can be added, removed or changed.
- Dictionary: A dictionary is an unordered collection of key-value pairs, where each key is unique and used to access its associated value. Dictionaries are mutable, meaning the key value pairs can be added, removed, or modified.
- Each of these data structures has its own use case and characteristics, and the characteristics which to use depends on the specific needs of a particular task.

Sort Syntax of List, Tuple, Set & Dictionary

• Here is an example of the syntax for lists, tuples, sets, and dictionaries in Python: # Creating a list fruits = ['apple', 'banana', 'cherry'] # Creating a tuple fruits = ('apple', 'banana', 'cherry') # Creating a set fruits = {'apple', 'banana', 'cherry'} # Creating a dictionary person = {'name': 'John Doe', 'age': 30, 'city': 'New York'}

Python Collection Type Array

- Python Collections (Arrays)
- There are four collection data types in the Python programming language:
- List is a collection which is ordered and changeable. Allows duplicate members.
- <u>Tuple</u> is a collection which is ordered and unchangeable. Allows duplicate members.
- <u>Set</u> is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
- <u>Dictionary</u> is a collection which is ordered** and changeable. No duplicate members.

Details of List Method:

```
append() Adds an element at the end of the list
fruits = ['apple', 'banana', 'cherry']
fruits.append("orange")
clear() Removes all the elements from the list
fruits = ['apple', 'banana', 'cherry', 'orange']
fruits.clear()
copy() Returns a copy of the list
fruits = ['apple', 'banana', 'cherry', 'orange']
x = fruits.copy()
count() Returns the number of elements with the specified value
fruits = ['apple', 'banana', 'cherry']
x = fruits.count("cherry")
print(len(fruits))
print(type(fruits))
```

Details of List Method:

```
extend() Add the elements of a list (or any iterable), to the end of the current list fruits = ['apple', 'banana', 'cherry'] cars = ['Ford', 'BMW', 'Volvo'] fruits.extend(cars)

index() Returns the index of the first element with the specified value fruits = ['apple', 'banana', 'cherry']
```

insert() Adds an element at the specified position

```
fruits = ['apple', 'banana', 'cherry']
fruits.insert(1, "orange")
```

pop() Removes the element at the specified position

```
fruits = ['apple', 'banana', 'cherry']
fruits.pop(1)
```

x = fruits.index("cherry")



Details of List Method:

remove() Removes the item with the specified value

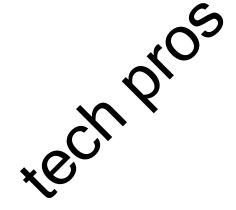
```
fruits = ['apple', 'banana', 'cherry']
fruits.remove("banana")
```

reverse() Reverses the order of the list

```
fruits = ['apple', 'banana', 'cherry']
fruits.reverse()
```

sort()Sorts the list

```
cars = ['Ford', 'BMW', 'Volvo',"pc"]
cars.sort()
print(cars[0:2])
```



Access List Item using For and While Loop:

Loop Through a List

```
thislist ="apple", "banana", "cherry"] for x in thislist: print(x)
```

Loop Through the Index Numbers

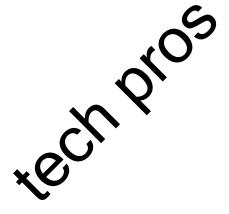
```
thislist = ["apple", "banana", "cherry"]
for i in range(len(thislist)):
    print(thislist[i])
```

While Loop

```
thislist = ["apple", "banana",
"cherry", "cpu"]
x = 0
while x<(len(thislist)):
    print(thislist[x])
    x+=1</pre>
```

PYTHON LIST METHODS

INPUT	METHOD	OUTPUT
[4, 3, 5]	sort()	[3, 4, 5]
[1, 2, 3]	clear()	[]
[1, 2, 3]	append(4)	[1, 2, 3, 4]
[1, 1, 1, 2]	count(1)	3
[1, 2, 3]	extend([4])	[1, 2, 3, 4]
[1, 2, 3]	insert(3, 4)	[1, 2, 3, 4]
[1, 2, 3]	index(2)	1
[1, 2, 3]	remove(2)	[1, 3]
[1, 2, 3]	pop(2)	[1, 2]
[1, 2, 3]	reverse()	[3, 2, 1]
[1, 2, 3]	copy()	[1, 2, 3]
[1, 2, 3]	len()	3
[1, 2, 3]	min()	1
[1, 2, 3]	max()	3



Details of Tuple Method:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
If tuple have a one item then use comma,
thistuple = ("apple",)
print(type(thistuple))
count()Returns the number of times a specified value occurs in a tuple
thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
x = thistuple.count(5)
print(x)
```

Details of Tuple Method:

index()Searches the tuple for a specified value and returns the position of where it was found

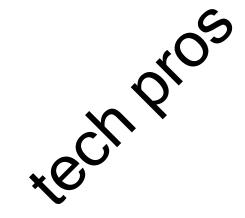
```
thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
x = thistuple.index(8)
print(x)
```

Access Tuple time using index:

print(thistuple[3])

Check if Item Exists

```
thistuple = ("apple", "banana", "cherry") if "apple" in thistuple: print("Yes, 'apple' is in the fruits tuple")
```



Access Tuple Item using For & While Loop:

```
For Loop

thistuple = ("apple", "banana", "cherry")

for x in thistuple:
    print(x)

Using index:

thistuple = ("apple", "banana", "cherry")

for i in range(len(thistuple)):
    print(thistuple[i])

while loop

thistuple = ("apple", "banana", "cherry")

i = 0

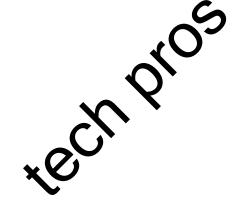
while i < len(thistuple):
    print(thistuple[i])
```

*ech bros

Then how can join a item in tuple?

Convert tuple to list and operate all list method.

```
thislist = ("apple", "banana", "cherry", "cpu")
newlist = list(thislist)
newlist.append("computer")
thislist = tuple(newlist)
print(tuple(thislist))
```



Details Of Set Method:

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

You cannot access items in a set by referring to an index or a key.

```
thisset = {"apple", "banana", "cherry"}
for x in thisset:
  print(x)
```

Add Items

```
thisset = {"apple", "banana", "cherry"}
thisset.add("orange")
print(thisset)
```



Details Of Set Method:

Add Any Iterable

thisset.discard("banana")

print(thisset)

```
thisset = {"apple", "banana", "cherry"}
mylist = ["kiwi", "orange"]
thisset.update(mylist)
print(thisset)

Remove Item
thisset = {"apple", "banana", "cherry"}
thisset.remove("banana")

If the item to remove does not exist, discard() will NOT raise an error.
thisset = {"apple", "banana", "cherry"}
```

Details Of Set Method:

The clear() method empties the set:

```
thisset = {"apple", "banana", "cherry"}
thisset.clear()
print(thisset)
```

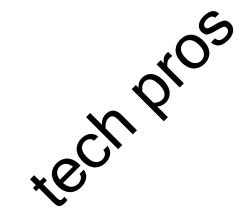
The del keyword will delete the set completely:

```
thisset = {"apple", "banana", "cherry"}
del thisset
print(thisset)
```

The union() method returns a new set with all items from both sets:

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3 = set1.union(set2)
print(set3)
```

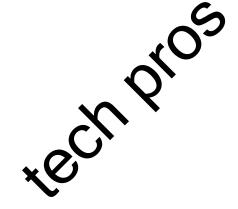
```
mydict = {
  "name": "monirul",
  "age": 24,
  "cgpa": 3.80,
  "dept": "cse",
  "Regular": True
print(mydict)
Print(len(mydict))
Print(type(mydict))
```



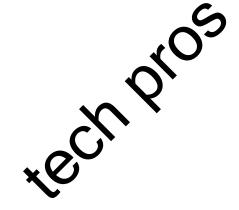
thisdict = {

```
"brand": "Ford",
 "model": "Mustang",
 "year": 1964
x = thisdict["model"]
x = thisdict.get("models") (give no error if not found)
x = thisdict.keys()
x = thisdict.values()
print(x)
if "model" in thisdict:
 print("Yes, 'model' is one of the keys in the thisdict dictionary")
```

Update Dictionary thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 thisdict.update({"year": 2020}) thisdict["color"] = "red" Removing Items thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 thisdict.pop("model") print(thisdict)

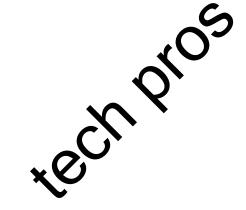


```
The popitem() method removes the last inserted item
thisdict = {
 "brand": "Ford",
 "model": "Mustang",
 "year": 1964
thisdict.popitem()
print(thisdict)
del thisdict["model"]
thisdict.clear()
```

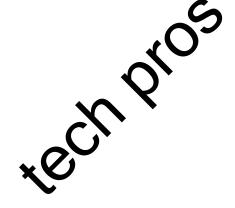


For Loop in Dictionaries:

```
thisdict =
 "brand": "Ford",
 "model": "Mustang",
 "year": 1964
for x, y in thisdict.items():
 print(x, y)
for x in thisdict.keys():
 print(x)
for x in thisdict.values():
 print(x)
```



• Copy a Dictionary
thisdict = {
 "brand": "Ford",
 "model": "Mustang",
 "year": 1964
}
mydict = thisdict.copy()
print(mydict)



Nested Dict:

```
    myfamily = {

  "child1": {
   "name": "Emil",
   "year" : 2004
  "child2" : {
   "name": "Tobias",
   "year": 2007
   "child3": {
   "name": "Linus",
    "year": 2011
```