Python Data Structures (Lavanya.R - 2247249)

The basic Python data structures in Python include list, set, tuples, and dictionary. Each of the data structures is unique in its own way. Data structures are “containers” that organize and group data according to type.

Lists

A list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements.

Access Items (Same for tuples)

Print the second item of the list:	print(thislist[1])
Print the last item of the list:	print(thislist[-1])
A range of indexes by specifying where to start and where to end the range.	print(thislist[2:5])
Returns the items from the beginning to, but NOT including, "kiwi"	print(thislist[:4])
Returns the items from "cherry" to the end	print(thislist[2:])
Returns the items from "orange" (-4) to, but NOT including "mango" (-1)	print(thislist[-4:-1])
Check if "apple" is present in the list	if "apple" in thislist: print("Yes, 'apple' is in the fruits list")

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
```

List Methods

append()	Adds an element at the end of the list	list.append (element)
clear()	Removes all the elements from the list	list.clear()
copy()	Returns a copy of the list	list.copy()
count()	Returns the number of elements with the specified value	List.count(element)
extend()	Add the elements of a list (or any iterable), to the end of the current list	List1.extend-(List2)
index()	Returns the index of the first element with the specified value	List.index(element[,start[,end]])
insert()	Adds an element at the specified position	list.insert(<position, element)
pop()	Removes the element at the specified position	list.pop(-[index])
remove()	Removes the first item with the specified value	list.remove(element)

List Methods (cont)

reverse()	Reverses the order of the list	list_name.reverse()
sort()	Sorts the list	List_name.sort()
del()	Element to be deleted is mentioned using list name and index.	del list.[-index]

List Comprehension

Containing only the fruits with the letter "a" in the name.	newlist = [x for x in fruits if "a" in x]
Only accept items that are not "apple":	newlist = [x for x in fruits if x != "-apple"]
Accept only numbers lower than 5:	newlist = [x for x in range(10) if x < 5]
Set the values in the new list to upper case:	newlist = [x.upper() for x in fruits]
Set all values in the new list to 'hello':	newlist = ['hello' for x in fruits]

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

List Comprehension (cont)

Return "orange" instead of "banana":	newlist = [x if x != "banana" else "orange" for x in fruits]

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []

for x in fruits:
if "a" in x:
newlist.append(x)

print(newlist)
```

The Syntax
newlist = [expression for item in iterable if condition == True]
The return value is a new list, leaving the old list unchanged.

Tuples

A tuple is a collection of objects which ordered and immutable. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Add Items

Since tuples are immutable, they do not have a build-in append() method, but there are other ways to add items to a tuple.
Convert the tuple into a list, add "orange", and convert it back into a tuple:

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)
```

Add tuple to a tuple.

You are allowed to add tuples to tuples, so if you want to add one item, (or many), create a new tuple with the item(s), and add it to the existing tuple
Create a new tuple with the value "orange", and add that tuple:

```
thistuple = ("apple", "banana", "cherry")
y = ("orange",)
thistuple += y

print(thistuple)
```

Access Items

You cannot access items in a set by referring to an index or a key.
But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

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

Once a set is created, you cannot change its items, but you can add new items.

Tuple Methods

count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Remove Items

Convert the tuple into a list, remove "apple", and convert it back into a tuple:

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)
```

The del keyword can delete the tuple completely:

```
thistuple = ("apple", "banana", "cherry")
del thistuple
print(thistuple) #this will raise an error because the tuple no longer exists
```

Change Tuple Values

Once a tuple is created, you cannot change its values. Tuples are unchangeable, or immutable as it also is called.
But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```

Sets

A Set is an unordered collection data type that is iterable, mutable and has no duplicate elements.

Set is define in { }

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Add Items

To add one item to a set use the add() method.
Example
Add an item to a set, using the add() method:

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

To add items from another set into the current set, use the update() method.

Example
Add elements from tropical into thisset:

```
thisset = {"apple", "banana", "cherry"}
tropical = {"pineapple", "mango", "papaya"}
thisset.update(tropical)
print(thisset)
```

The object in the update() method does not have to be a set, it can be any iterable object (tuples, lists, dictionaries etc.).

Remove Set Items

remove()	thisset.remove("banana")
discard()	thisset.discard("banana")
clear()	method empties the set
del keyword	will delete the set completely(del thisset)

We can also use the pop() method to remove an item, but this method will remove the last item. Remember that sets are unordered, so you will not know what item that gets removed.

The return value of the pop() method is the removed item.

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

Set Methods

add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not

Set Methods (cont)

issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

Dictionary

Dictionary in Python is a collection of keys values, used to store data values like a map, which, unlike other data types which hold only a single value as an element.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Adding Items in Dictionary

Adding an item to the dictionary is done by using a new index key and assigning a value to it:

Example

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
thisdict["color"] = "red"
print(thisdict)
```

The **update()** method will update the dictionary with the items from a given argument. If the item does not exist, the item will be added.

The argument must be a dictionary, or an iterable object with key:value pairs.

Example

Add a color item to the dictionary by using the update() method:

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
thisdict.update({"color": "red"})
```

Removing Items in Dictionary

The **pop()** method removes the item with the specified key name:	thisdict.pop("model")
The **popitem()** method removes the last inserted item	thisdict.popitem()
The **del** keyword removes the item with the specified key name:	del thisdict["model"]

Removing Items in Dictionary (cont)

The **del** keyword can also delete the dictionary completely:	del thisdict
The **clear()** method empties the dictionary	thisdict.clear()

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
```

Dictionary Methods

clear()	Removes all the elements from the dictionary	dictionary.clear()
copy()	Returns a copy of the dictionary	dictionary.copy()
fromkeys()	Returns a dictionary with the specified keys and value	dict.fromkeys(keys, value)
get()	Returns the value of the specified key	dictionary.get(keyname, value)
items()	Returns a list containing a tuple for each key value pair	dictionary.items()
keys()	Returns a list containing the dictionary's keys	dictionary.keys()

Dictionary Methods (cont)

pop()	Removes the element with the specified key	dictionary.pop(keyname, defaultvalue)
popitem()	Removes the last inserted key-value pair	dictionary.popitem()
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value	dictionary.setdefault(keyname, value)
update()	Updates the dictionary with the specified key-value pairs	dictionary.update(iterable)
values()	Returns a list of all the values in the dictionary	dictionary.values()

Python Modules

Import our program as python module

-> Create file in notepad using .py extension

-> Upload it in sample data

-> Copy path of the uploaded file

Code:

```
from google.colab import files
!cp path /content
```


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com