
Data Analytics
+ Python
Advanced /
compound
Data Types
• Lists
• Tuples
• Sets
• Dictionaries

Lists

There are 3 other types: tuple, set and dictionary

Used to store multiple items inside a variable

Created always with []

There is always a strict order BUT they can be modified

You CAN have duplicates in a list

You can make lists out of any data type

```
thislist = ["apple", "banana", "cherry"]
```

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))

print(thislist[1])

thislist[1] = "blackcurrant“

thislist[1:3] = ["blackcurrant", "watermelon"]

thislist.append("orange")

thislist.insert(1, "orange")
```

```
thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]
thislist.extend(tropical)

thislist.remove("banana")

thislist.pop(1)

thislist.pop()

del thislist[0]
```

```
thislist = ["apple", "banana", "cherry"]

del thislist

thislist.clear()

thislist.sort()

thislist.sort(reverse = True)

mylist = thislist.copy()
```

```
list1 = ["apple", "banana", "cherry"]
list2 = [“kiwi", “strawberry", “grape"]

list3 = list1 + list2
print(list3)
```

Tuples

A collection which is ordered and UNCHANGEABLE

Written with ()

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

You can then go about finding length and indexing.

You CANNOT change, add or remove once it’s created

(there are workarounds but that’s for a later time)

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

Let’s pack a tuple (assign a value)

```
fruits = ("apple", "banana", "cherry")

(green, yellow, red) = fruits
print(green)
print(yellow)
print(red)
```

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

You can add tuples together like you do with lists

You can multiple by doing this:

```
mytuple = fruits * 2
print(mytuple)
```

Sets

A collection which is UNORDERED, UNCHANGEABLE & UNINDEXED

Written in {}

```
thisset = {"apple", "banana", "cherry“}
```

Do not allow duplicates

No particular order

Unchangeable

```
thisset = {"apple", "banana", "cherry“}
```

Can find the length the same way

Can be any data type

You can access values the same way

.pop() and .clear() work the same

```
thisset.add("orange")
```

```
Mylist = [“kiwi”, “orange”]
thisset.update(mylist)
```

```
thisset = {"apple", "banana", "cherry“}
```

```
thisset.remove("banana")
```

```
thisset.discard("banana")
```

```
thisset1 = {“apple”, “banana”, “cherry”}
thisset2 = {“kiwi”, “grape”}
thisset3 = thisset1.union(thisset2)
```

OR replace .union with .update

Dictionaries

Used to store data values in key:value pairs.

A collection which is ordered, changeable & do not allow duplicates.

`Written with {}`

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

```
print(thisdict["brand"])
```

You can change dictionaries

NO duplicates allowed

```
print(len(thisdict))
```

```
x = thisdict["model"]
print(x)
```

```
x = thisdict.get("model")
print(x)
```

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

```
thisdict["year"] = 2018
print(thisdict)

thisdict["color"] = "red"
print(thisdict)

thisdict.update({"color": "red"})
print(thisdict)

thisdict.pop("model")
print(thisdict)

thisdict.popitem()
print(thisdict)
```

Practice

- Create a file and name it listpractice
- Create a list, find the length, replace a part of it with something else and append it
- Create a tuple and multiply it by 3
- Create a set, add your list to it and remove a string
- Create a dictionary and print out 1 value, replace another one

- Save your file and send it to me in Slack via private message