# Python Course

List, Dictionaries & Tuples

Feb 2023 Matin KarimPour



# A list is a sequence

Like a string, a list is a sequence of values. In a string, the values are characters; in a list, they can be any type. The values in list are called elements or sometimes items.

```
python:)
[10, 20, 30, 40]
['crunchy frog', 'ram bladder']
['spam', 2.0, 5, [10, 20]]
cheeses = ['Cheddar', 'Edam', 'Gouda']
numbers = [17, 123]
empty = []
print(cheeses, numbers, empty)
# ['Cheddar', 'Edam', 'Gouda'] [17,
123 [ ]
```

# Lists are mutable

```
python:)
numbers = [17, 123]
numbers[1] = 5
print(numbers)
# [17, 5]
cheeses = ['Cheddar', 'Edam', 'Gouda']
'Edam' in cheeses
# True
'Brie' in cheeses
# False
```

### Traversing a list

```
python:)
for cheese in cheeses:
   print(cheese)
for i in range(len(numbers)):
   numbers[i] = numbers[i] * 2
```

### List operations

```
python:)
a = [1, 2, 3]
b = [4, 5, 6]
c = a + b
print(c)
# [1, 2, 3, 4, 5, 6]
[0] * 4
# [0, 0, 0, 0]
[1, 2, 3] * 3
#[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

#### List slices

```
python:)
t = ['a', 'b', 'c', 'd', 'e', 'f']
t[1:3]
t[:4]
# ['a', 'b', 'c', 'd']
t[3:]
# ['d', 'e', 'f']
t[1:3] = ['x', 'y']
print(t)
```

#### **List methods**

```
python:)
t = ['a', 'b', 'c']
t.append('d')
print(t)
# ['a', 'b', 'c', 'd']
t1 = ['a', 'b', 'c']
t2 = ['d', 'e']
t1.extend(t2)
print(t1)
# ['a', 'b', 'c', 'd', 'e']
t = ['d', 'c', 'e', 'b', 'a']
t.sort()
print(t)
# ['a', 'b', 'c', 'd', 'e']
```

### Deleting elements

```
python:)
t = ['a', 'b', 'c']
x = t.pop(1)
print(t)
print(x)
# b
del t[1]
print(t)
t.remove('b')
print(t)
t = ['a', 'b', 'c', 'd', 'e', 'f']
del t[1:5]
print(t)
```

### Lists and functions

```
python:)
nums = [3, 41, 12, 9, 74, 15]
print(len(nums))
# 6
print(max(nums))
# 74
print(min(nums))
print(sum(nums))
# 154
```

# Lists and strings

```
python:)
s = 'spam'
t = list(s)
print(t)
s = 'pining for the fjords'
t = s.split()
print(t)
# ['pining', 'for', 'the', 'fjords']
x = ' '.join(t)
print(x)
# 'pining for the fjords'
```

# Dictionaries

#### **Dictionaries**

A dictionary is like a list, but more general. In a list, the index positions have to be integers; in a dictionary, the indices can be (almost) any type.

```
python:)
eng2sp = dict()
print(eng2sp)
eng2sp['one'] = 'uno'
print(eng2sp)
eng2sp = {'one': 'uno', 'two': 'dos',
'three': 'tres'}
print(eng2sp)
print(eng2sp['two'])
len(eng2sp)
'one' in eng2sp
'uno' in eng2sp
# Fals
vals = list(eng2sp.values())
'uno' in vals
```

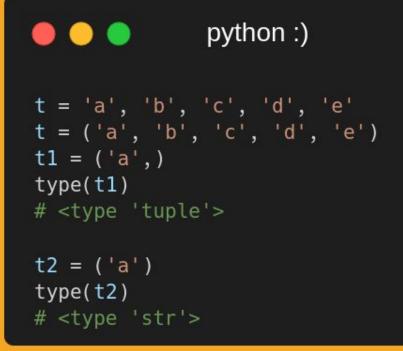
#### Looping and dictionaries

```
python:)
counts = { 'chuck' : 1 , 'annie' : 42,
'jan': 100}
for key in counts:
    print(key, counts[key])
#jan 100
#chuck 1
#annie 42
for key in counts:
    if counts[key] > 10:
        print(key, counts[key])
#jan 100
#annie 42
```

# Tuples

### Tuples are immutable

A tuple is a sequence of values much like a list. The values stored in a tuple can be any type, and they are indexed by integers. The important difference is that tuples are immutable. Tuples are also comparable and hashable so we can sort lists of them and use tuples as key values in Python dictionaries.



## Tuples

```
python:)
numbers = (1, 2, 3)
len(numbers)
# 3
values = (1, 3, 5, 7, 9)
values[2]
# 5
values[2:4]
# (5, 7)
values[0] = 2
# TypeError: 'tuple' object does not
support item assignment
```

### Tuples

```
python:)
vowels = ("a", "e", "i", "o", "u")
for vowel in vowels:
 print(vowel.upper())
# E
# 0
# U
coordinates = 4.21, 9.29
print(type(coordinates))
# <class 'tuple'>
x, y = coordinates
# 4.21
def adder_subtractor(num1, num2):
    return (num1 + num2, num1 - num2)
adder_subtractor(3, 2)
```

## Any Questions?

