

Developer Operations

Python Overview 2: Useful Data Structures

Presentation Overview

- More on Strings
- Lists
- Tuples
- Dictionaries

Recap: all variables are objects

- Everything is an object
- Data type is a property of the *object* and not of the variable

```
>>> x = 7
>>> print(x)
7
>>> x = 'Hello'
>>> print(x)
'Hello'
>>>
```

More on strings

- Python ***slice*** notation can be used to quickly access elements of strings and substrings

```
>>> s = '012345'
```

```
>>> s[3]
```

```
'3'
```

Element 3 (counting from zero)

```
>>> s[1:4]
```

```
'123'
```

Element 1 up to, but not including, element 4

```
>>> s[2:]
```

```
'2345'
```

From character 2 to the end of the string

```
>>> s[:4]
```

```
'0123'
```

Full string up to, but not including, element 4

```
>>> s[-2]
```

```
'4'
```

*2nd element from the **end***

```
>>>
```

String methods

- Python has a wide range of built-in string methods; e.g.

```
>>> s = 'Hello'
```

```
>>> len(s)
```

```
5
```

```
>>> s.upper()
```

```
'HELLO'
```

```
>>> str(10.3)
```

```
'10.3'
```

```
>>>
```

Number of characters in the string

Convert to upper case

String representation of non-string object

- See <https://docs.python.org/3/library/string.html> for a *lot* more string methods

Lists

- *Ordered* collection of data
- Data can be of different types
- Same subset (slice) operations as Strings

```
>>> x = [1, 'hello', 3.14159]
>>> print(x)
[1, 'hello', 3.14159]
>>> x[2]
3.14159
>>> x[:2]
[1, 'hello']
>>>
```

Lists: Modifying Content

- Lists are *mutable*
 - *i.e. they can be edited by adding, removing, updating elements, etc*
- **`x[i] = a`** reassigns the i^{th} element to the value `a`
- Since `x` and `y` point to the same list object, *both* are changed
- The method **`append`** also modifies the list

```
>>> x = [1,2,3]
>>> y = x
>>> x[1] = 15
>>> print (x)
[1, 15, 3]
>>> print (y)
[1, 15, 3]
>>> x.append(12)
>>> print(x)
[1, 15, 3, 12]
>>> print(y)
[1, 15, 3, 12]
>>>
```

Lists: Modifying Contents


- The method **append** modifies the list and returns **None**
- List addition (+) returns a new list

```
>>> x = [1,2,3]
>>> y = x
>>> z = x.append(12)
>>> z == None
True
>>> y
[1, 2, 3, 12]
>>> x = x + [9,10]
>>> x
[1, 2, 3, 12, 9, 10]
>>> y
[1, 2, 3, 12]
>>>
```


Tuples

- Tuples are *immutable* versions of lists
- One strange point is the format to make a tuple with one element:
' ,' is needed to differentiate from the mathematical expression (2)

```
>>> x = (1,2,3)
>>> x[1:]
(2, 3)
>>> y = (2,)
>>> y
(2, )
>>>
```



Dictionaries

- A set of key-value pairs
- Very useful for storing certain kinds of data

```
>>> temps = {'Dublin':15, 'Paris':18, 'Madrid':25,
             'New York':23, 'London':16}
>>> temps
{'London': 16, 'Dublin': 15, 'New York': 23,
'Madrid': 25, 'Paris': 18}
>>> temps['Dublin']
15
>>>
```

Dictionaries

- Dictionaries are *mutable*

```
>>> temps = {'Dublin':15, 'Paris':18, 'Madrid':25,
             'New York':23, 'London':16}
>>> temps['Dublin'] = 13
>>> temps
{'London': 16, 'Dublin': 13, 'New York': 23,
'Madrid': 25, 'Paris': 18}
```

Dictionaries: Adding elements

- Assigning to a key that does not exist adds an entry

```
>>> temps['Rome']=27
>>> temps
{'New York': 23, 'Paris': 18, 'Rome': 27, 'Dublin': 13, 'Madrid': 25, 'London': 16}
>>>
```

Dictionaries: Deleting Elements

- The **del** method deletes an element from a dictionary

```
>>> temps
{'New York': 23, 'Paris': 18, 'Rome': 27, 'Dublin': 13, 'Madrid': 25, 'London': 16}
>>> del(temps['New York'])
>>> temps
{'Paris': 18, 'Rome': 27, 'Dublin': 13, 'Madrid': 25, 'London': 16}
>>>
```

Copying lists

- **list()** can create a new list as a copy of an existing one

```
>>> first = [1,2,3]
>>> second = list(first)
>>> first[1] = 5
>>> print (first)
[1, 5, 3]
>>> print (second)
[1, 2, 3]
>>>
```

- Or can use **copy()** method

```
>>> first = [1,2,3]
>>> second = first.copy()
>>>
```

Copying dictionaries

- **dict()** can create a new dictionary as a copy of an existing one

```
>>> tempsyest = {'Paris':18, 'Rome':21}
>>> tempstoday = dict(tempsyest)
>>> tempstoday['Rome'] = 23
>>> print(tempsyest)
'Paris': 18, 'Rome': 21
>>> print(tempstoday)
'Paris': 18, 'Rome': 23
```

- Or can use **copy()** method

```
>>> tempsyest = {'Paris':18, 'Rome':21}
>>> tempstoday = tempsyest.copy()
>>>
```

Data type summary

- Lists, Tuples, and Dictionaries can store any type (including other lists, tuples, and dictionaries!)
- Only lists and dictionaries are mutable
- All variables are references