Python Class 1: Syntax

Betül Demirkaya

August 3, 2015

Quiz

Object Types
- String
- Int
- Float
- List
- Dictionary

Conditionals

Loop

Functions

Appendix

Quiz

- Please go to: http://betuld.github.io/quiz
  http://betuld.github.io/quiz2

String

- Any group of characters recognized as text.
- Written between single quotes, double quotes or triple quotes.

```
>>> myname='Betul'
>>> myage='34'
>>> intro="I'm Betul."
>>> long_intro="""Hello!
... I'm Betul.
... What's up?"""
>>> long_intro
"Hello!\nI'm Betul.\nWhat's up?"
```

String

- You can call any character in the string.

```
>>> myname[0]
'B'
```

- Strings are immutable.

```
>>> myage[0]=2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

- But you can split a string into words.

```
>>> intro.split()
["I'm", 'Betul.']
```

- Or into any other chunks using a character.

```
>>> long_intro.split('\n')
['Hello!', "I'm Betul.", "What's up?"]
```

String

- It requires a little more work to split a string into letters.

```
>>> [letter for letter in myname]
['B', 'e', 't', 'u', 'l']
```

- Let's combine them again.

```
>>> myletters=[letter for letter in myname]
>>> ''.join(myletters)
'Betul'
>>> '-'.join(myletters)
'B-e-t-u-l'
```

Int

- Integers.
- You can do mathematical operations using these.
  - Usual suspects: + - * /
  - Exponentiate: **
  - Remainder: %
- Remember the results are *always* rounded down!
- You can assign numbers using different operators.

```
>>> mynumber=1
>>> mynumber+=1
>>> mynumber
2
```

Float

- Real numbers.
- Written by adding the decimal to an integer.

```
>>> 12.0
12.0
>>> float(12)
12.0
```

List

- Collection of any type objects - even lists.

```
>>> myletters
['B', 'e', 't', 'u', 'l']
```

- Let's call the first four elements.

```
>>> myletters[0:4]
['B', 'e', 't', 'u']
>>> myletters[:4]
['B', 'e', 't', 'u']
>>> myletters[:-1]
['B', 'e', 't', 'u']
```

- Let's call all elements except the first one.

```
>>> myletters[1:5]
['e', 't', 'u', 'l']
>>> myletters[1:]
['e', 't', 'u', 'l']
```

List

- Let's call every other element.

```
>>> myletters[::2]
['B', 't', 'l']
```

- Let's change the first element.

```
>>> myletters[0]='b'
>>> myletters
['b', 'e', 't', 'u', 'l']
```

- Let's add an element.

```
>>> myletters.append('D')
>>> myletters
['b', 'e', 't', 'u', 'l', 'D']
```

Dictionary

- It is what it sounds like.
- Here is how you create one.

```
>>> id={'name':'Betul','last_name':'Demirkaya','age':34}
```

- Unlike lists, there is no order to elements.
- You call elements using keys.

```
>>> id
{'age': 34, 'last_name': 'Demirkaya', 'name': 'Betul'}
>>> id.keys()
['age', 'last_name', 'name']
>>> id.values()
[34, 'Demirkaya', 'Betul']
>>> id['age']
34
```

Conditionals

```
if 1==1:
    print 'Condition 1 is satisfied.'
elif 2==2: #Evaluated only if condition 1 was not satisfied!
    print 'Condition 2 is satisfied.'
else:
    print 'No condition is satisfied.'
```

Examples with `for` loop

```
>>> even_numbers=[]
>>> for i in range(1,6):
...     if i%2==0:
...         even_numbers.append(i)
...
>>> for letter in 'word':
...     print letter
...
w
o
r
d
```

Functions

- They help write cleaner code.
- Keep them simple.
- You can return any type of object.
- Don't forget to add `return` for output.

Lab

- Start with the easiest solution you can think of.
- Write out comments.
- Use clear variable names.

Terminal Commands

- `cd`: change directory
- `pwd`: show the current directory
- `ls`: list folders and files in the current directory