Python Language Semantics

Thursday, November 12, 2020 12:39 PM

Language Semantics

Indentations instead of braces

- Other languages often use brackets etc to build code blocks. For example consider this Arduino for loop syntax

```
void loop() {
  for (int i = 0; i <= 255; i++) {
    analogWrite(PWMpin, i);
    delay(10);
  }
}
```

- Python uses white space (4 spaces or 1 tab) to denote code blocks

```
Class_list = ['Tom', 'nick', 'krish', 'jack']
for student in Class_list:
    print(student)
```

- You don't need semicolon to finish a statement, but you can use semicolon to separate multiple statements on the same line

```
Student1 = 'Tom'; Student2 = 'Nick'; Student3 = 'Krish'
```

- Comments can be written by first typing the # symbol
  - You can also highlight multiple lines and type Ctrl+1 to toggle section on/off into comments

```
Student1 = 'Tom'; Student2 = 'Nick'; Student3 = 'Krish'
#We can brainstorm names in this section of code
#Student1 = 'Odin'; Student2 = 'Loki'; Student3 = 'Ragnar'
```

- Functions are called with parenthesis by passing zero or more arguments and the function can, if desired, return values to a variable

```
x = quadratic_formula(a,b,c)
```

- Naming a variable creates a reference to the object being named, it does not create or recreate the object

```
Restaurant = 'Red Iguana'
Name = Restaurant
```

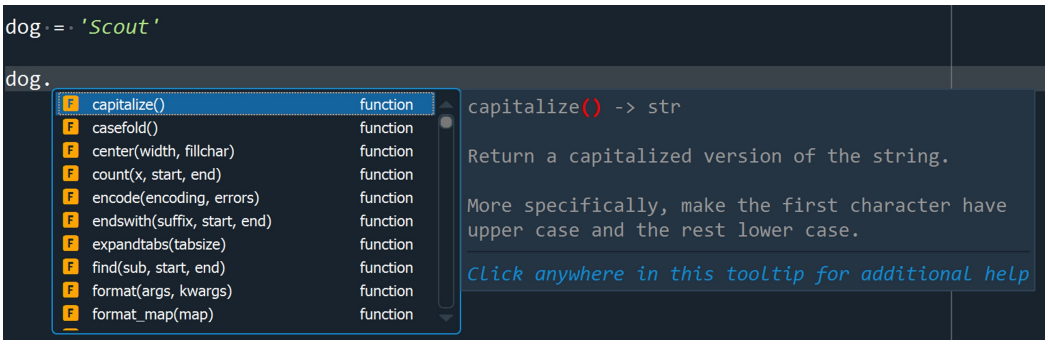
The data 'Red Iguana' did not get copied, we just have the variables Restaurant and Name both pointing to the same data

- Variables can change type (int, float, string etc) without problem

```
Restaurant = 'Red Iguana'
Name = Restaurant
Restaurant = 500
```

The data 'Red Iguana' did not get copied, we just have the variables Restaurant and Name both pointing to the same data

- Variables have lots of attributes and methods than can be accessed by typing variable name period then hitting tab



- Duck Typing is when you don't care about the type of the object, you just care if it has a certain method or behavior.
  - "Walks like a duck, quacks like a duck, it's a duck"

```
def isiterable(obj):
    try:
        iter(obj)
        return True
    except TypeError: # Not iterable
        return False

isiterable('a string') # output would be True
isiterable([1,2,3]) # output would be True
isiterable(3) # output would be False
```

- Importing modules is possible in python. A module is any file ending in ".py" So if you did a bunch of work on a module you can just call it later

```
#name_joiner.py

def student_name(first,last):
    return first + ' ' + last
```

Then call it

```
import name_joiner
Student1 = 'Tom'; Student2 = 'Nick'; Student3 = 'Krish'
new_name = name_joiner.student_name(Student1, Student2)
```

Great cheat sheet available here
http://sixthresearcher.com/wp-content/uploads/2016/12/Python3_reference_cheat_sheet.pdf