Python_code

July 13, 2020

1 Python code

2 From "Programming with Data", Chapter 2 of *Data Science for Mathematicians*

2.1 Section 2.3.1

Rather than using braces or some other kind of punctuation to enclose the body of functions or control statements (like `if/then/else` statements or `for` or `while` loops), Python uses whitespace:

[1]:
```
for i in range(10):
    if i % 2 == 0:
        print('{number} is even'.format(number = i))
    else:
        print('{number} is odd'.format(number = i))
```

```
0 is even
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
```

Variables should be named in sensible, comprehensible ways. It's probably okay to use something like `i`, `j`, or `k` as a simple index in a `for` loop, but other variables should be named to indicate what they are supposed to store:

[2]:
```
last_name = "Raleigh"
```

[3]:
```
# We have to define previous_count and new_count
# for the line in the text to work and not throw an error.
previous_count = 5
new_count = 2
total_count = previous_count + new_count
```

One helpful convention for naming functions or methods is to use verbs that communicate the "action" of the function. The following code defines a Python function to find the divisors of a number:

[4]:
```python
def find_divisors(n):
    div_list = []
    for i in range(n):
        if n % (i + 1) == 0:
            div_list.append(i + 1)
            return div_list
```

If a function returns a logical value, it might be named as follows:

[5]:
```python
def are_equal(x, y):
    if x == y:
        return True
    else:
        return False
```

Mathematicians are accustomed to writing out long, multi-line expressions. This is much harder to do with code and the results are difficult to read and debug. Consider the following Python code, which computes the sample standard deviation of a list of numbers, using the standard mathematical formula converted directly into Python code.

[6]:
```python
# We have to import the statistics module (calling it stats)
import statistics as stats

def calculate_sample_variance_short(data):
    return sum([(data_value - stats.mean(data))**2
                for data_value in data])/(len(data) - 1)
```

A much better approach is to keep expressions simple by first defining simple subexpressions on their own lines. Following that, you can define the main expression as a simple formula using the previously defined subexpressions. So, for example, note the improved readability in the next Python example, which accomplishes the same task. While the former is more concise, it's also less readable than the second and less clear about the sequence of steps involved in the calculation.

[7]:
```python
def calculate_sample_variance_readable(data):
    data_mean = stats.mean(data)
    n = len(data)
    squared_deviations = [(data_value - data_mean)**2
                          for data_value in data]
    return sum(squared_deviations)/(n - 1)
```

2.2 Section 2.3.2

To illustrate the code in this section, we'll create a small dataframe.

[8]:
```
import pandas as pd

datalist = [['Adelphi University', '$41,922', '$17,150'],
            ['Agnes Scott College', '$44,606', '$13,177'],
            ['Alabama State University', '$20,733', '$6,292'],
            ['Albany College of Pharmacy', '$37,565', '$11,811'],
            ['Albertus Magnus', '$34,081', '$14,466']]

data = pd.DataFrame(datalist, columns = ['Name', 'Tuition', 'Room and Board'])
data
```

[8]:
```
                         Name  Tuition Room and Board
0          Adelphi University  $41,922        $17,150
1         Agnes Scott College  $44,606        $13,177
2    Alabama State University  $20,733         $6,292
3  Albany College of Pharmacy  $37,565        $11,811
4             Albertus Magnus  $34,081        $14,466
```

[9]:
```
data["Tuition"] = data["Tuition"].str.replace("$", "")
data["Tuition"] = data["Tuition"].str.replace(",", "")
data["Tuition"] = data["Tuition"].astype(float)
```

[10]:
```
data
```

[10]:
```
                         Name  Tuition Room and Board
0          Adelphi University  41922.0        $17,150
1         Agnes Scott College  44606.0        $13,177
2    Alabama State University  20733.0         $6,292
3  Albany College of Pharmacy  37565.0        $11,811
4             Albertus Magnus  34081.0        $14,466
```

Given the utility of the task performed by the code above, it's not much more effort to write a little function that does the same thing, but for any column of any data set:

[11]:
```
def convert_currency_to_float(data, col):
    data[col] = data[col].str.replace("$", "")
    data[col] = data[col].str.replace(",", "")
    data[col] = data[col].astype(float)
    return data
```

Let's see how the function works on the "Room and Board" column.

[12]:
```
convert_currency_to_float(data, "Room and Board")
data
```

[12]:
```
                         Name  Tuition  Room and Board
0          Adelphi University  41922.0         17150.0
1         Agnes Scott College  44606.0         13177.0
```

```
2    Alabama State University  20733.0          6292.0
3  Albany College of Pharmacy  37565.0         11811.0
4             Albertus Magnus  34081.0         14466.0
```

2.3 Figure 2.8

```
[13]: my_list = [
          [(i + 1) for i in range(10)],
          ["Sean", "Raleigh"],
          {"a": 1, "b": 2, "c": 3, "d": 4, "e": 5}
      ]
```

```
[14]: my_list
```

```
[14]: [[1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
       ['Sean', 'Raleigh'],
       {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}]
```