6_python

January 4, 2024

1 Coding is Just Coding

Once you learn one coding language, you can read (and sometimes even write) in a lot of other coding languages. Today we're going to go through how to code what we did on the first day in Python, so that we see how the syntax is slightly different, but you can still read a lot of it. The point of this is to realize that if someone hands you a code script in a different language, you shouldn't freak out.

1.1 Getting started

In R Studio, create a python script. This will require the `retriculate` package and you may also need to install the `IRkernel` package, as well.

We're now going to go through what we did on day one.

2 Basic Data Types

[14]:

```
# Andie Creel / January 2024 / Goal: Redo day one in python

# Run basic arithmatic

2 + 3

# variable assingment is done with an '=' sign, instead of the '<-' sign
a = 2
b = 3

a + b



# Numeric -- integer: no decimal points
myInt = 1

# Numeric -- floating point: decimal points
myNum = 2.4

# logical (Boolean): a true/false statement. Use parentheses to evaluate if␣
 ↪something is true or false
```

```
myBool_1 = (3 < 4)
myBool_2 = (3 > 4)

# character (string)
myChar_a = "a"
myChar_b = 'b'
```

3 Ways to store datatypes

You will need to install the `numpy` package by running `pip install numpy` in the terminal. NumPy is the main package for scientific computing in python.

3.1 Vectors and Matrices

Notice that indexing in python starts at 0, rather than 1 (as it did in R). In python, we just use lists instead of vectors.

[63]:

```
import numpy as np

# Lists can contain elements of different data types
myVec_n = [1, 2, 3, 4, 5]
print(myVec_n[0])

myVec_s = ["str", "b", "c"]
print(myVec_s[0])

myVec_all = ["str", 1, True]
print(myVec_all)
```

```
1
str
['str', 1, True]
```

[ ]:

```
# NumPy array (similar to R matrix): should contain elements of the same data␣
 ↪type
# In this case, we're creating a 2x5 matrix
# the . here works similar to %>% in dplyr
myMat_n = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).reshape(2, 5)
myMat_n
```

3.2 Lists

[32]:

```
# Lists: Can contain elements of different data types, including other lists or␣
 ↪arrays
myList = [2, "c", myMat_n]

# Accessing the first element of the list
myList[0]  # returns numeric (2 in this case)
```

```
[32]: 2
```

```
[33]: myList[1] # returns C
```

```
[33]: 'c'
```

```
[34]: myList[2] # returns the matrix
```

```
[34]: array([[ 1,  2,  3,  4,  5],
             [ 6,  7,  8,  9, 10]])
```

3.3 Data frames

To work with data frames, we need to install and load the `pandas` package. Run `pip install pandas` in your terminal.

```
[35]: import pandas as pd

      # Create a DataFrame from the NumPy array
      myDF = pd.DataFrame(myMat_n)
      myDF
```

```
[35]:    0  1  2  3   4
      0  1  2  3  4   5
      1  6  7  8  9  10
```

```
[36]: # Print column names (initially they are just integer indices)
      print(myDF.columns)
```

```
RangeIndex(start=0, stop=5, step=1)
```

Unlike R, python automatically names unamed columns with integrers.

```
[37]: # Rename the columns
      myDF.columns = ["age_yr", "weight_lb", "income_$", "height_ft", "height_in"]
      myDF
```

```
[37]:    age_yr  weight_lb  income_$  height_ft  height_in
      0       1          2         3          4          5
      1       6          7         8          9         10
```

```
[38]: # investigate one column (index is the left column, value is the right column)
      myDF['age_yr']
```

```
[38]: 0    1
      1    6
      Name: age_yr, dtype: int64
```

[41]:

```
# Create a new column
myDF['nonsense'] = myDF['age_yr'] + myDF['weight_lb']
myDF['nonsense']
```

[41]:

```
0     3
1    13
Name: nonsense, dtype: int64
```

[46]:

```
# Create a DataFrame
myPpl = pd.DataFrame({
    'gender': ["Male", "non-binary", "Female"],
    'male': [True, False, False],
    'height': [152, 171.5, 165],
    'weight': [81, 93, 78],
    'age': [42, 38, 26]
})

# Reference one column (either of these work)
myPpl['male']
```

[46]:

```
0     True
1    False
2    False
Name: male, dtype: bool
```

[47]:

```
myPpl.male
```

[47]:

```
0     True
1    False
2    False
Name: male, dtype: bool
```

4 Functions

`def` stands for definition. The syntax for writing a function is different, and is a good example of how white space is important in python (notice that there are no parentheses).

[54]:

```
def myF(x):
    y = x - x**2
    return y

myF(.5)
```

[54]:

```
0.25
```

5 Loops

Loops are another example where you can read the code even if you don't know python. However, they have some differnt syntax with the range function, specifically that the last value is excluded.

```
[55]: # Notice that 5 doesn't print
      for i in range(1, 5):  # range(start, stop) in Python is inclusive of start and␣
       ↪exclusive of stop
          print(i)
```

```
1
2
3
4
```

```
[64]: # combining loop and function
      for i in range(1,5):
          y = myF(i/4)
          print(y)
```

```
0.1875
0.25
0.1875
0.0
```

6 If Else Statements

To do an if else statement, we need to introduce `lambda` functions in python. Lambda functions are one off functions that are simple enough they do not warrant defining. Using lambda functions allows for quick coding, but do reduce the readability of the code.

6.1 Lambda Function

```
[65]: # A simple lambda function that adds two numbers
      myAdd = lambda x, y: x + y

      myAdd(2,3)
```

```
[65]: 5
```

In the pandas apply functions, axis specifies if you're working across columns or rows. axis = 0: The function is applied to each column. This is the default behavior. axis = 1: The function is applied to each row.

```
[69]: # Using vectorized operations to adjust age if male
      myPpl['age_new_m'] = myPpl.apply(lambda row: row['age'] - 3 if row['male'] else␣
       ↪row['age'], axis=1)
      myPpl
```

```
[69]:        gender   male  height  weight  age  age_new_m
      0        Male   True   152.0      81   42         39
      1  non-binary  False   171.5      93   38         38
      2      Female  False   165.0      78   26         26
```

7 Advanced

I wrote this is in a Jupyter Notebook, which is the pythons verion of an R notebook. To write a Jupyter Notebook using R, you need to do a few aditional (and advanced) steps.

In an R consol, run `install.packages('IRkernel')`

In the Terminal, run `jupyter notebook`

This will open up a Jupyter Notebook host in your web browser. You can create a new .ipynb file (aka a Jupyter Notebook).

Unlike an R Markdown file, you cannot automatically knit it to an HTML or pdf document. However, you can download it as an HTML, or you can download a LaTex file and compile the LaTex file to a PDF on your computer (which is a more advanced step then downloading the HTML).

```
[ ]:
```

```
[ ]:
```