Getting Started with Python

CAC 350

Last Time

* Discussed why study data science, what it is, and how it relates to other fields

* Today:
  * Reading quiz
  * Getting started with Python - making sure everyone is on the same page
  * A little numpy

Installing Python/ Jupyter Notebook

* Any issues?
* Go ahead and open Jupyter Notebook

?

* Shorthand for accessing help documentation
* Example:

  L=[1,2,3]

  L.append?

* L?

docstring

* What's the purpose of a docstring?

```
def square(n):
    '''Takes in a number n, returns the square of n'''
    return n**2
```

* What's the purpose of a docstring?

```
print(square.__doc__)
```

* Command line:

square?

* What's the difference between square? and square??

Tab Completion

* Another tool to help you...
* Autocompletion and exploration, like a dir function
  * object, module, or namespace.<literally press tab>
  * Example: import math
  * math.<literally press tab>

Pasting

* Jupyter Notebook doesn't have this magic
* %magic will list all of the magic statements
* Cheatsheet: https://www.edureka.co/blog/wp-content/uploads/2018/10/Jupyter_Notebook_CheatSheet_Edureka.pdf
* Documentation: https://ipython.readthedocs.io/en/stable/interactive/magics.html

Saving Scripts

%%writefile

In [15]:
```
def square(x):
    return x**2

for N in range(1,4):
    print(N, "squared is", square(N))
```

```
1 squared is 1
2 squared is 4
3 squared is 9
```

In [17]:
```
%%writefile myscript.py
def square(x):
    return x**2

for N in range(1,4):
    print(N, "squared is", square(N))
```

```
Writing myscript.py
```

Running Existing Code

* myscript.py contains previously entered code that I saved
* %run myscript.py
* Advantages?

Try This

import math

find the square root of 43

find the square root of 21

add the two together

In/Out

* When you see the prompt:

In[2]: 'some math problem'

Out[2]:1.345

* What do these really represent?
* Why is this helpful?

Fix This

```
import math

find the square root of 43

find the square root of 21

add the two together
```

Suppressing Output

* Why might you want to do it?
* How do you do it?

Errors and Debugging

* Traceback: print of the stack trace when an error occurs - includes information about the cause of the error

Errors and Debugging

Find the error:

```
def favorite_ice_cream():
    ice_creams = [
        "chocolate",
        "vanilla",
        "strawberry"
    ]
    print(ice_creams[3])

favorite_ice_cream()
```

Errors and Debugging

Find the error:

```
def some_function()
    msg = "hello, world!"
    print(msg)
     return msg
```

Errors and Debugging

```
def print_message(day):
    messages = {
        "monday": "Hello, world!",
        "tuesday": "Today is tuesday!",
        "wednesday": "It is the middle of the week.",
        "thursday": "Today is Donnerstag in German!",
        "friday": "Last day of the week!",
        "saturday": "Hooray for the weekend!",
        "sunday": "Aw, the weekend is almost over."
    }
    print(messages[day])

def print_friday_message():
    print_message("Friday")

print_friday_message()
```

* How many levels does the traceback have?
* What is the function name where the error occurred?
* On which line number in the function did the error occur?
* What is the type of error?
* What is the error message?

Profiling and Timing Code

* Donald Knuth: "We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil."

* But...once done, we may want to check:
  * execution time of a given command
  * is there a bottleneck somewhere?

Timing Code

* What's the value?
* Difference between %timeit and %%timeit?
* %timeit L = [n**2 for n in range(1000)]
* %%timeit
  L = []
  for n in range(1000):
      L.append(n**2)

%time

Profiling Scripts

In [60]:
```
def sum_of_lists(N):
    total = 0
    for i in range(5):
        L = [j ^ (j >> i) for j in range(N)]
        total += sum(L)
    return total
```

In [61]:
```
%prun sum_of_lists(1000000)
```

```
         14 function calls in 1.008 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        5    0.894    0.179    0.894    0.179 <ipython-input-60-f105717832a2>:4(<listcomp>)
        5    0.056    0.011    0.056    0.011 {built-in method builtins.sum}
        1    0.044    0.044    0.994    0.994 <ipython-input-60-f105717832a2>:1(sum_of_lists)
        1    0.013    0.013    1.008    1.008 <string>:1(<module>)
        1    0.000    0.000    1.008    1.008 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Operations

* Can do mathematical operations, boolean comparisons, assigning variables

```
In [47]: 2+2

Out[47]: 4

In [48]: x = 2

In [49]: y = _47

In [50]: z = x + y

In [51]: z

Out[51]: 6
```

```
In [52]: 2 > 3, 3 > 2

Out[52]: (False, True)

In [53]: 2 == 2

Out[53]: True

In [54]: [2,3] == [2, 3]

Out[54]: True
```

```
In [55]: type(z)

Out[55]: int

In [56]: z = z/5

In [57]: type(z)

Out[57]: float

In [58]: z

Out[58]: 1.2
```

Simple Problems

1. Perform the arithmetic operation, 182 modulo 13 and store the result in a variable, named "output"
2. Print the value and data type of "output"
3. Check if the value stored in "output" is equal to 0
4. Repeat steps 1-3 with 182 divided by 13
5. Report if the data type of "output" is the same in both cases

Data Types

Not what they seem...

* Python is actually implemented in C
* When we say, x = 5...we really have a Python object pointing to a space in memory where there is a C integer containing the number 5

Lists

* Dynamically-typed list...Python pointer to a block of pointers that each point to their own memory location, which keeps track of the overhead of that location

* NumPy-style arrays are statically-typed, contiguous block of cells of the same data type
    * Disadvantage: Lack of flexibility
    * Advantage: More efficient for storing and manipulating data

Let's create a numpy array

numpy Array

```
In [63]: import numpy as np

In [64]: np.array([1,4,2,5,3])
Out[64]: array([1, 4, 2, 5, 3])

In [65]: np.array([1.5,4,2,5.2,3])
Out[65]: array([1.5, 4. , 2. , 5.2, 3. ])

In [66]: np.array([1,2,3,4],dtype='float32')
Out[66]: array([1., 2., 3., 4.], dtype=float32)

In [67]: np.array([range(i,i+3) for i in [2,4,6]])
Out[67]: array([[2, 3, 4],
                [4, 5, 6],
                [6, 7, 8]])
```

Let's Play

Could also use np.random.randn(25)

* Create an array with the following data...

[85,62,78,64,25,12,74,96,63,45,78,20,5,30,45,78,45,96,65,45,74,12,78,23,8]

* Find the max, min, mean, variance, standard dev., and median
* Plot a histogram with the info

Code

In [70]:
```
data1 = np.array([85,62,78,64,25,12,74,96,63,45,78,20,5,30,45,78,45,96,65,45,74,12,78,23,8])
```

In [71]:
```
max=np.max(data1)
```

In [73]:
```
min = np.min(data1)
```

In [74]:
```
mean = np.mean(data1)
```

In [75]:
```
variance = np.var(data1)
standarddev=np.std(data1)
```

In [76]:
```
median = np.median(data1)
```

In [78]:
```
def printStats():
    print('Max:',max)
    print('Min:',min)
    print('Mean:',mean)
    print('Variance:',variance)
    print('Standard Deviation:',standarddev)
    print('Median:',median)
printStats()
```

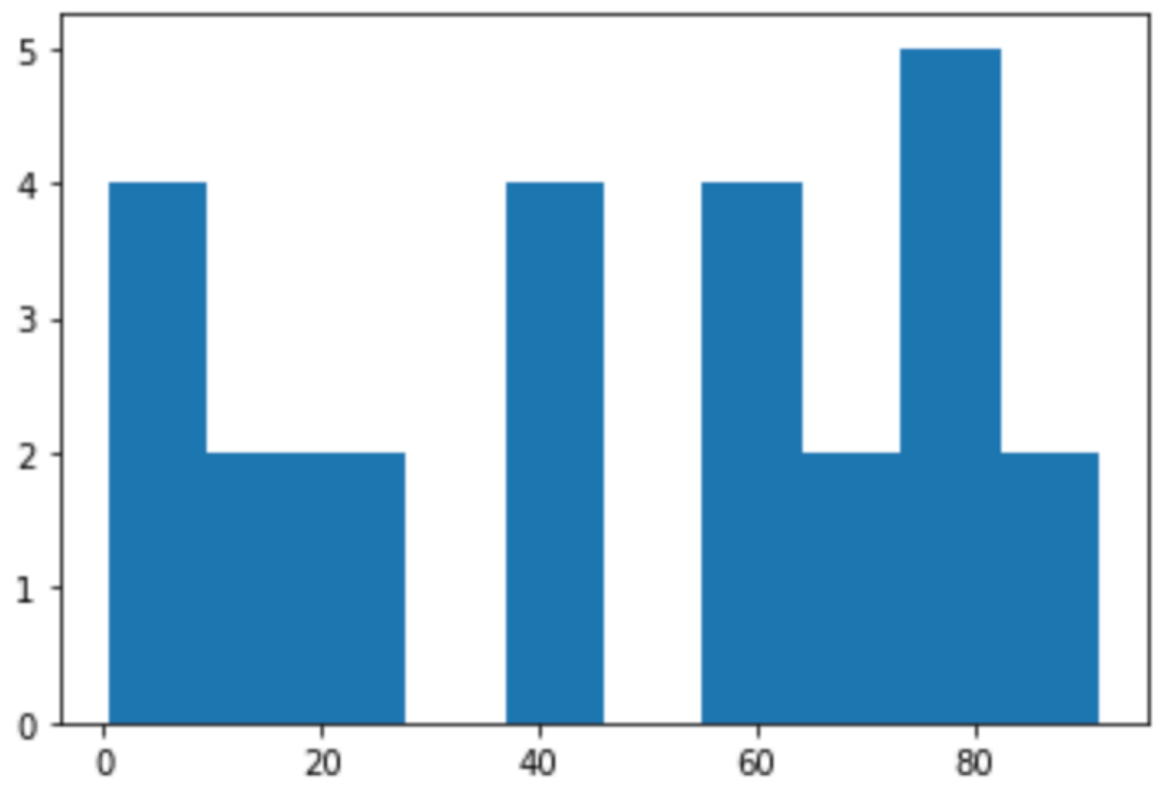
Code

In [79]:
```
import matplotlib.pyplot as plt
```

```
Matplotlib is building the font cache; this may take a moment.
```

In [80]:
```
plt.figure()
hist1, edges1 = np.histogram(data1)
plt.bar(edges1[:-1],hist1,width=edges1[1:]-edges1[:-1])
```

Out[80]: <BarContainer object of 10 artists>



matplotlib.pyplot.bar(*x*, *height*, *width=0.8*, *bottom=None*, *, *align='center'*, *data=None*, ***kwargs*)

Next Class

* Please read Chapter Two of Python Data Science Handbook: Numpy
* Reading quiz will be posted online