IceCube Bootcamp 2019: Introduction to Python

Kayla Leonard

Introduction to Python

- Information and notes in this font
- `Code, syntax, and examples in this font`

Please feel free to stop me and ask questions at any time!

Simple Variable Types

Python type	Description	Examples
int	integer	3, 0, -77, 10293
float	decimal numbers	1.5, 7e3, 1.0
str	string (text)	'Madison, WI', '27'
complex	complex number	7+4.3j
bool	conditional values	True, False
NoneType	empty variables	None

See Bootcamp_Tutorial_Part2.ipynb

Compound Variable Types

Python type	Description	Examples
list	ordered mutable list	[1,2,7,3,3,1,99]
tuple	ordered immutable list	(1,2,7,3,3,1,99)
set	No repeats, unordered	{1,2,3,7,99}
dict	Key-value pairs	{‘a’:1, ‘b’:2, ‘c’:2}

See Bootcamp_Tutorial_Part2.ipynb

List Indexing

- Can call specific element of a list using brackets []
- To get a portion of the list, use a colon :
- The numbering starts at **0** (not one)
- Example:
  - my_list = [1,2,7,3,3,1,99]
  - my_list[0] = 1
  - my_list[2:5] = [7,3,3]

Begins at 2nd element and goes up to but *not including* 5th element

- Downside: you must remember where each item is

See Bootcamp_Tutorial_Part2.ipynb

Dictionaries

- Dictionaries are a way to store key-value pairs
- Example:

```
person = {‘age’:24, ‘height’:5.83, ‘name’: ‘Kayla’}

print(person.keys)
    >> [‘age’, ‘height’, ‘name’]

person[‘age’]
    >> 24
person[‘name’]
    >> ‘Kayla’
```

You can store multiple types of variables in the same dictionary

You can reference data by key so you don’t have to remember what number it was in the list

See Bootcamp_Tutorial_Part2.ipynb

NumPy

- **NumPy is an external python package that has very useful mathematical tools**

- import numpy as np
- Numbers like:
  - np.pi, np.e, np.inf
- Trigonometry:
  - np.sin(x)
  - np.rad2deg(np.pi/2)

    >> 90.0

- Random numbers:
  - np.random.random()
    - Random number between 0 and 1
  - np.random.normal(x,s)
    - Sample from Gaussian distribution centered at x with std. dev. s

See Bootcamp_Tutorial_NumPy.ipynb

NumPy Arrays

- Arrays
  - np.mean(array)
  - np.median(array)
- Array Creation
  - np.zeros(10)

    ```
    >> array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
    ```
  - np.linspace(0,10,11)

    ```
    >> array([ 0., 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.])
    ```
  - np.logspace(2,3,6)

    ```
    >> array([ 100. , 158.48931925, 251.18864315,
    398.10717055, 630.95734448, 1000.])
    ```

See Bootcamp_Tutorial_Part3.ipynb

Lists vs. NumPy Arrays

```
list1 = [1,2,3,4]
list2 = [5,6,7,8]
array1 = np.array([1,2,3,4])
array2 = np.array([5,6,7,8])
list1+list2 = [1,2,3,4,5,6,7,8]
array1+ array2 = array([6,8,10,12])
```

→ Addition with lists appends

→ Addition with arrays adds elementwise!

See Bootcamp_Tutorial_NumPy.ipynb

Lists vs. NumPy Arrays

```
list1 = [1,2,3,4]
array1 = np.array([1,2,3,4])
list1**2
  >> TypeError: unsupported operand type(s)
array1**2
  >> array([1,4,9,16])
```

Arrays allow for elementwise manipulation!

See Bootcamp_Tutorial_NumPy.ipynb

Comparisons

Python syntax	Description
==	Check if equal to
!=	Check if not equal to
<	Check if less than
>	Check if greater than
<=	Check if less than or equal to
>=	Check if greater than or equal to
and &	Check if both are true
or |	Check if either are true
not	Check if False

See Bootcamp_Tutorial_Part2.ipynb

Control Flow: Conditional

- **If statements**

```
if condition:
    stuff here
```

- **If-else statements**

```
if condition:
    stuff here
else:
    stuff here
```

- **Elif statements**

```
if condition1:
    stuff here
elif condition2:
    stuff here
elif condition3:
    stuff here
else:
    stuff here
```

See Bootcamp_Tutorial_Part2.ipynb

Control Flow: Loops

- **For loops**

```
for item in list:
    print(item)
```

- **While loops**

```
i = 0
while i<10:
    print(i)
    i = i+1
```

See Bootcamp_Tutorial_Part2.ipynb

Control Flow: Ways to Iterate

```
my_list = ['apple','banana','carrot']
```

- If you want the item:

```
for item in my_list:
    print(item)
>> apple
   banana
   carrot
```

- If you want the number:

```
for i in range(len(my_list)):
    print(i)
>> 0
   1
   2
```

- If you want both:

```
for i, item in enumerate(my_list):
    print(i)
    print(item)
>> 0
   apple
   1
   banana
   2
   carrot
```

See Bootcamp_Tutorial_Part2.ipynb

List Comprehension

Using for loop:

```
list1 = [1,2,3,4,5]
list2 = []
for i in list1:
    list2.append(i**2)
print(list2)
>> [1,4,9,16,25]
```

Using list comprehension:

```
list1 = [1,2,3,4,5]
list2 = [i**2 for i in list1]
print(list2)
>> [1,4,9,16,25]
```

See Bootcamp_Tutorial_Part2.ipynb

Functions

- Skeleton:

```
def fuction_name(x,y):
    stuff here
    return something
```

- Example:

```
def add_numbers(x,y):
    z = x + y
    return z
```

See Bootcamp_Tutorial_Part2.ipynb

File Input and Output

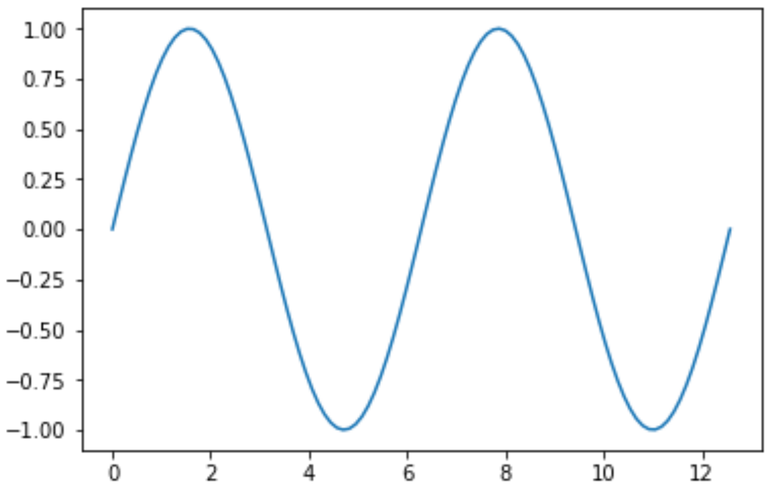
```
f = open("filename.ext",'r+')
for line in f:
        print(line)
f.write("New stuff here \n")
f.close()
```

See Bootcamp_Tutorial_Part1.ipynb

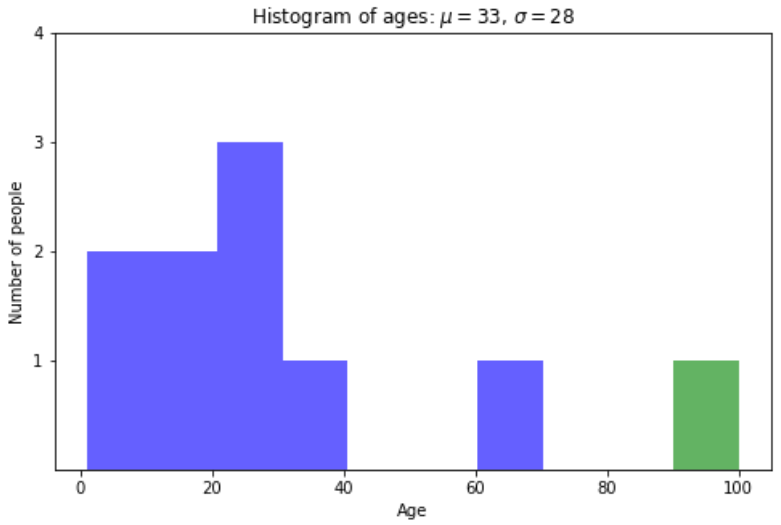
Basic Plotting: plot & hist

```
import matplotlib.pyplot as plt
```

```
plt.plot(xs,ys)
plt.show()
```



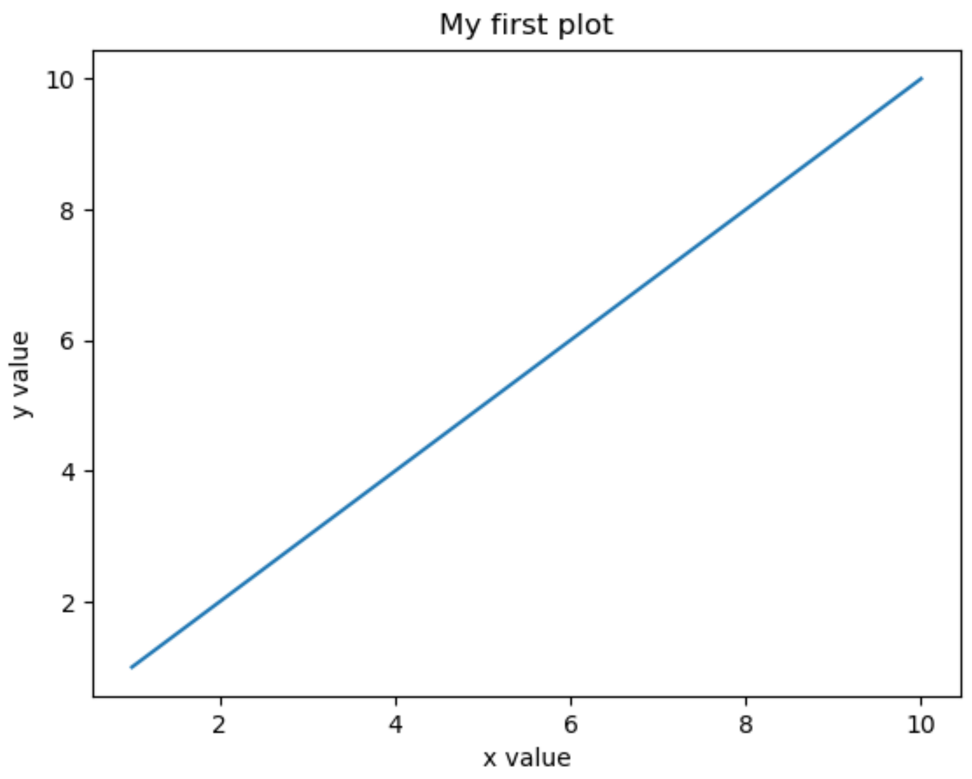
```
plt.hist(xs,ys)
plt.show()
```



See Bootcamp_Tutorial_Part4.ipynb

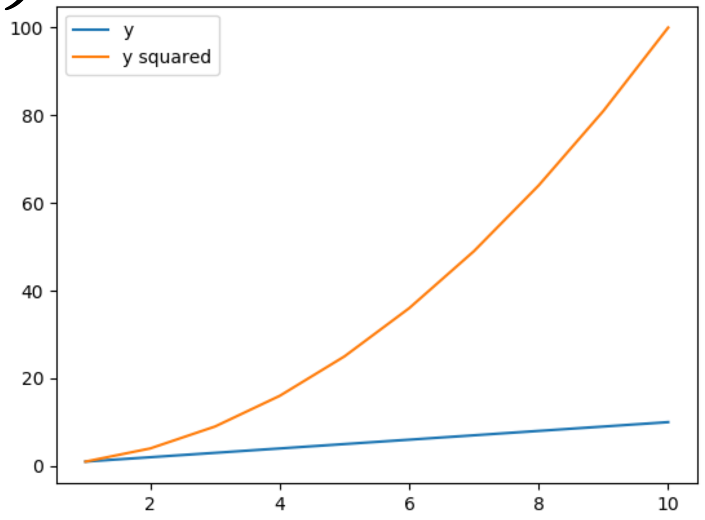
Basic Plotting: Labelling

```
xs = [1,2,3,4,5,6,7,8,9,10]
ys = [1,2,3,4,5,6,7,8,9,10]
plt.plot(xs,ys)
plt.title('My first plot')
plt.xlabel('x value')
plt.ylabel('y value')
plt.show()
```



Basic Plotting: Legends

```
xs = [1,2,3,4,5,6,7,8,9,10]
y1 = [1,2,3,4,5,6,7,8,9,10]
y2 = [1,4,9,16,25,36,49,64,81,100]
plt.plot(xs,y1,label='y')
plt.plot(xs,y2,label='y squared')
plt.legend()
plt.show()
```

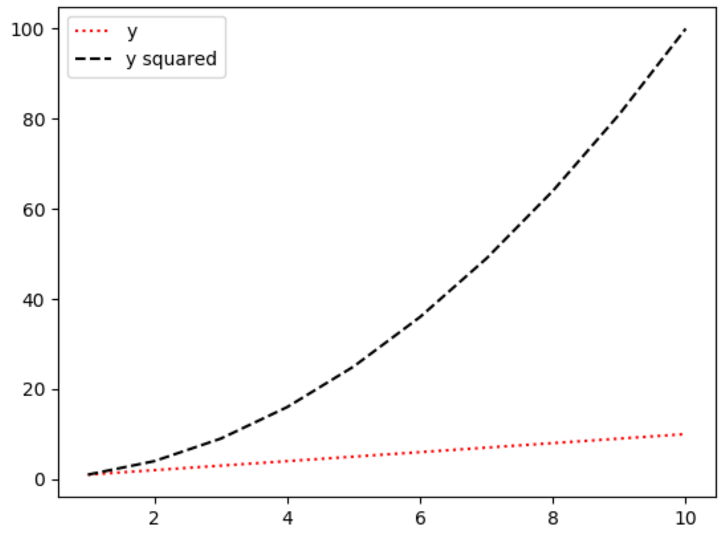


Basic Plotting: Customization

- color
  - ‘r’ = red, ‘b’ = blue, ‘g’ = green, ‘k’ = black
- alpha
  - transparency; decimal from 0 to 1
- linestyle
  - ‘dotted’, ‘dashed’, ‘solid’
- histtype
  - ‘step’, ‘filled’
  - Note: for histograms only

Basic Plotting: Customization

```
xs = [1,2,3,4,5,6,7,8,9,10]
y1 = [1,2,3,4,5,6,7,8,9,10]
y2 = [1,4,9,16,25,36,49,64,81,100]
plt.plot(xs,y1,label='y',color='r',linestyle='dotted')
plt.plot(xs,y2,label='y squared',color='k',
      linestyle='dashed')
plt.legend()
plt.show()
```



Questions?