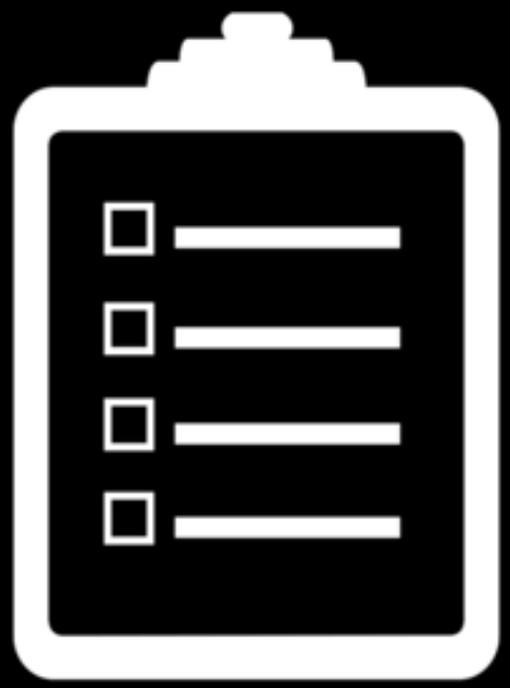
Intro to NumPy, Matplotlib & Pandas

Brought to you by CAD™

Intro to NumPy, ~~Matplotlib & Pandas~~

Brought to you by CAD™

Game Plan



- Intro
- Python List
- NumPy
- Matplotlib (part 2)
- Pandas (part 2)

What are NumPy, Matplotlib and Pandas?

What are NumPy, Matplotlib and Pandas?

- Python libraries!

What are NumPy, Matplotlib and Pandas?

- Applications:
  - Data Science
  - Scientific Computing
  - Data Visualization

Python List

- **my_list = [a , b, c]**
- Powerful data structure -> collection values or variables
- Can contain any type
- Can contain **different** types at once
- …and can even contain lists

Question?

- *Which **data types** does this list contain?*
  x = ["you", 2, "are", "so", True]

A. int

B. float

C. str

D. bool

E. list

Python List

- **Easy** element access:
  - Index operator —> **[ ]**
  - List slicing —> **[start : end]**

Inclusive **Exclusive**

Question?

- *What are two ways to obtain the **last two** elements of the the officer list?. List size is 13*

A. officers[: 2]

B. officers[11:]

C. officers[‘last two’]

D. officers[-2:]

E. officers[-2]

Python List

- **Easy** to change elements:
  - officers**[-2] =** ‘Juan Trejo’
  - Concatenation —> list **+** list
  - Adding —> officers **+=** [‘New Guy’]
  - Deleting —> **del(** officers[-1] **)**

Python List

- Python lists are ***powerful***. However they have ***limitations***

What if we want to analyze our collection of data? Can we perform calculations on it?

Answer...

- No… But why?

```
time     = [5.2, 5.0, 5.5, 4.8, 5.0, 5.3] #seconds
distance = [1.5, 2.5, 0.5, 1.0, 2.3, 2.9] #meters
```

```
# find speed (m/s) -> speed = distance/time
speed    = distance / time
```

```
TypeError: unsupported operand type(s) for /:
'list' and 'list'
```

NumPy

- Numeric Python
- Alternative to Python list: NumPy Array
- Calculations over entire arrays
- Easy and fast

NumPy

- Back to our example

```
time     = [5.2, 5.0, 5.5, 4.8, 5.0, 5.3] #seconds
distance = [1.5, 2.5, 0.5, 1.0, 2.3, 2.9] #meters
```

```
import numpy as np

np_time     = np.array(time)
np_distance = np.array(distance)

# find speed (m/s) -> speed = distance/time
speed = np_distance / np_time
```

NumPy

- NumPy arrays their own Python type
- Can only contain ***one*** type - *What happens if a list has multiple?*
- Behave ***differently*** from list

Question?

- *How would you add two NumPy arrays? How would you concatenate them?*

```
l1 = [1, 2, 3, 4, 5]
l2 = [1, 1, 1, 1, 1]

np_l1 = np.array(l1)
np_l2 = np.array(l2)

np_l1 + np_l2  # [2, 3, 4, 5, 6]
np.concatenate((np_l1, np_l2), 0) # [1, 2, 3, 4, 5, 1, 1, 1, 1, 1]
```

NumPy

- NumPy supports ***n dimensional*** arrays - *Check it's type!*
- 2D arrays can be created from a *list of lists* but allow fast calculations and subsetting

NumPy

```
                    0      1      2      3      4
np_2d = np.array([[1.73, 1.68, 1.71, 1.80, 1.79], 0
                  [65.4, 59.2, 63.6, 88.4, 68.7]]) 1

np_2d[0]         # first row
>> array([1.73, 1.68, 1.71, 1.80, 1.79])
np_2d[0][2]     # element in row = 0, col = 2
>> 1.71
np_2d[0,2]      # same as above
>> 1.71
np_2d[:,1:3]    # what do you think?
>> array([[  1.68,    1.71],
          [ 59.2 ,   63.6 ]])
```

Question?

- *What is the result of the following? Try it!*

```
import numpy as np
x = np.array([[1, 2, 3],
              [1, 2, 3]])
y = np.array([[1, 1, 1],
              [1, 2, 3]])
z = x * y
```

NumPy

- NumPy is a great tool for ***data analysis***
- Useful methods:
  - mean( )
  - median( )
  - std( )
  - sum( ), sort( ) —> available for list but faster!

NumPy

- **Summary:**
  - NumPy is a great alternative to the regular Python list if you want to do **Data Science** in Python.
  - NumPy arrays can only hold elements of the **same basic type**.
  - Next to an efficient data structure, NumPy also offers tools to calculate summary **statistics** and to simulate statistical distributions.