Administrivia

- Homework 9 due Friday
- Midterm 2 April 14th at 7pm
REVIEW
TypeError: object of type 'int' has no len()

What code produces this error?

a) a=[1,3,5]
   b=len(a[1])

b) a="ABC"
   b=a+1

c) a=[2,4,6]
   b=(a[2,a[3]])
What error will this code produce?

a) `TypeError: int()` argument must be a string or a number, not 'tuple'
b) `IndexError: tuple index out of range`
c) `AttributeError: 'tuple' object has no attribute 'append'`
d) `TypeError: 'int' object is not iterable`
x = []
for c in "ABCDEFEG":
    if c < "D":  
        continue 
    x.append(c)

a) ["A","B","C","D","E","F","G"]
b) ["A","B","C"]
c) ["D","E","F","G"]
d) []
```python
a={"D":2,"O":5,"G":3}
for k in "DOGGY":
    print(a[k])
```

What will this code output before it crashes?

a) “D” “O” “G” and “G”
b) 2 5 3 and 3
c) None None None and None
d) Nothing at all
Course Summary (so far...)  
1. Python fundamentals  
2. Data wrangling  
3. Data visualization  
4. Simulation  
5. Random processes  
6. Optimization
NUMPY
Numpy

- Module for Python to extend its numerical capabilities
- Designed for more efficient computation
- Designed for manipulating arrays and matrices

import numpy as np
Arrays

• Numpy arrays are similar to lists:
  – Represent a collection of items
  – Can be indexed

• Numpy arrays are different than lists:
  – Fixed size
  – All elements have the same type
  – Can do operations on all elements
Arrays

- Can be created from lists

```python
np.array([1, 2, 3])
```
x=np.array([1]*2)
x+=1

What is the final value of x?
a) array([2])
b) array([1,1,1])
c) array([2,2])
d) array([3])
Data type

• Many possible types in numpy
  – Boolean
  – integers (8, 16, 32, 64 bits)
  – floats (16, 32, and 64 bits)
  – complex (64 and 128 bits)

```python
a=[3,2,4]
x=np.array(a,dtype=np.float64)
x.dtype
```
arange

• Returns array over a range (like list range)
  – Argument 1: Start
  – Argument 2: End
  – Argument 3: Step size

\[ x = \text{np.arange}(10, 25, 5.0) \]

\[ \text{len}(x) \]
linspace

• Returns array of evenly spaced values
  – Argument 1: start of range
  – Argument 2: end of range
  – Argument 3: number of points in range

```python
x = np.linspace(0, 1, 100)
y = x**2
plt.plot(x, y, 'g--')
```
zeros

• Returns array of zeros
  – Argument 1: the number of zeros

```python
x = np.zeros(100)
x.dtype
x.size
```
Why use numpy?

• Extremely powerful!

```python
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
plt.plot(x, y, 'g--')
```
ND Arrays

- Arrays can be *multidimensional*
- Let’s make a 3x2 array
  - 2 dimensional array
  - 3 rows, 2 columns

```python
a=[[1,2],[3,4],[5,6]] # List of # lists!
b=np.array(a)
```
What will produce this array?
a) np.array([[1,2,3],[1,2,3]])
b) np.array([[2,3]])
c) np.array([[3,2]])
d) np.array([[1,1],[2,2],[3,3]])
2D Arrays

<table>
<thead>
<tr>
<th>3 rows</th>
<th>4 columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
</tr>
</tbody>
</table>
2D indexing

• We must specify both the row \textit{and} column number to retrieve an element
• Row is first, then column:
\texttt{array[row][colmn]}
2D Arrays

\[
a[1][2]
\]

\[
\begin{array}{cccc}
5 & 2 & 1 & 2 \\
4 & 3 & 2 & 1 \\
9 & 4 & 3 & 8 \\
\end{array}
\]
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

How can we index 5?

a) `a[1][2]`

b) `a[2][1]`

c) `a[1][1]`

d) `a[2][2]`
zeros

• Returns array of zeros
  – Argument 1: a tuple/list of dimensions

```python
x = np.zeros(((10, 10))
x.shape
```
Looping over 2D arrays

```python
x=np.zeros((3,3))
for i in range(3):
    print x[i]
```
Looping over 2D arrays

```python
x=np.zeros((3,3))
for i in range(3):
    x[i][0]=1
    x[i][1]=2
    x[i][2]=3
print x
```
Looping over 2D arrays

```python
x = np.zeros((3, 3))
for i in range(3):  # for each row
    x[i][0] = 1
    x[i][1] = 2  # columns in the row
    x[i][2] = 3
print(x)
```
Looping over 2D arrays

\[
x[i][0] = 1 \\
x[i][1] = 2 \\
x[i][2] = 3
\]

for \( j \) in range(3):

\[
x[i][j] = j + 1
\]
Looping over 2D arrays

```python
x = np.zeros((3, 3))
for i in range(3):
    x[i][0] = 1
    x[i][1] = 2
    x[i][2] = 3
for j in range(3):
    x[i][j] = j + 1
```
Looping over 2D arrays

```python
x=np.zeros((3,3))
for i in range(3):
    for j in range(3):
        x[i][j]=j+1
```
\[
\begin{array}{|c|c|c|c|}
\hline
5 & 2 & 1 & 2 \\
4 & 3 & 2 & 1 \\
9 & 4 & 3 & 8 \\
\hline
\end{array}
\]

\(m=3\)

\(n=4\)
for i in range(m):
    for j in range(n):
        x[i][j] = 0

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>
for i in range(m):
    for j in range(n):
        x[i][j]=0

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>2</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
for i in range(m):
    for j in range(n):
        x[i][j]=0
for i in range(m):
    for j in range(n):
        x[i][j]=0
for i in range(m):
    for j in range(n):
        x[i][j]=0

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>2</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

i      j
0      0
for i in range(m):
    for j in range(n):
        x[i][j] = 0
for i in range(m):
    for j in range(n):
        x[i][j] = 0
for i in range(m):
    for j in range(n):
        x[i][j]=0
for i in range(m):
    for j in range(n):
        x[i][j] = 0
for i in range(m):
    for j in range(n):
        x[i][j]=0
for i in range(m):
    for j in range(n):
        x[i][j]=0
for i in range(m):
    for j in range(n):
        x[i][j]=0

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
for i in range(m):
    for j in range(n):
        x[i][j] = 0
for i in range(m):
  for j in range(n):
    x[i][j]=0
for $i$ in range($m$):
    for $j$ in range($n$):
        $x[i][j]=0$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

$i$ $j$
---
1 3
```python
for i in range(m):
    for j in range(n):
        x[i][j]=0
```
for i in range(m):
    for j in range(n):
        x[i][j] = 0

\[
\begin{array}{cccc}
  0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 \\
  9 & 4 & 3 & 8 \\
\end{array}
\]
for i in range(m):
    for j in range(n):
        x[i][j]=0

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

i: 2
j: 0
for i in range(m):
    for j in range(n):
        x[i][j]=0
for i in range(m):
    for j in range(n):
        x[i][j] = 0
for i in range(m):
    for j in range(n):
        x[i][j] = 0
for i in range(m):
    for j in range(n):
        x[i][j]=0
```python
x = np.zeros((3, 3))
for i in range(3):
    for j in range(3):
        x[i][j] = i
```

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
```python
x = np.zeros((3, 3))
for i in range(3):
    for j in range(3):
        x[i][j] = j
```

A:

```
0 0 0
1 1 1
2 2 2
```  

B:

```
0 1 2
0 1 2
0 1 2
```  

C:

```
0 1 2
1 1 2
2 2 2
```
```python
x=np.zeros((3,3))
for i in range(3):
    for j in range(3):
        x[i][j]=i+j
```
BASIC SIMULATION
Example

• A kitten knocks a cup off of a 1-meter high table. How long until it hits the ground?
• $g = -9.8 \text{m/s}^2$
• $v_0 = 0 \text{m/s}, y_0 = 1 \text{m}$
• $v_{t+1} = v_t + g \Delta t$
• $y_{t+1} = y_t + v_t \Delta t$
• $\Delta t = ?$
import numpy as np

# Parameters of simulation
n=100           # number of data points to plot
start=0.0       # start time of simulation
end=1.0         # ending time of simulation
g=-9.8          # acceleration of gravity

# State variable initialization

t=np.linspace(start,end,n+1)    # time in seconds
y=np.zeros(n+1)                 # height in meters
v=np.zeros(n+1)                 # velocity in m/s (v0=0m/s)
y[0]=1.0                        # initialize height to 1m

for i in range(1,n+1):
    v[i]=v[i-1]+g*(t[i]-t[i-1])
    y[i]=y[i-1]+v[i-1]*(t[i]-t[i-1])
    if y[i] <= 0: # glass has hit the ground
        v[i]=0
        y[i]=0