Administrivia

• Homework 8 due on Friday
REVIEW
TypeError: Can't convert 'int' object to str implicitly

What code produces this error?

a) 1 + (1,2)

b) "3" + len("3")

c) 5 + ''.join([0,3])
TypeError: can only concatenate tuple (not "int") to tuple

What code produces this error?

a) for i in zip("ABC"): 
   print(i[0])

b) for i in range("ABC"): 
   print(i+1)

c) for i in enumerate("ABC"): 
   print(i+1)
Course Summary (so far…)

1. Python fundamentals
2. Data wrangling
3. Data visualization
4. Simulation
5. Random processes
6. Optimization
import csv
in_file=open("file.csv")
reader=csv.DictReader(in_file)
for row in reader:
    print(row["column1"])
    print(row["column2"])

DictReader
Continue

- Skips to the next iteration of a loop
- Useful for filtering out unwanted data

```python
for i in range(100):
    if i % 2 == 0:
        continue

print(i)
```
x=[]
for i in range(100):
    if i < 95:
        continue
    x.append(i)

a) [95, 96, 97, 98, 99, 100]
b) [96, 97, 98, 99]
c) [95, 96, 97, 98, 99]
d) [96, 97, 98, 99, 100]
EXCEPTIONS
Exceptions

• Represent computation reaching an exceptional (unexpected or unusual) state
• Exceptions are “thrown” when we reach the state

print(x)

• If exception is not caught (or handled) Python will print a trace
  – list of lines of code that were running
Handling Exceptions

- Exceptions can be caught using the *try*/except* structure

```python
try:
    a=[1,2]
    print(a[2])
except:
    print("Oh no! An error!")
```
Throwing Exceptions

• Exceptions can be thrown with the “raise” structure

raise Exception("Don’t do that!")
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

```python
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
```python
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
```
array

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

```python
x = np.arange(10, 25, 5.0)
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--')
```
Arrays

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

\[ a = \begin{bmatrix} [1,2], [3,4], [5,6] \end{bmatrix} \] # List of lists!

\[ b = \text{np.array}(a) \]
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

<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>
2D indexing

• We must specify both the row *and* column number to retrieve an element
• Row is first, then column:
  \[ a[r][c] \]
2D Arrays

```plaintext
a[1][2]
```

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
```

The element `a[1][2]` corresponds to the number 2.
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

```python
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
```

```python
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
```
<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

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<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>
```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
```python
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>0</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>

```
$i$

```

```
$j$

```

```
0
```

```
0
```
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>0</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>

$i$  $j$
0    1
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

<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>2</td>
<td></td>
</tr>
<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

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
<tr>
<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>
</tr>
</tbody>
</table>

\(i\) \(j\)
\(l\) \(3\)
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>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>
</tr>
<tr>
<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>
</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

<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>0</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

i j
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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

i  j
2  1
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>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>8</td>
</tr>
</tbody>
</table>

i | j
---|---
2 | 2
for i in range(m):
    for j in range(n):
        x[i][j]=0

```
 0  0  0  0  0
 0  0  0  0  0
 0  0  0  0  0
 0  0  0  0  0
```

i

```
 2  3
```

j
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
```
```python
x = np.zeros((3, 3))
for i in range(3):
    for j in range(3):
        x[i][j] = j
```
```python
x = np.zeros((3, 3))
for i in range(3):
    for j in range(3):
        x[i][j] = i + j
```

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
```python
a={"D":2,"O":5,"G":3}
for k in "DOGGY":
    print a[k]
```

What error will this code produce?

a) SyntaxError: invalid syntax
b) KeyError: 'Y'
c) TypeError: list indices must be integers, not str
d) There is no error.
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 3
c) None None None None

d) Nothing at all
```python
x=[]
for c in "ABCDEFG":
    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) []
How can we index 5?

a) a[1][2]
b) a[2][0]
c) a[1][1]
d) a[2][2]