CS101: Intro to Computing Fall 2015

Lecture 17

Administrivia

- Homework 11 due on today
- Homework 12 released today
  - Data manipulation and visualization
  - Simple simulation with numpy
- Midterm 2: November 16th

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])
```

```
a=1,2,3
b=a[3].append(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) `TypeError: object of type 'int' has no len()`

d) `TypeError: 'int' object is not iterable`

HOMEWORK 11/12 HELP

Coding tips

1. **START EARLY!**
2. Break the problem down and write functions.
3. Document your functions ***before*** writing them.
4. Test every function (even every ***line***) you write to make sure it works (write then execute, write then execute).

Don’t just stare at your code!

- Make the ***computer*** work, not you!
- Add print statements (especially in loops!)

Data! Data!
Data!
I can't make
bricks without
clay.

Debugging Tips

1. Comment out lines to narrow down your search for the problem.
2. Add print statements.
3. Trace troublesome input through your code line by line.
4. Explain the problem to someone else.
5. Take a break.

Ultimate Coding Secret

- If you're confused, frustrated, and can't make progress…

Always approach a case with an absolutely blank mind.

- … start over.

DictReader

```
import csv
in_file=open(”file.csv")
reader=csv.DictReader(in_file)
for row in reader:
    print row["column1"]
    print row["column2"]
```

Continue

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

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

Continue

```
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]

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

```
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)

```
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=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
  - Arguemnt 3: number of points in range

```
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

```
x=np.zeros(100)
x.dtype
x.size
```

Example

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

Why use numpy?

- Extremely powerful!

```
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=[[1,2],[3,4],[5,6]] # List of
                      # lists!

b=np.array(a)
```

1	**1**
2	**2**
3	**3**

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

4 columns

3 rows

5	**2**	**1**	**2**
4	**3**	**2**	**1**
9	**4**	**3**	**8**

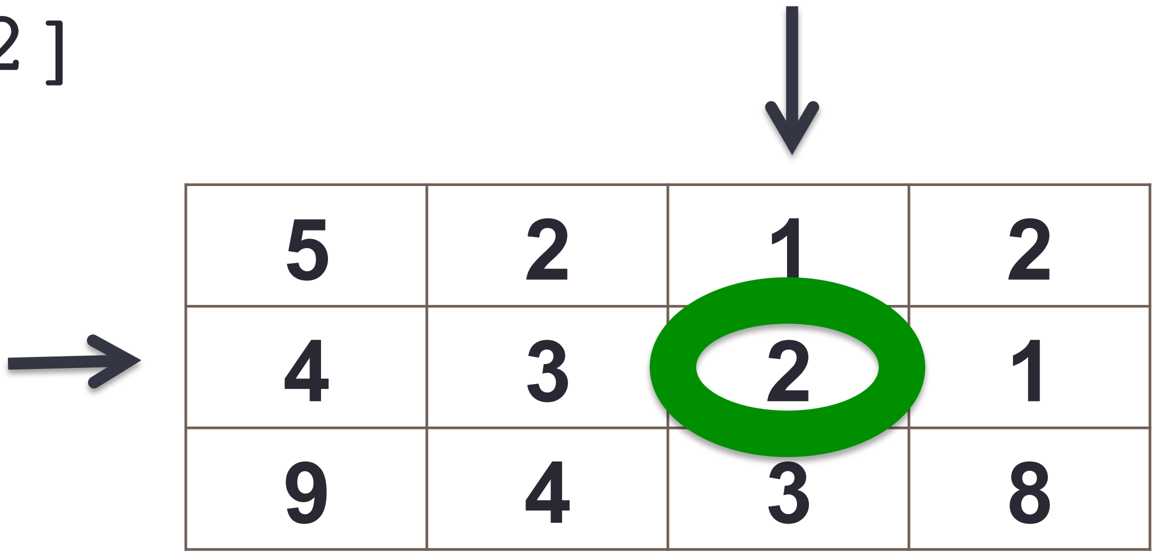
2D indexing

- We must specify both the row *and* column number to retrieve an element
- Row is first, then column:

```
a[r][c]
```

2D Arrays

`a[1][2]`



5	2	1	2
4	3	2	1
9	4	3	8

1	4
2	5
3	6

How can we index 5?

a) `a[1][2]`

b) `a[2][1]`

c) `a[1][1]`

d) `a[2][2]`

Example

- 20 kittens knock 20 cups off of a series of tables at 1-meter intervals. How long until they hit the ground?
- $g=-9.8m/s^2$
- $v_0=0m/s$, $y_0=1m$
- $v_{t+1}=v_t + g*\Delta t$
- $y_{t+1}=y_t+v_t*\Delta t$
- $\Delta t=?$

zeros

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

```
x=np.zeros((10,10))
x.shape
```