CS101: Intro to Computing
Fall 2015

Lecture 18

Administrivia

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

REVIEW

```
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

```
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) []

NUMPY AND 2D ARRAYS

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	1
2	2	2

What will produce this array?

a) `np.array([[1,1,1],[2,2,2]])`

b) `np.array([[1,2],[1,2],[1,2]])`

2D Arrays

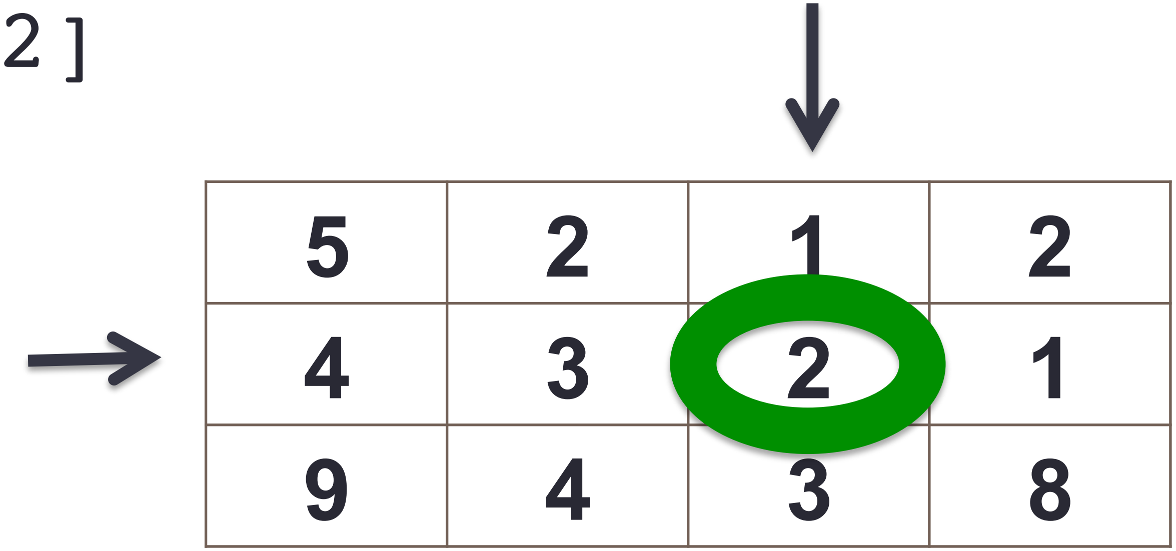
4 columns

3 rows

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

2D Indexing

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



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

1	2
3	4
5	6

How can we index 5?

a) `a[1][2]`

b) `a[2][0]`

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
```

Looping over 2D arrays

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

Looping over 2D arrays

```
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

```
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

```
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

```
x=np.zeros((3,3))
for i in range(3):
  for j in range(3):
    x[i][j]=j+1
```

$$
m=3 \left\{ \overbrace{\begin{array}{|c|c|c|c|} \hline 5 & 2 & 1 & 2 \\ \hline 4 & 3 & 2 & 1 \\ \hline 9 & 4 & 3 & 8 \\ \hline \end{array}}^{n=4} \right.
$$

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

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

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

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

i

0

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

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

i	j
0	0

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

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

i	j
0	0

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

0	2	1	2
4	3	2	1
9	4	3	8

i	j
0	0

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

0	2	1	2
4	3	2	1
9	4	3	8

i	j
0	1

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

0	0	1	2
4	3	2	1
9	4	3	8

i	j
0	1

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

0	0	0	2
4	3	2	1
9	4	3	8

i	j
0	2

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

0	0	0	0
4	3	2	1
9	4	3	8

i	j
0	3

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

0	0	0	0
4	3	2	1
9	4	3	8

i

1

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

0	0	0	0
4	3	2	1
9	4	3	8

i	j
1	0

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

0	0	0	0
0	3	2	1
9	4	3	8

i	j
1	0

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

0	0	0	0
0	0	2	1
9	4	3	8

i	j
1	1

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

0	0	0	0
0	0	0	1
9	4	3	8

i	j
1	2

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

0	0	0	0
0	0	0	0
9	4	3	8

i	j
1	3

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

0	0	0	0
0	0	0	0
9	4	3	8

i

2

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

0	0	0	0
0	0	0	0
9	4	3	8

i	j
2	0

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

0	0	0	0
0	0	0	0
0	4	3	8

i	j
2	0

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

0	0	0	0
0	0	0	0
0	0	3	8

i	j
2	1

```
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	8

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

i	j
2	3

```
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

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

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

```
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

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

A

0	1	2
1	2	3
2	3	4

B

0	1	2
1	2	3
2	3	4

C

0	1	2
2	3	4
4	5	6

Example

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

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

Handling Exceptions

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

```
try:
  a=[1,2]
  print a[2]
except:
  print “Oh no!”
```

Throwing Exceptions

- Exceptions can be thrown with the “raise” structure

```
raise Exception(“Oh no!”)
```