# CS101: Intro to Computing
# Fall 2015

Lecture 20

# Administrivia

- Homework 13 released today
- Midterm 2: November 16th

# REVIEW

```
x=np.zeros((2,2))
y=np.array([[1,2],[3,4]])
for i in range(2):
  for j in range(2):
      x[i][j]=y[j][i]
```

A

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

B

| | |
|---|---|
| 1 | 3 |
| 2 | 4 |

C

| | |
|---|---|
| 1 | 3 |
| 4 | 2 |

```
for i in range(5):
   try:
      print 5.0/i
   except:
      print i
```

a) 0 5 2.5 1.67 1.25 1

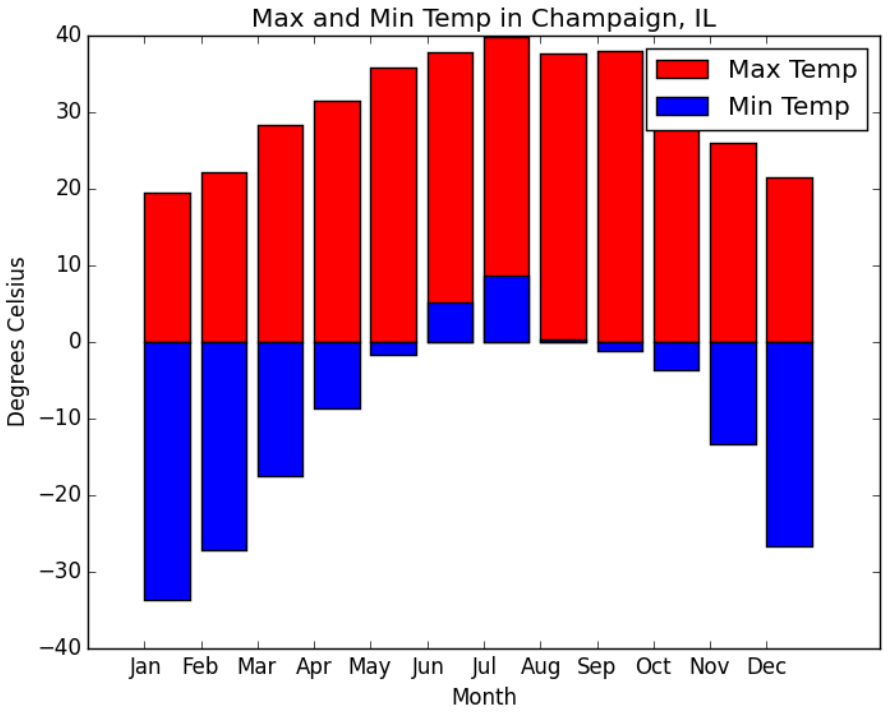
b) 5 2.5 1.67 1.25 1

c) 0 5 2.5 1.67 1.25

d) Nothing at all

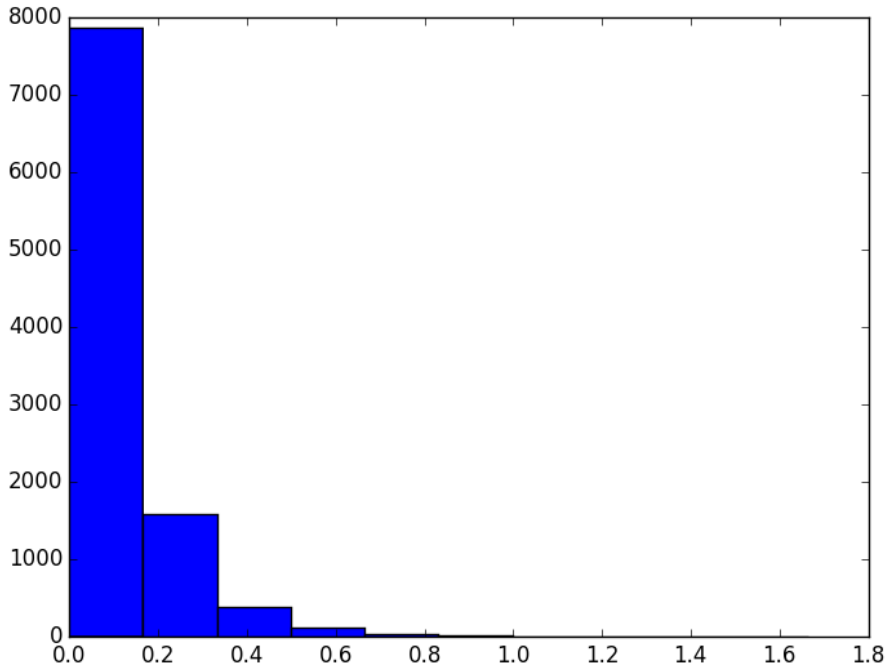
# Bar Graph

```
plt.bar(x,y)
```



# Histogram

```
plt.hist(x)
```



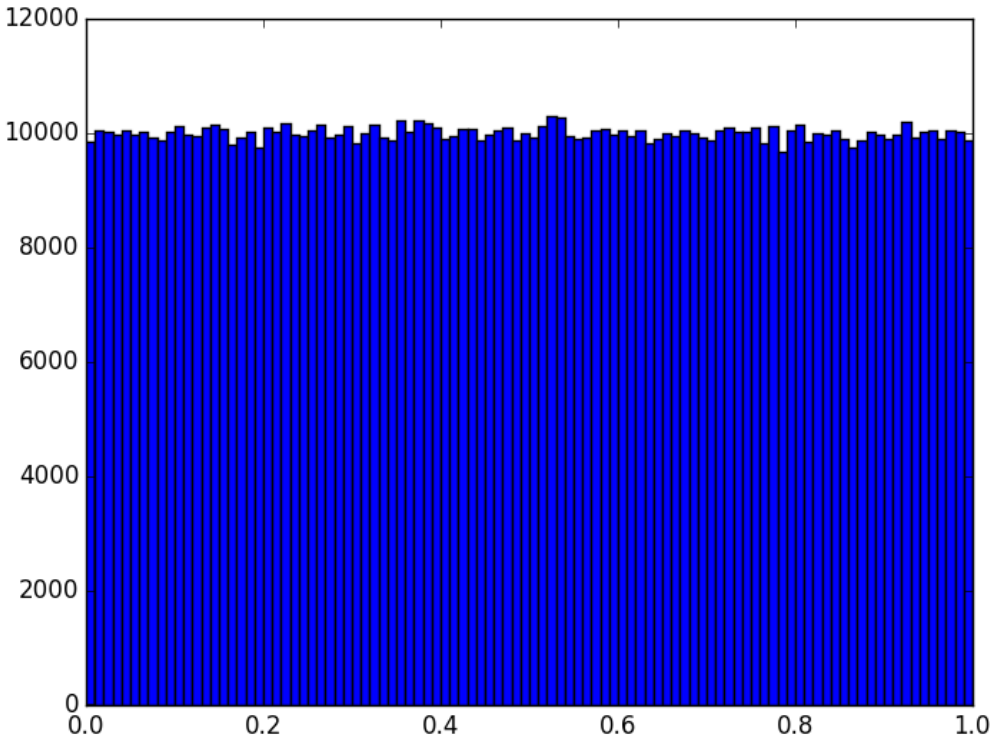
# PSEUDORANDOM NUMBERS

# Pseudorandom numbers

- PRNG - a program that generates *seemingly* random numbers
- Starting from an initial input (seed) PRNG generates “unpredictable” numbers

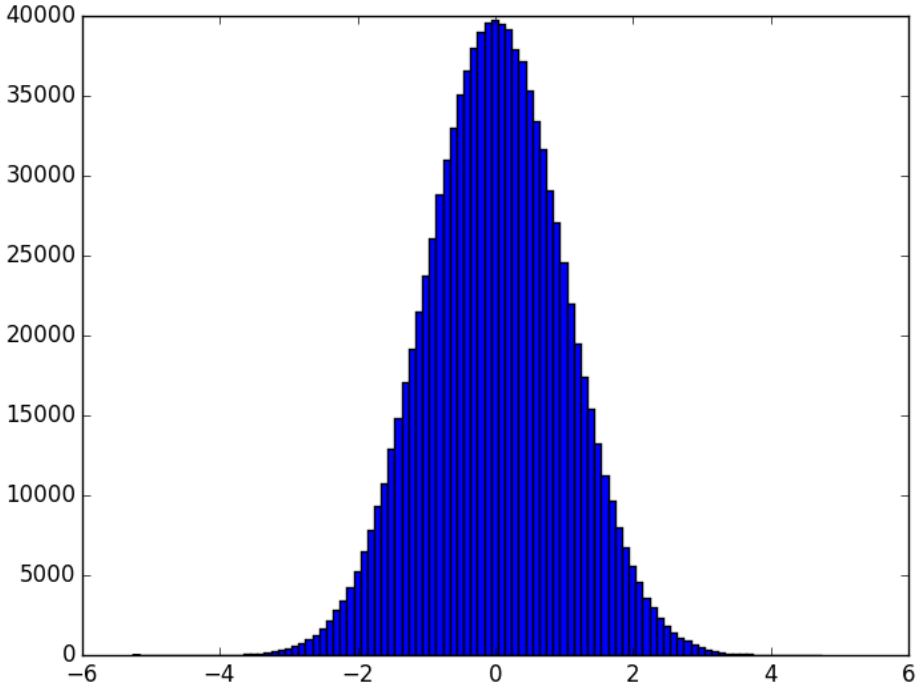
# Rand

- Creates an array sampled from **uniform distribution** on interval [0,1)



# Randn

- Creates an array sampled from **standard normal distribution**



# Choice

- Generates a random sample from a 1-D array

```
x=np.arange(1,7)
c=np.random.choice(x)
```

# Choice

- Can generate samples without replacement

```
x=np.arange(1,53)
c=np.random.choice(
   x,size=5,
   replace=False)
```

# Shuffle

- Randomly reposition elements of a 1-D array
- Creates a random permutation of the array elements

```
x=np.arange(1,53)
np.random.shuffle(x)
```

# Descriptive Statistics

- Can compute basic descriptive statistics like:
  - Min, max, percentile
  - Mean, median, standard deviation

```
x=np.random.rand(1000)
print np.median(x)
print np.mean(x)
```

# Example

- A random walk in 2 dimensions
  - Start in middle
  - Take one step in a random direction: north, south, east, or west

# Matshow

- Pyplot function to display a 2d array as an image

# OPTIMIZATION

# Optimization

- Given a function f(x), find the x such that f(x) is maximized/minimized
- Goal: search through the domain for the optimum x
- Many clever ways to do this, but let's start with something naïve

# Brute force search

- Search through the entire domain of f
- Also called an “exhaustive search”

# Example

On vacation, you found n items of varying weight and value.

Your bag has a weight limit of 50 pounds.

What is the best set of items to take on the flight?

# Random search

- We can also randomly sample the domain of f
- Might not find the *true* optimum, but could find one that is “good enough”
- How could we sample from the domain of the previous problem?