Numerical Python

optimization

CS101 Lecture #19

2016-12-05

Administrivia

Administrivia

- Homework #9 is due Friday, Dec. 9.
- Homework #10 is due Tuesday, Dec. 20.
- Midterm #2 is Monday, Dec. 19 from 7–10 p.m.

Warmup Question

Question #1

```
def fact( n ):
    if n <= 1:
        return 1
    else:
        ???
```

Which line of code correctly makes `fact` return the factorial $n!$?

- A `return fact( n - 1 ) * fact( n )`
- B `return fact( n - 1 ) * n`
- C `return ( n - 1 ) * n`
- D `return fact( n - 2 ) * n`

Question #1

```
def fact( n ):
    if n <= 1:
        return 1
    else:
        ???
```

Which line of code correctly makes `fact` return the factorial $n!$?

- A `return fact( n - 1 ) * fact( n )`
- B `return fact( n - 1 ) * n` ⋆
- C `return ( n - 1 ) * n`
- D `return fact( n - 2 ) * n`

Randomness Refresher

Randomness refresher

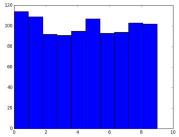
- `randint( start,end,size=tuple )`
- `uniform( start,end,size=tuple )`
- `randn( d0,d1,d2,... )`
- Note that the interfaces for each are slightly different.

Question #2

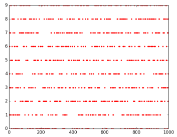
```
x = np.random.randint( 0,10, size=(1000,1) )
plt.hist( x )
plt.show()
```

What is a possible output of this code?

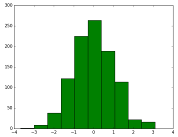
A



B



C

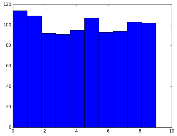


Question #2

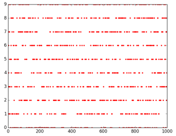
```
x = np.random.randint( 0,10, size=(1000,1) )
plt.hist( x )
plt.show()
```

What is a possible output of this code?

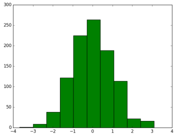
[✓ A]



B



C



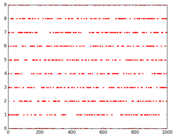
★

Question #3

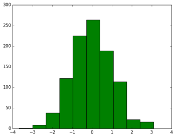
```
x = np.random.uniform( size=(1000,1) )
plt.plot( x, 'c.' )
plt.ylim( (-1,2) )
plt.show()
```

What is a possible output of this code?

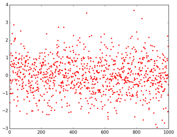
A



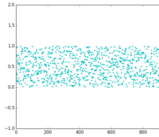
B



C



D

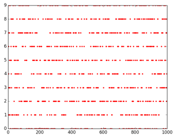


Question #3

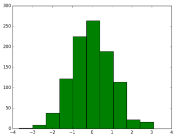
```
x = np.random.uniform( size=(1000,1) )
plt.plot( x, 'c.' )
plt.ylim( (-1,2) )
plt.show()
```

What is a possible output of this code?

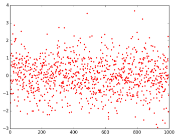
A



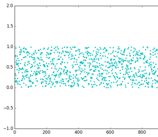
B



C



D



★

http://matplotlib.org/api/colors_api.html

Optimization

Optimization

On vacation, you purchase a range of n souvenirs of varying weight and value. When it comes time to pack, you find that your bag has a weight limit of 50 pounds. What is the best set of items to take on the flight?

Optimization

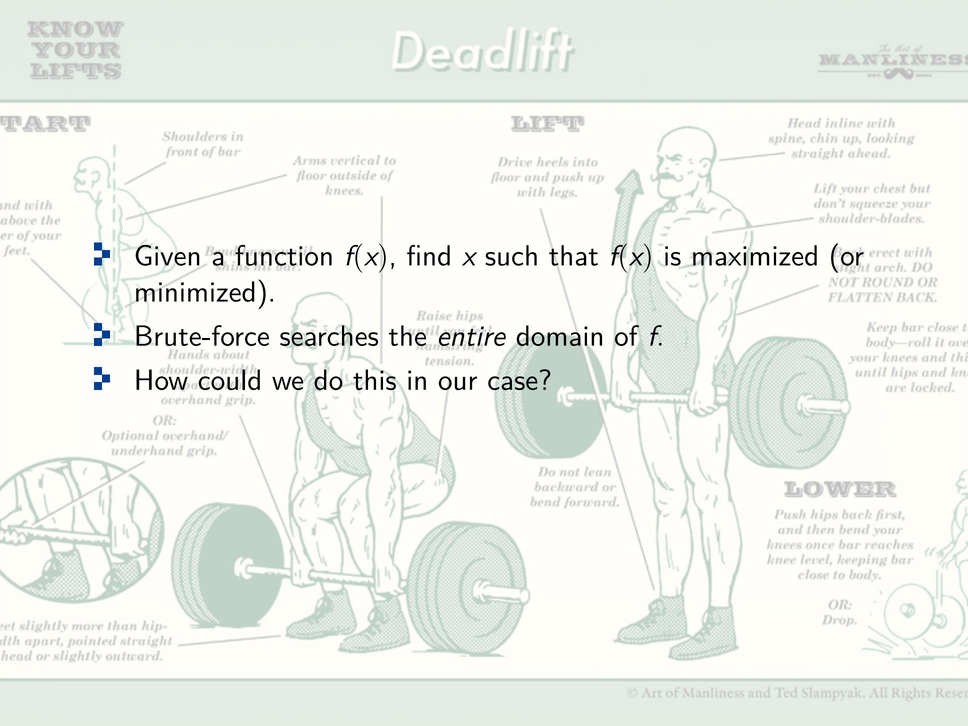
- Given a function $f(x)$, find x such that $f(x)$ is maximized (or minimized).
- The goal is to search the domain for the optimal x yielding the optimal $f(x)$.
- Many clever techniques exist, but we'll start with a naïve approach.

Setup

```
import numpy as np

n = 10
items   = list( range( n ) )
weights = np.random.uniform( size=(n,1) ) * 50
values  = np.random.uniform( size=(n,1) ) * 10(
```

Setup

```
def f( wts, vals ):
    total_weight = 0
    total_value = 0

    for i in range( len( wts ) ):
        total_weight += wts[ i ]
        total_value  += vals[ i ]

    if total_weight >= 50:
        return 0
    else:
        return total_value
```


KNOW YOUR LIFTS
Deadlift
The Art of MANLINESS
START
Shoulders in front of bar
Arms vertical to floor outside of knees.
above the
feet.
Hands about shoulder-width
overhand grip.
OR: Optional overhand/ underhand grip.
Raise hips
tension.
LIFT
Drive heels into floor and push up with legs.
Head inline with spine, chin up, looking straight ahead.
Lift your chest but don't squeeze your shoulder-blades.
NOT ROUND OR FLATTEN BACK.
Keep bar close
are locked.
Do not lean backward or bend forward.
LOWER
Push hips back first, and then bend your knees once bar reaches knee level, keeping bar close to body.
OR: Drop.
© Art of Manliness and Ted Slampyak. All Rights

- Given a function $f(x)$, find x such that $f(x)$ is maximized (or minimized).
- Brute-force searches the *entire* domain of f.
- How could we do this in our case?

Optimization

- Two useful functions from `itertools` to keep in mind:
  - `combinations`: provide all subsets of size `n`.
  - `product`: replace nested `for` loops.

Optimization

- `combinations`: provide all subsets of size `n`.

```
import itertools

a = [ 1,2,3,4 ]
for x in itertools.combinations( a,2 ):
    print( x )
```

Optimization

- `product`: replace nested `for` loops.
- Can use `repeat=n` argument as well.

```
import itertools

a = [ 1,2,3,4 ]
b = [ 'g','h','i' ]
for x in itertools.product( a,b ):
    print( x )
for x in itertools.product( a, repeat=3 ):
    print( x )
```

Question #4

```
x = 'ABCD'
z = 'XYZ'

for a in itertools.product( x,y ):
    print( ' '.join( a ) )
```

Which of the following is *not* printed?

A `'A X'`

B `'B D'`

C `'C X'`

D `'D Z'`

Question #4

```
x = 'ABCD'
z = 'XYZ'

for a in itertools.product( x,y ):
    print( ' '.join( a ) )
```

Which of the following is *not* printed?

A `'A X'`

B `'B D'` ⋆

C `'C X'`

D `'D Z'`

Setup

```
import itertools

max_value = 0.0
max_set = None
for i in range(n):
    for set in itertools.combinations( items,i ):
        wts  = []
        vals = []
        for item in set:
            wts.append( weights[ item ] )
            vals.append( values[ item ] )
        value = f( wts,vals )
        if value > max_value:
            max_value = value
            max_set = set
```

Optimization

- Brute-force search of a password:

```
def check_password( pwd ):
    if pwd == 'pas':
        return True
    else:
        return False

chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvw
for pair in itertools.product( chars, repeat=3 ):
    pair = ''.join( pair )
    if check_password( pair ):
        print( pair )
```

Optimization

- Brute-force search of a password:

$$\begin{aligned} & 2 \times n(\text{alphabet}) + n(\text{digits}) + n(\text{special}) \\ = \; & 2 \times 26 + 10 + \{24\text{–}32\} \\ = \; & \{86\text{–}94\} \end{aligned}$$

per letter! This gets very big very quickly!

Administrivia

- Homework #9 is due Friday, Dec. 9.
- Homework #10 is due Tuesday, Dec. 20.
- Midterm #2 is Monday, Dec. 19 from 7–10 p.m.