Welcome to INFO 206B

Understand and harness the power of data
Welcome to INFO 206B

Understand and harness the power of data

data structures
data analytics
Welcome to INFO 206B

- Professor Hany Farid
- hfarid@berkeley.edu
- office: South Hall 203A
- farid.berkeley.edu/info206b
Welcome to INFO 206B

- Enrollment cap and waitlist
- Lectures
  - MW 10:00 - 12:00
  - two 45-minute lectures
  - hands-on coding
- Ask questions during class
- Laptops and phones
- Readers: Srikar Varanasi & Anna Waldo
Welcome to INFO 206B

- Grading
  - 6 short assignments (25%)
  - 3 long assignments (25%)
  - 2 exams (50%)
- Don’t fall behind in this class!
- Honor code
Python
In [3]:
   # My first Python program
   print("Hello World")

Hello World
# This is a function consisting of:
#   1. a header (def ...): "def" is a keyword
#   2. a body (print ...): the body is indented using <tab>

def say_introduction():
    print("My name is Inigo Montoya.")

def threaten_vengeance():
    print("You killed my father.")
    print("Prepare to die.")

print("Hello.")

Hello.
Hello.
My name is Inigo Montoya.
You killed my father.
Prepare to die.
Hello.
You killed my father.
Prepare to die.

# This is a function consisting of:
#   1. a header (def ...): "def" is a keyword
#   2. a body (print ...): the body is indented using <tab>

def say_introduction():
    print("My name is Inigo Montoya.")

def threaten_vengeance():
    print("You killed my father.")
    print("Prepare to die.")
    # Do something...

print("Hello.")
say_introduction()  # this is a function call
threaten_vengeance()  # this is another function call
print("Hello.")
threaten_vengeance()  # and another
In [16]:

# We can use functions that someone else wrote
#
# In these examples, we *pass* parameters to a function
from simplefunctions import print_sqrt # make a function available to you

print_sqrt(4) # call the function
print_sqrt(9)

2.0
3.0
# We can use functions that someone else wrote

```python
from simplefunctions import print_date_and_time  # make a function available to you

print_date_and_time()  # call the function
```

2019-06-08 10:06:47.104904
Variables

meaning_of_life = 42
variables

meaning_of_life = 42

print( meaning_of_life )
Variables

meaning_of_life = 42

print( meaning_of_life )

output: 42
While Loop

while condition:
    body
While Loop

```python
while condition:
    body
```

an expression that evaluates to a boolean value (True/False)
While Loop

```
while condition:
    body
```

one or more lines of code (indented, just like a function definition)
Conditional Operators

<  less than
>  greater than
== equal to
>= greater than or equal to
<= less than or equal to
!= not equal to
While Loop

t = 100
while t >= 0:
    print(t)
    t = t - 1

print("blast off")
compute the whole-number factors of a number

the factors of 42 are 1, 2, 3, 6, 7, 14, 21, 42
n = 42
f = 1

while f <= n:
    if n % f == 0:
        print(str(f) + " is a factor of " + str(n))

    f = f + 1

print("And that's all the factors of " + str(n))
n = 42
f = 1

while f <= n:
    if n % f == 0:
        print(str(f) + " is a factor of " + str(n))

    f = f + 1

print("And that's all the factors of " + str(n))
```
import math

n = 42
f = 1

while f <= n:
    if n % f == 0:
        print(str(f) + " is a factor of " + str(n))
    f = f + 1

print("And that's all the factors of " + str(n))
```
n = 42
f = 1

while f <= n:
    if n % f == 0:
        print(str(f) + " is a factor of " + str(n))
    f = f + 1

print("And that's all the factors of " + str(n))
n = 42
f = 1

while f <= n:
    if n % f == 0:
        print(str(f) + " is a factor of " + str(n))
    f = f + 1

print("And that's all the factors of " + str(n))
n = 42
f = 1

while f <= n:
    if n % f == 0:
        print(str(f) + " is a factor of " + str(n))

    f = f + 1

print("And that's all the factors of " + str(n))
n = 42
f = 1

while f <= n:
    if n % f == 0:
        print(str(f) + " is a factor of " + str(n))
    f = f + 1

print("And that's all the factors of " + str(n))
n = 42
f = 1

while f <= n:
    if n % f == 0:
        print(str(f) + " is a factor of " + str(n))
    f = f + 1

print("And that's all the factors of " + str(n))
temperature = 72

if temperature <= 32:
    print("It's freezing.")
else:
    print("It’s not so cold.")
```python
import random

# Conditionals

# Set temperature
temperature = 72

# Check conditions and print corresponding message
if temperature <= 32:
    print("It's freezing.")
elif temperature <= 50:
    print("It's cool.")
elif temperature <= 75:
    print("It's warm.")
else:
    print("It's hot.")
```
The 7 moons of Jupiter

moons1 = "Io"
moons2 = "Europa"
moons3 = "Ganymede"
moons4 = "Callisto"
moons5 = "Iapetus"
moons6 = "Ione"
moons7 = "Tethys"
# Create and initialize a list

```
moons = ['Io', 'Europa', 'Ganymede', 'Callisto', 'Iapetus', 'Ione', 'Tethys']
```
# Create and initialize a list
moons = ["Io", "Europa", "Ganymede", "Callisto", "Iapetus", "Ione", "Tethys"]

# Print a few items of the list
print("Jupiter's first moon is " + moons[0] + ".")
print("Jupiter's second moon is " + moons[1] + ".")
# Create and initialize a list
moons = ["Io", "Europa", "Ganymede", "Callisto", "Iapetus", "Ione", "Tethys"]

# Print a few items of the list
print("Jupiter's first moon is " + moons[0] + ".")
print("Jupiter's second moon is " + moons[1] + ".")

Jupiter's first moon is Io.
Jupiter's second moon is Europa.
# Create and initialize a list
moons = ["Io", "Europa", "Ganymede", "Callisto", "Iapetus", "Ione", "Tethys"]

# Print a few items of the list
print("Jupiter's first moon is " + moons[0] + ".")
print("Jupiter's second moon is " + moons[1] + ".")

# Print the entire list
print(moons)

['Io', 'Europa', 'Ganymede', 'Callisto', 'Iapetus', 'Ione', 'Tethys']
# Create and initialize a list
moons = ["Io", "Europa", "Ganymede", "Callisto", "Iapetus", "Ione", "Tethys"]

# Print a few items of the list
print("Jupiter's first moon is " + moons[0] + ".")
print("Jupiter's second moon is " + moons[1] + ".")

# Print the entire list
print(moons)

# A list can contain anything
numbers = [3, 7, 42, 19, 26]
print(numbers)

[3, 7, 42, 19, 26]
# Create and initialize a list
moons = ["Io", "Europa", "Ganymede", "Callisto", "Iapetus", "Ione", "Tethys"]

# Print a few items of the list
print("Jupiter's first moon is " + moons[0] + ".")
print("Jupiter's second moon is " + moons[1] + ".")

# Print the entire list
print(moons)

# A list can contain anything
numbers = [3, 7, 42, 19, 26]
print(numbers)

# A list can be empty
nothing = []
print(nothing)
characters = ["Horton", "Lorax", "Mayzie"]

# change an element in the list
characters[1] = "Thing One"
print(characters)
characters = ["Horton", "Lorax", "Mayzie"]

# change an element in the list
characters[1] = "Thing One"
print(characters)

["Horton", "Thing One", "Mayzie"]
# lists and loops

nums = [1, 7, 4, 10, 8]
i = 0

while i < len(nums):
    if (nums[i] % 2 == 0):
        print(str(nums[i]) + " is even")
    i = i + 1
# while loop
i = 1
while i <= 10:
    print(i)
    i = i + 1

# for loop
for i in range(1,11):
    print(i)
# while loop
i = 1
while i <= 10:
    print(i)
    i = i + 1

# for loop
for i in range(1, 11):
    print(i)
# lists and loops

```python
nums = [1, 7, 4, 10, 8]
for i in range(0, len(nums)):
    if (nums[i] % 2 == 0):
        print(str(nums[i]) + " is even")
```
# lists and loops
nums = [1, 7, 4, 10, 8]
for n in nums:
    if n % 2 == 0:
        print(str(n) + " is even")
# lists and loops
nums = [1, 7, 4, 10, 8]
for n in nums:
    if( n % 2 == 0 ):
        print(str(n) + " is even")

# lists and loops
nums = [1, 7, 4, 10, 8]
for i in range(0,len(nums)):
    if( nums[i] % 2 == 0 ):
        print(str(nums[i]) + " is even")
Reverse List

$L = ["a", "b", "c", "d", "e", "f"]$
Reverse List

L = ["a", "b", "c", "d", "e", "f"]
Reverse List

\[
L = ["a", "b", "c", "d", "e", "f"]
\]
Reverse List

L = ["a", "b", "c", "d", "e", "f"]
Reverse List

L = ["a", "b", "c", "d", "e", "f"]

for every item in list:
    swap item with corresponding item at end of list
Reverse List

L = ["a", "b", "c", "d", "e", "f"]

for first half of items in list:
    swap item with corresponding item at end of list
Reverse List

L = ["a", "b", "c", "d", "e"]

for first half (round down) of items in list:
    swap item with corresponding item at end of list
Reverse List (swap)

L = ["a", "b", "c", "d", "e"]
Reverse List (swap)

L = ['a', 'b', 'c', 'd', 'e']

L[0] = L[4]
L[4] = L[0]
Reverse List (swap)

L = ["a", "b", "c", "d", "e"]

L[0] = L[4]  \quad \Rightarrow \quad [ebcde]
L[4] = L[0]
Reverse List (swap)

L = ["e", "b", "c", "d", "e"]

L[0] = L[4]
L[4] = L[0]  \rightarrow  [ebcde]
Reverse List (swap)

$L = [\text{"a"}, \text{"b"}, \text{"c"}, \text{"d"}, \text{"e"}]$

temp = $L[0]$  

$L[0] = L[4]$  

$L[4] = \text{temp}$
Reverse List (swap)

L = ["a", "b", "c", "d", "e"]

temp = L[1]      b
L[1] = L[3]      [edcda]
L[3] = temp      [edcba]
# Reverse a list

L = [1, 3, 5, 7, 9, 11, 13, 15]
i = 0

while i < len(L) / 2:
    j = len(L) - 1 - i

    # Swap the items at i and j
    temp = L[i]
    L[i] = L[j]
    L[j] = temp

i = i + 1
Reversing a list

# Reverse a list
L = [1, 3, 5, 7, 9, 11, 13, 15]
i = 0
while i < len(L) / 2:
    j = len(L) - 1 - i
    # Swap the items at i and j
    temp = L[i]
    L[i] = L[j]
    L[j] = temp
    i = i + 1
# Reverse a list

L = [1, 3, 5, 7, 9, 11, 13, 15]
i = 0

while i < len(L) / 2:
    j = len(L) - 1 - i
    temp = L[i]
    L[i] = L[j]
    L[j] = temp

    # Swap the items at i and j
    i = i + 1

    i = 1
    j = 8-1-1 = 6
# Reverse a list

L = [1, 3, 5, 7, 9, 11, 13, 15]

i = 0

while i < len(L) / 2:
    j = len(L) - 1 - i
    temp = L[i]
    L[i] = L[j]
    L[j] = temp
    i = i + 1

# Swap the items at i and j

i = 2
j = 8-1-2 = 5
# Reverse a list

L = [1, 3, 5, 7, 9, 11, 13, 15]

i = 0

while i < len(L) / 2:
    j = len(L) - 1 - i
    # Swap the items at i and j
    temp = L[i]
    L[i] = L[j]
    L[j] = temp
    i = i + 1
Reversing a list

# Reverse a list
L = [1, 3, 5, 7, 9, 11, 13, 15]
i = 0
while i < len(L) / 2:
    j = len(L) - 1 - i
    # Swap the items at i and j
    temp = L[i]
    L[i] = L[j]
    L[j] = temp
    i = i + 1
Reversing a list

L = [1, 3, 5, 7, 9, 11, 13, 15]
for i in ???:
    j = ???
    temp = L[i]
    L[i] = L[j]
    L[j] = temp
Reversing a list

L = [1, 3, 5, 7, 9, 11, 13, 15]
for i in range(0,len(L)/2):
    j = ???
    temp = L[i]
    L[i] = L[j]
    L[j] = temp
Reversing a list

L = [1, 3, 5, 7, 9, 11, 13, 15]
for i in range(0,len(L)/2):
    j = len(L) - 1 - i
    temp = L[i]
    L[i] = L[j]
    L[j] = temp.
More on lists

\[ x = 5 \]

\[ x : \boxed{5} \]
More on lists

\( x = ["a", "b", "c"] \)

\( x : \begin{array}{c}
1000
\end{array} \rightarrow \begin{array}{c}
a \\
b \\
c
\end{array} \)

address

content
More on lists

\[x = \text{["a", "b", "c"]}\]
\[y = x\]
More on lists

```python
x = ["a", "b", "c"]
y = x
y[0] = "d"

print(y)  # Output: ['a', 'd', 'c']
print(x)  # Output: ['a', 'b', 'c']
```
# add one to each element of each sub-list
L = [[1,2,3], [4,5,6], [7,8,9]]
# add one to each element of each sub-list
L = [[1,2,3], [4,5,6], [7,8,9]]

for i in range(0,len(L)):
    L[i] = L[i] + 1
# add one to each element of each sub-list
L = [[1,2,3], [4,5,6], [7,8,9]]

for i in range(0,len(L)):
    L[i] = L[i] + 1

for i in range(0,len(L)):
# add one to each element of each sub-list
L = [[1,2,3], [4,5,6], [7,8,9]]

for i in range(0,len(L)):
    L[i] = L[i] + 1

for i in range(0,len(L)):
    L[i][0] = L[i][0] + 1
    L[i][1] = L[i][1] + 1
    L[i][2] = L[i][2] + 1
# add one to each element of each sub-list
L = [[1,2], [3,4,5], [6,7,8,9]]

for i in range(0,len(L)):
    L[i][0] = L[i][0] + 1
    L[i][1] = L[i][1] + 1
    L[i][2] = L[i][2] + 1
# add one to each element of each sub-list
L = [[1,2], [3,4,5], [6,7,8,9]]

for i in range(0,len(L)):
    L[i][0] = L[i][0] + 1
    L[i][1] = L[i][1] + 1
    L[i][2] = L[i][2] + 1
# add one to each element of each sub-list
L = [[1,2], [3,4,5], [6,7,8,9]]

for i in range(0,len(L)):
# add one to each element of each sub-list
L = [[1,2], [3,4,5], [6,7,8,9]]

for i in range(0,len(L)):
    for j in range(0,len(L[i])):
# add one to each element of each sub-list
L = [[1,2], [3,4,5], [6,7,8,9]]

for i in range(0,len(L)):
    for j in range(0,len(L[i])):  
        L[i][j] = L[i][j] + 1
# the function `sqrt` takes as input a number and returns a number
from math import sqrt
x = sqrt(4)
print(sqrt(4))

# the function `+` takes as input two numbers and returns a number
x = 8 + 12
print(8 + 12)

# the function `len` takes as input a string and returns an integer
print(len("eggplant"))

# the return value of one function can be the input to another
print(int(8.485) + 12)
print(int( sqrt(72) ) + 12)
Returning Parameters

def compute_four():
    return 24 / 4 - 2

x = compute_four()
print(x)

y = 24 / compute_four()
print(y)

print(compute_four())
```python
def compute_four():
    return 24 / 4 - 2

x = compute_four()
print(x)  # 4

y = 24 / compute_four()
print(y)

print(compute_four())
```
Returning Parameters

def compute_four():
    return 24 / 4 - 2

x = compute_four()
print(x)  # 4

y = 24 / compute_four()
print(y)  # 6

print(compute_four())
def compute_four():
    return 24 / 4 - 2

x = compute_four()
print(x)  # 4

y = 24 / compute_four()
print(y)  # 6

print(compute_four())  # 4
def add_five(x):
    x = x + 5
    return x

z = 4
add_five(z)
print(z)

x = add_five(z)
print(x)

print(add_five(z))
def add_five(x):
    x = x + 5
    return x

z = 4
add_five(z)
print(z)

x = add_five(z)
print(x)

print(add_five(z))

Passing & Returning Parameters
def add_five(x):
    x = x + 5
    return x

z = 4
add_five(z)
print(z)  # 4

x = add_five(z)
pprint(x)  # 9

print(add_five(z))
def add_five(x):
    x = x + 5
    return x

z = 4
add_five(z)
print(z)  # Output: 4

x = add_five(z)
print(x)  # Output: 9

print(add_five(z))  # Output: 9
from random import uniform

def cheap_magic_8_ball():
    r = uniform(0, 4)

    if r > 3:
        return "Most likely"
    elif r > 2:
        return "Ask again later"
    elif r > 1:
        return "Don't count on it"
    else:
        return "No"

print(cheap_magic_8_ball())
from random import uniform

def cheap_magic_8_ball():
    r = uniform(0, 4)

    if r > 3:
        print("Most likely")
    elif r > 2:
        print("Ask again later")
    elif r > 1:
        print("Don't count on it")
    else:
        print("No")

cheap_magic_8_ball()
Local vs. Global Variables

def some_function():
    global x
    x = x + 1
    print(x)

x = 10
some_function()
Local vs. Global Variables

def some_function():
    x = 4
    print(x)

some_function()
print(x)
print(x)
Local vs. Global Variables

def some_function():
    x = 4
    print(x)

some_function()
print(x)  
error: x is not defined
Local vs. Global Variables

def print_x():
    global x
    x = x + 1
    print(x)

x = 5
# x is a global variable
print_x()
print_x()
Local vs. Global Variables

```python
def print_x():
    global x
    x = x + 1
    print(x)

x = 5
print_x()  # x is a global variable
print_x()
```

6
6
Local vs. Global Variables

def print_x():
    global x
    x = x + 1
    print(x)

x = 5  # x is a global variable
print_x()  # 6
print_x()  # 7
def print_x():
    #global x
    x = x + 1
    print(x)

x = 5  # x is a global variable
print_x()
Local vs. Global Variables

def print_x():
    #global x
    x = x + 1
    print(x)

x = 5
# x is a global variable
print_x()  # error: local variable 'x' referenced before assignment
Local vs. Global Variables

```python
def print_x():
    #global x
    #x = x + 1
    print(x)

x = 5  # x is a global variable
print_x()
```
Local vs. Global Variables

def print_x():
    #global x
    #x = x + 1
    print(x)

x = 5  # x is a global variable
print_x()  # 5
from math import pi

def compute_circle_area(r):
    global pi
    pi = 3  # changes the value of pi, now and forever. Scary!
    return pi * r * r

compute_circle_area(10)
Local vs. Global Variables

```python
x = 1

def test(x):
    global x
    x = x + 1
    print(x)

test(10)  # error: name 'x' is local and global
```
A class combines (and abstracts) data and functions

An object is an instantiation of a class
Classes & Objects

class

object
Classes & Objects

A class combines (and abstracts) data and functions

An object is an instantiation of a class

String is a built-in class, append is a function

Int is a built-in class, + is a function

We can define our own classes
myball = Ball(10.0, 15.0, 0.0, -5.0)

constructor:
Classes & Objects

myball = Ball(10.0, 15.0, 0.0, -5.0)

constructor:
  - allocate memory for a Ball object
myball = Ball(10.0, 15.0, 0.0, -5.0)

constructor:

- allocate memory for a Ball object
- initializes the Ball object with values
Classes & Objects

myball = Ball(10.0, 15.0, 0.0, -5.0)

constructor:
  • allocate memory for a Ball object
  • initializes the Ball object with values
  • returns address of the Ball object
Classes & Objects

myball = Ball(10.0, 15.0, 0.0, -5.0)

constructor:
• allocate memory for a Ball object
• initializes the Ball object with values
• returns address of the Ball object
• similar to a list
import drawSvg as draw

D = draw.Drawing(200, 200, origin='center')  # define drawing canvas
EARTH_GRAVITY_ACCELERATION = -9.8  # acceleration due to gravity, m/sec^2
BALL_RADIUS = 10  # radius of the ball in pixels

class Ball:
    def __init__(self, start_x, start_y, start_v_x, start_v_y, color='blue'):
        # Ball location and velocity
        self.x = start_x
        self.y = start_y
        self.v_x = start_v_x
        self.v_y = start_v_y

        # Ball color, for drawing purposes
        self.color = color

    def update_position(self, timestep):
        self.x = self.x + timestep * self.v_x
        self.y = self.y + timestep * self.v_y

    def update_velocity(self, timestep):
        self.v_y = self.v_y + timestep * EARTH_GRAVITY_ACCELERATION

    def animate_step(self, timestep):
        self.update_position(timestep)
        self.update_velocity(timestep)

    def draw(self):
        D.append(draw.Circle(self.x, self.y, BALL_RADIUS, fill=self.color))
Classes & Objects (methods)

```
import drawSvg as draw

D = draw.Drawing(200, 200, origin='center') # define drawing canvas
EARTH_GRAVITY_ACCELERATION = -9.8 # acceleration due to gravity, m/sec^2
BALL_RADIUS = 10 # radius of the ball in pixels

class Ball:
    def __init__(self, start_x, start_y, start_v_x, start_v_y, color='blue'):
        # Ball location and velocity
        self.x = start_x
        self.y = start_y
        self.v_x = start_v_x
        self.v_y = start_v_y

        # Ball color, for drawing purposes
        self.color = color

    def update_position(self, timestep):
        self.x = self.x + timestep * self.v_x
        self.y = self.y + timestep * self.v_y

    def update_velocity(self, timestep):
        self.v_y = self.v_y + timestep * EARTH_GRAVITY_ACCELERATION

    def animate_step(self, timestep):
        self.update_position(timestep)
        self.update_velocity(timestep)

    def draw(self):
        D.append(draw.Circle(self.x, self.y, BALL_RADIUS, fill=self.color))
```
import drawSvg as draw

D = draw.Drawing(200, 200, origin='center')  # define drawing canvas
EARTH_GRAVITY_ACCELERATION = -9.8  # acceleration due to gravity, m/sec^2
BALL_RADIUS = 10  # radius of the ball in pixels

class Ball:
    def __init__(self, start_x, start_y, start_v_x, start_v_y, color='blue'):
        # Ball location and velocity
        self.x = start_x
        self.y = start_y
        self.v_x = start_v_x
        self.v_y = start_v_y

        # Ball color, for drawing purposes
        self.color = color

    def update_position(self, timestep):
        self.x = self.x + timestep * self.v_x
        self.y = self.y + timestep * self.v_y

    def update_velocity(self, timestep):
        self.v_y = self.v_y + timestep * EARTH_GRAVITY_ACCELERATION

    def animate_step(self, timestep):
        self.update_position(timestep)
        self.update_velocity(timestep)

    def draw(self):
        D.append(draw.Circle(self.x, self.y, BALL_RADIUS, fill=self.color))
import drawSvg as draw

D = draw.Drawing(200, 200, origin='center') # define drawing canvas
EARTH_GRAVITY_ACCELERATION = -9.8  # acceleration due to gravity, m/sec^2
BALL_RADIUS = 10  # radius of the ball in pixels

class Ball:
    def __init__(self, start_x, start_y, start_v_x, start_v_y, color='blue'):
        # Ball location and velocity
        self.x = start_x
        self.y = start_y
        self.v_x = start_v_x
        self.v_y = start_v_y

        # Ball color, for drawing purposes
        self.color = color

    def update_position(self, timestep):
        self.x = self.x + timestep * self.v_x
        self.y = self.y + timestep * self.v_y

    def update_velocity(self, timestep):
        self.v_y = self.v_y + timestep * EARTH_GRAVITY_ACCELERATION

    def animate_step(self, timestep):
        self.update_position(timestep)
        self.update_velocity(timestep)

    def draw(self):
        D.append(draw.Circle(self.x, self.y, BALL_RADIUS, fill=self.color))
myball = Ball(10.0, 15.0, 0.0, -5.0)

def __init__(self, start_x, start_y, start_v_x, start_v_y, color='blue'):
    # Ball location and velocity
    self.x = start_x
    self.y = start_y
    self.v_x = start_v_x
    self.v_y = start_v_y

    # Ball color, for drawing purposes
    self.color = color

x: 10.0
y: 15.0
v_x: 0.0
v_y: -5.0
color: blue
myball = Ball(10.0, 15.0, 0.0, -5.0)

myball.update_position(0.1)  # update_position(myball, 0.1)

def update_position(self, timestep):
    self.x = self.x + timestep * self.v_x  # myball.x = myball.x + ...  
    self.y = self.y + timestep * self.v_y