Essential Data Skills for Business Analytics

Lecture 1 (01/27): Introduction to Python

Decision, Operations & Information Technologies
Robert H. Smith School of Business
Spring, 2020


Welcome
Bienvenue
Velkomin
Bonvenon
Benvingut
Welkom
Willkommen
Dobrodošli
Mirëpres
Hospedar
Swagata
Yokôso
Tervetuloa
Vítejte
欢迎
Verwelkom
Acollir
Ongietorri
Selamat Datang
Bienvéni
Velkommen
Herzlich
Toivottaa
Degemer
EKomoMai
Maligayang Pagdating
Bonavinuta
Hwangyong Hamnida
Teretulnud
Dobredojde
Добре дошъл
Sugeng Rawuh
Siyakwamukela
Swagatham
Fáilte
Laukiamas
Swaagat
Namaste
Heten
iBiala
Qaimarutin
Akwaba
Goscic
Recoger
Kaabo
Croesawu
Sambut
Karibuni
Karibu
Khoshumadi
Hozta
Murakaza Neza
Sannu da zuwa
Kalosllthate
Moguah
Woezor
Wëllkom
MirëSevini
Uvitani
Ontvangen
Hoan Nghênh
GnindiTonHap

When and where

- Lecture discussion
  - Monday, Wednesday 8:00 – 9:15, 9:30 – 10:45
  - Room: VMH 3522
- TA Office hour
  - Time: TBD
  - Location: 2115 Susquehanna Hall
- My office
  - Room: 4316 Van Munching Hall
  - Time: TBD

What cover in this course

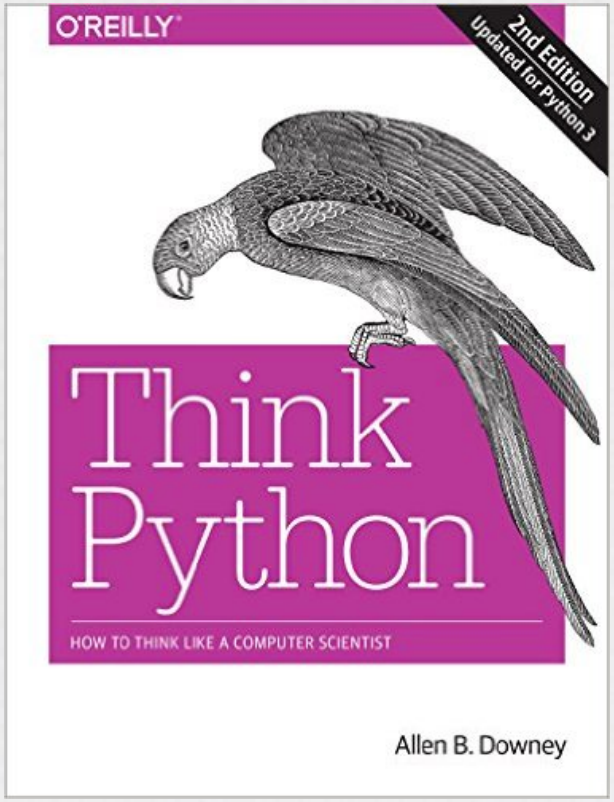
UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Basic python programming
  - ❑ Install and run python
  - ❑ Variables, expressions, statements
  - ❑ Functions, parameters, recursions
  - ❑ Control structures
  - ❑ Lists, tuples, dictionaries
  - ❑ File operations
  - ❑ String operations
  - ❑ Modules
  - ❑ Regular expression
  - ❑ Visualization
- Python packages for real-world applications
  - ❑ Database operations (MySQL)
  - ❑ Data manipulation: Pandas
  - ❑ Scientific computing: NumPy
  - ❑ Natural language processing: NLTK
  - ❑ Machine learning: scikit-learn

Recommended textbooks

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Think Python – How to Think Like a Computer Scientist. Available from Amazon.com
- The Python Tutorial



Prerequisites

- No prior programming needed
- Recommended
  - ❑Basic computer knowledge (e.g., software installation…)
  - ❑Database: SQL knowledge

Lab sessions

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- **Must attend**
- Within each lab, you need to finish a given task.
  - Typical tasks: reviewing what we've learned in lectures for that week.
- For some labs, you need to submit reports.
- With one lab missing, you will receive 2 points deduction from your final grade. **With 5 or more labs missing, you will automatically receive F for this course.**

Assignments

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- All assignments are hands-on programming tasks. You can discuss with other students, TA, or instructor. But you must work on the final submission by yourself.
- All assignments due Mondays at 8:00am.
- Each assignment will take about 1-2 hours on average.
- Late submission will receive credit deduction:
  - 0-1 day: 10%
  - 1-2 days: 20%
  - 2-3 days: 30%
  - 3-5 days: 50%
  - > 5 days: will not accept

Exams

- A midterm
- 2 quizzes
- I don't know exactly when or where either are yet.
  - When I find out, I will send out an email and post it on Canvas.
- They will be closed book written tests.

Final project

- A group project
  - ❑No more than 3 persons in your group
- An example project can be found here: http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/
- Some projects will be posted soon
- OR
- Your customized project (discuss with me in advance)

Grading policy

Participation	5%*1=5%
Midterm exam	25%*1=25%
Quizzes	10%*2 = 20%
Final project	15%*1=15%
Assignments	5%*7 =35%

Letter grades are assigned as follows:

Letter Grade	Points
A+	100 – 97
A	96.9 – 93
A-	92.9 – 90
B+	89.9 – 87
B	86.9 – 83
B-	82.9 – 80
C+	79.9 – 77
C	76.9 – 73
C-	72.9 – 70
D+	69.9 – 67
D	66.9 – 63
D-	62.9 – 60
F	Below 60

Attendance

- Encourage you to attend every lecture session and lab session
  - ❑Might have some random attendance checking
  - ❑Receive 'F' if absence for 5+

- Failing to attend midterm or quizzes will receive 'F' for the course except extreme reasons.

Getting help

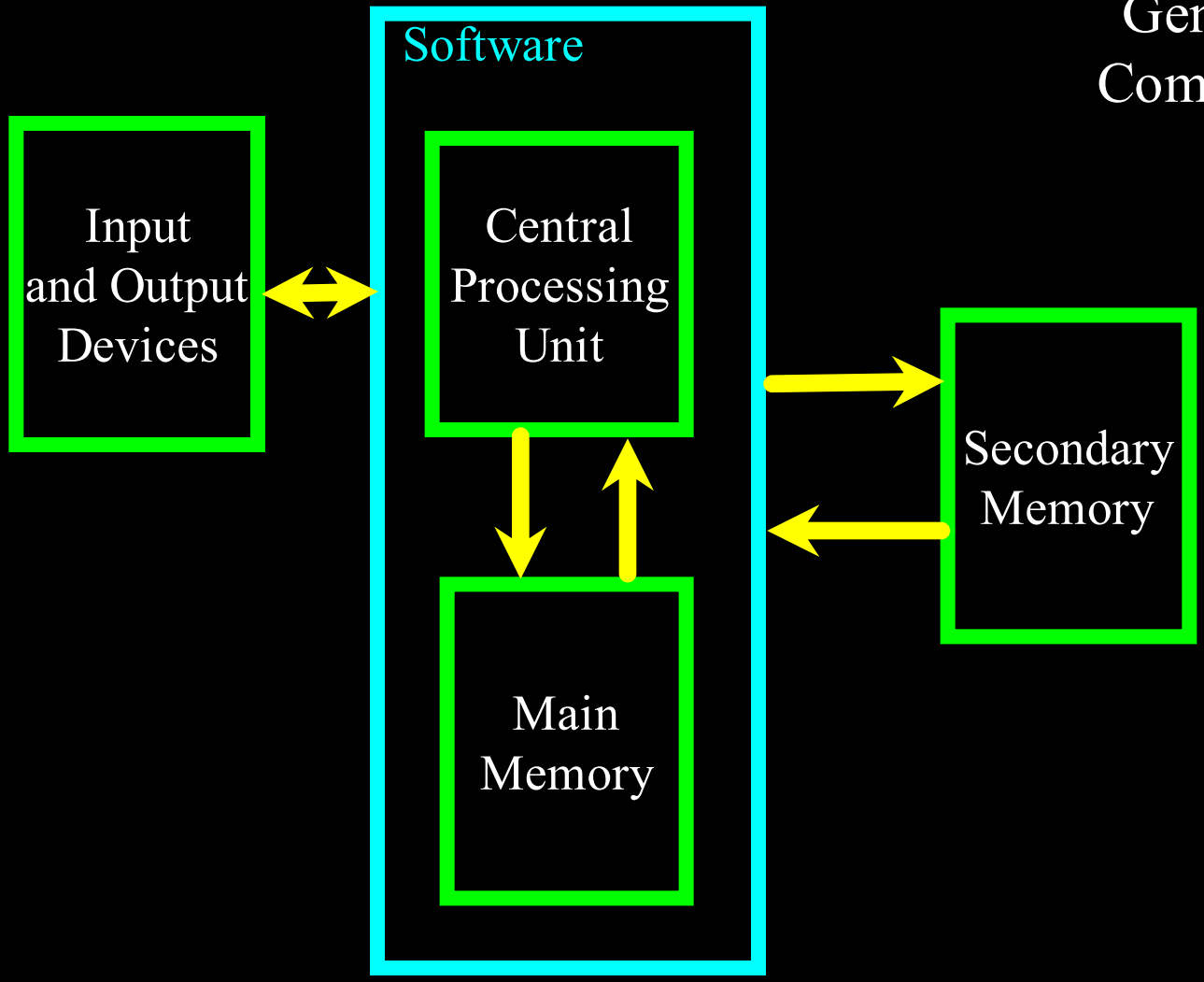
- Teaching Assistant: Yash Srivastava (yash.srivastava@rhsmith.umd.edu)
- Please email both TA and me with 'BMGT404' in the title
- Regular Office Hour
  - TBD
- Non-regular Office Hour
  - Appointments by email preferred
- Course webpage and EMLS

Hardware Architecture

http://upload.wikimedia.org/wikipedia/commons/3/3d/RaspberryPi.jpg

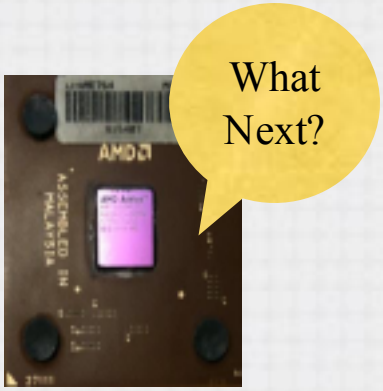
UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

Generic Computer



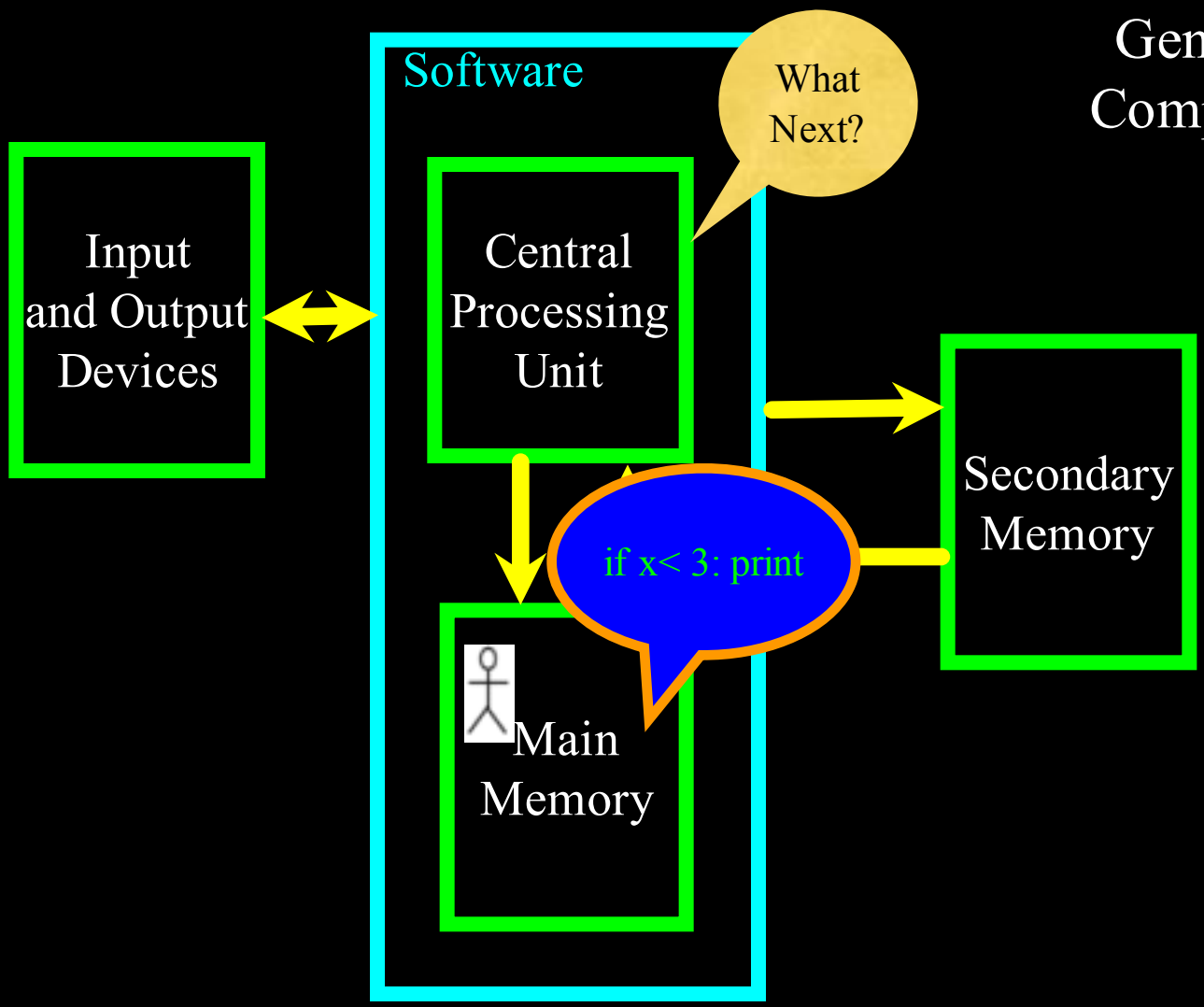
Definitions

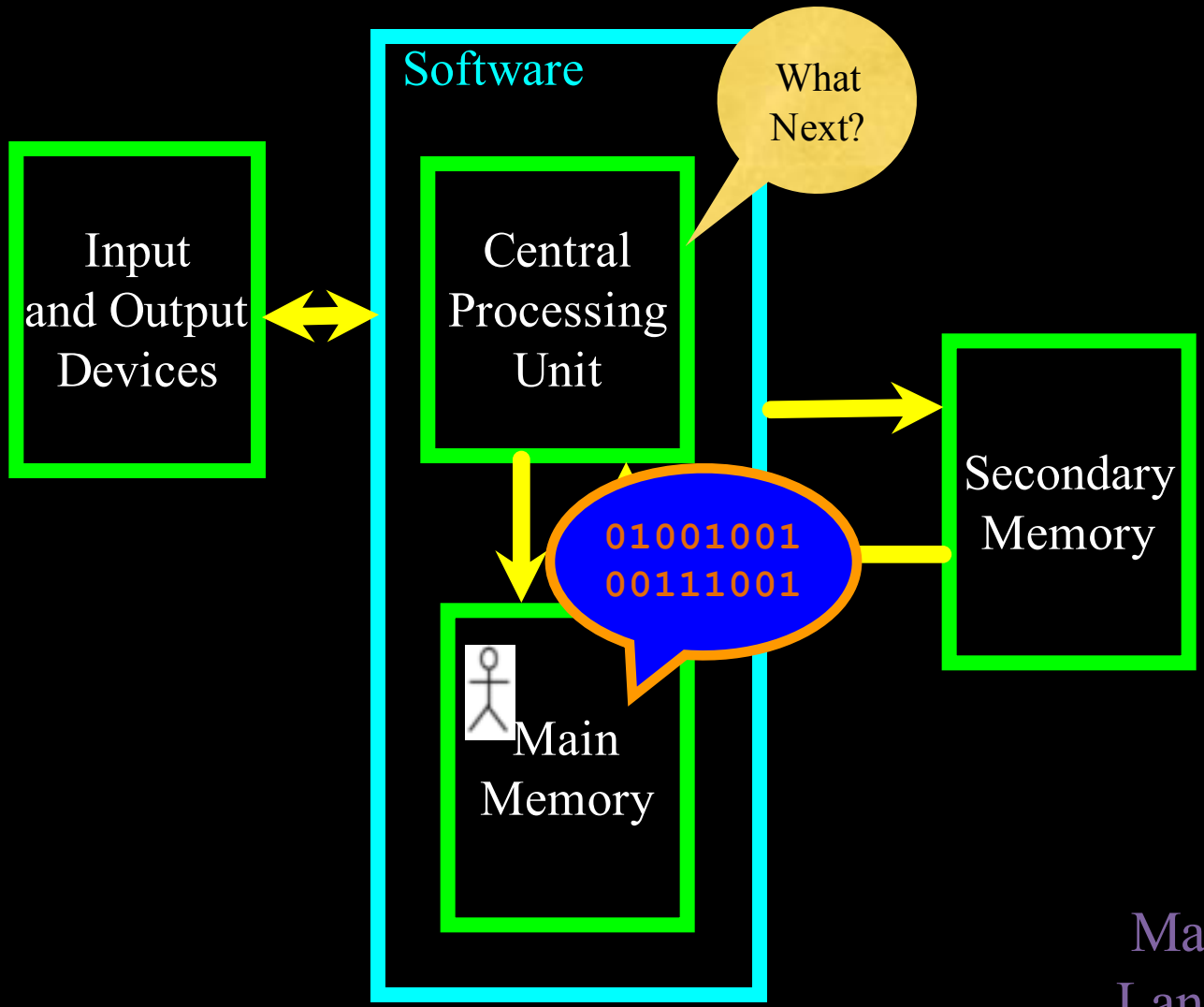
- Central Processing Unit: Runs the Program - The CPU is always wondering "what to do next"? Not the brains exactly - very dumb but very very fast
- Input Devices: Keyboard, Mouse, Touch Screen
- Output Devices: Screen, Speakers, Printer, DVD Burner
- Main Memory: Fast small temporary storage - lost on reboot - aka RAM
- Secondary Memory: Slower large permanent storage - lasts until deleted - disk drive / memory stick



UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

Generic Computer





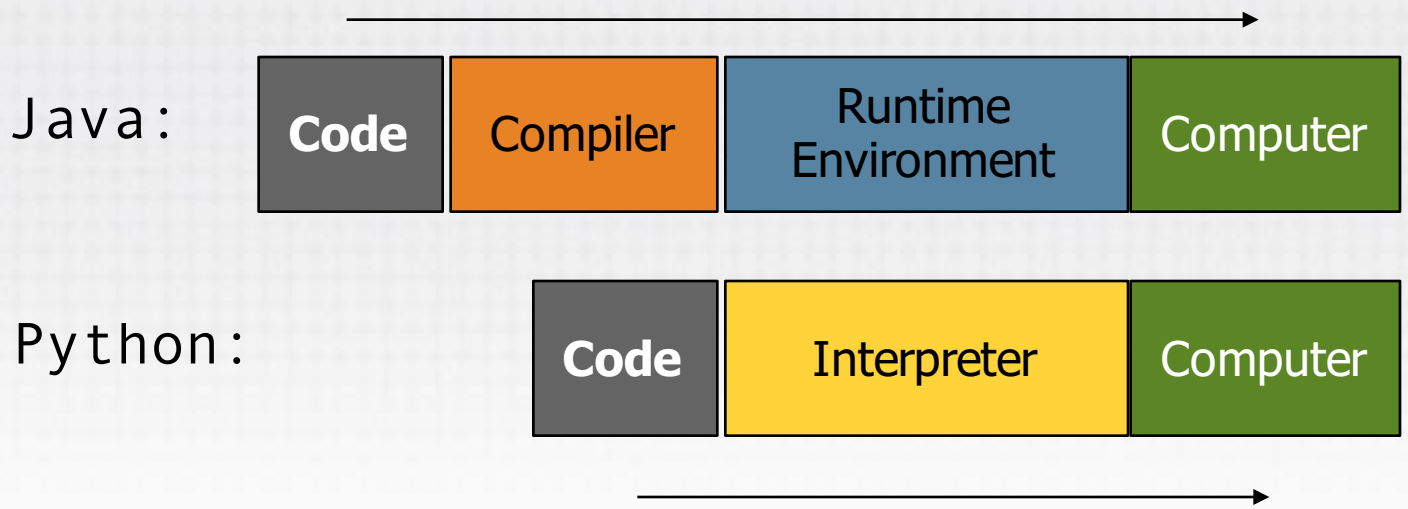
Machine Language

Python as a Language

Python is the language of the Python Interpreter and those who can converse with it. An individual who can speak Python is known as a Pythonista. It is a very uncommon skill, and may be hereditary. Nearly all known Pythonistas use software initially developed by Guido van Rossum.

Introduction to Python

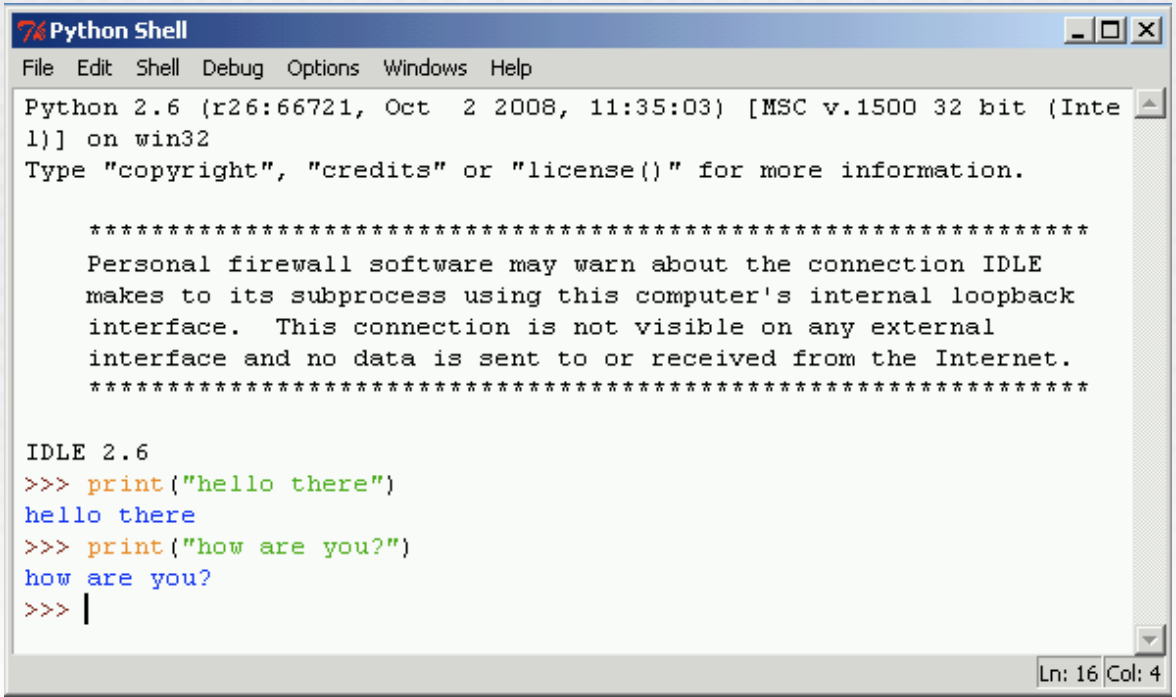
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
  - ❑ Not compiled like Java
  - ❑ Code is written and then directly executed by an interpreter
  - ❑ Type commands into interpreter and see immediate results



Introduction to Python

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- **Python is Interactive**: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
  - ❑ Allows you to type commands one-at-a-time and see results
  - ❑ A great way to explore Python's syntax



Introduction to Python

- **Python is Object-Oriented**: Python supports Object-Oriented style or technique of programming that encapsulates code within objects. (we will NOT cover this in this course)

- **Python is a Beginner's Language**: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing, numerical computing, web browsers to games.

Why Python?

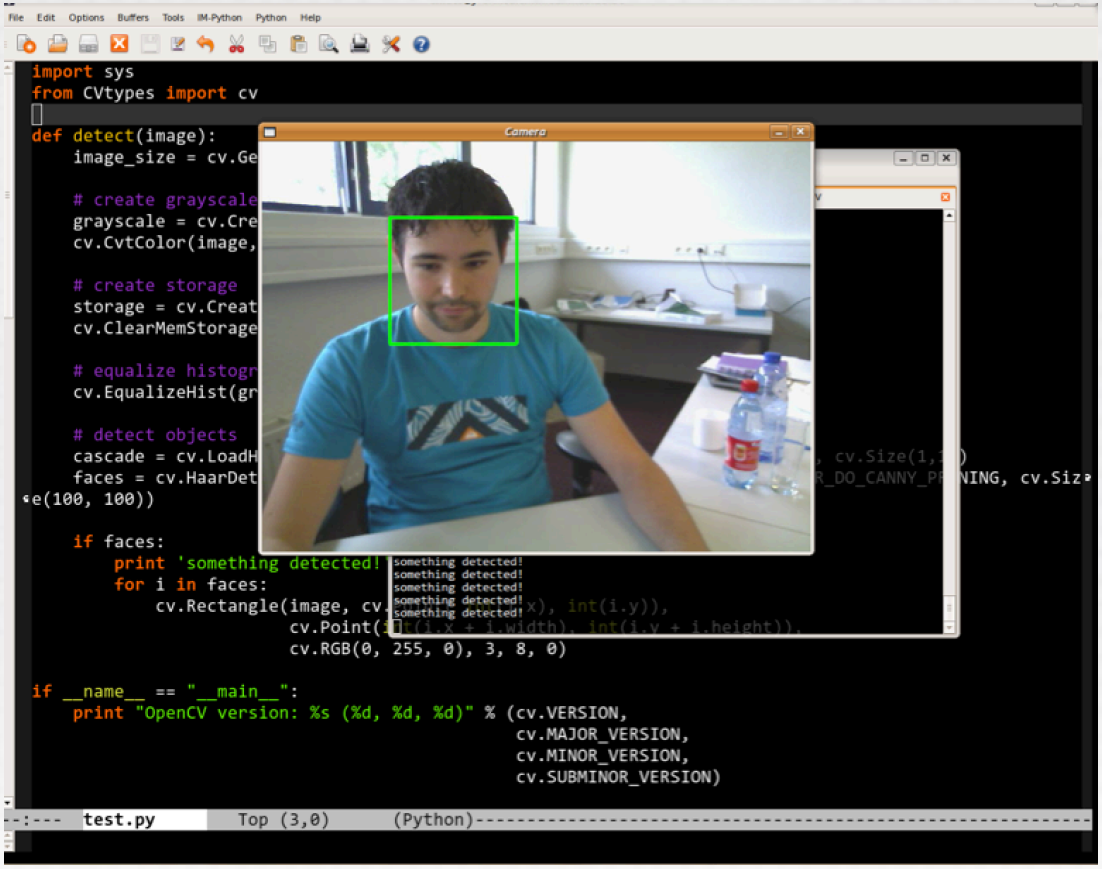
UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- **Easy-to-learn**: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read**: Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain**: Python's source code is fairly easy-to-maintain.
- **A broad standard library**: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive mode**: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable**: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable**: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases**: Python provides interfaces to all major commercial databases.
- **GUI Programming**: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable**: Python provides a better structure and support for large programs than shell scripting.

What Python can do (1)?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- With Python and OpenCV, we can do face detection

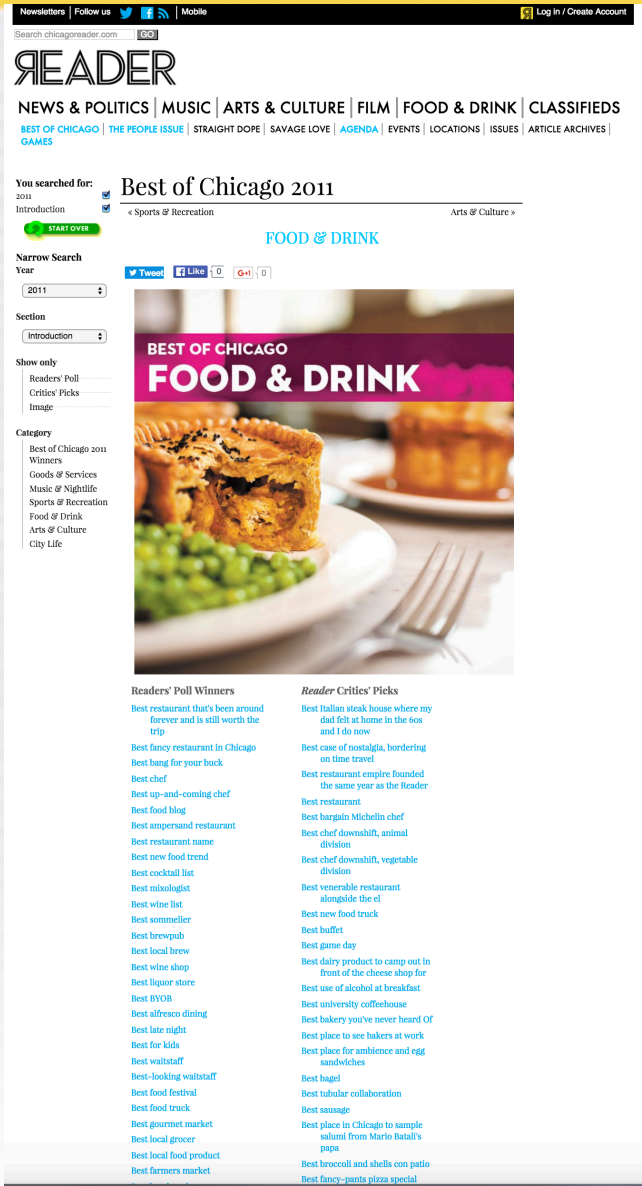


What Python can do (2)?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- With BeautifulSoup and urllib, we can write a Python script to get the winners of the Food and Drink categories in the Chicago Readers' Best of 2011 list.

```python
from bs4 import BeautifulSoup
from urllib2 import urlopen
from time import sleep # be nice

BASE_URL = "http://www.chicagoreader.com"

def make_soup(url):
    html = urlopen(url).read()
    return BeautifulSoup(html, "lxml")

def get_category_links(section_url):
    soup = make_soup(section_url)
    boccat = soup.find("dl", "boccat")
    category_links = [BASE_URL + dd.a["href"] for dd in boccat.findAll("dd")]
    return category_links

def get_category_winner(category_url):
    soup = make_soup(category_url)
    category = soup.find("h1", "headline").string
    winner = [h2.string for h2 in soup.findAll("h2", "boc1")]
    runners_up = [h2.string for h2 in soup.findAll("h2", "boc2")]
    return {"category": category,
            "category_url": category_url,
            "winner": winner,
            "runners_up": runners_up}

if __name__ == '__main__':
    food_n_drink = ("http://www.chicagoreader.com/chicago/"
                    "best-of-chicago-2011-food-drink/BestOf?oid=4106228")

    categories = get_category_links(food_n_drink)

    data = [] # a list to store our dictionaries
    for category in categories:
        winner = get_category_winner(category)
        data.append(winner)
        sleep(1) # be nice

    print data
```



What Python can do (3)?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

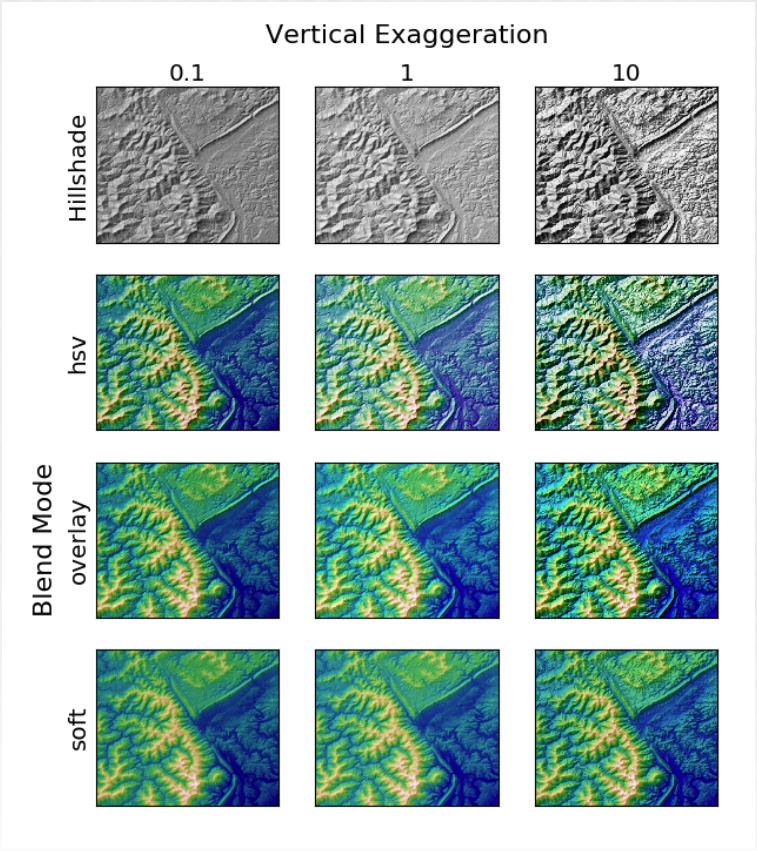
- With MySQLdb, we can write a Python script to fetch all records in the coworker database.

```
 1 #!/usr/bin/python
 2
 3 import MySQLdb
 4
 5 db = MySQLdb.connect(host="localhost", port=3306, user="foo", passwd="bar", db="qoz")
 6 cursor = db.cursor()
 7 cursor.execute("SELECT name, phone_number FROM coworkers")
 8 data = cursor.fetchall()
 9 for row in data :
10     print row[0],",",row[1]
11
12
~
~
~
~
~
~
~
```

What Python can do (4)?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- With matplotlib, we can make beautiful plots.




```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.cbook import get_sample_data
from matplotlib.colors import LightSource

dem = np.load(get_sample_data('jacksboro_fault_dem.npz'))
z = dem['elevation']

#-- Optional dx and dy for accurate vertical exaggeration ----------------------
# If you need topographically accurate vertical exaggeration, or you don't want
# to guess at what *vert_exag* should be, you'll need to specify the cellsize
# of the grid (i.e. the *dx* and *dy* parameters).  Otherwise, any *vert_exag*
# value you specify will be realitive to the grid spacing of your input data
# (in other words, *dx* and *dy* default to 1.0, and *vert_exag* is calculated
# relative to those parameters).  Similarly, *dx* and *dy* are assumed to be in
# the same units as your input z-values.  Therefore, we'll need to convert the
# given dx and dy from decimal degrees to meters.
dx, dy = dem['dx'], dem['dy']
dy = 111200 * dy
dx = 111200 * dx * np.cos(np.radians(dem['ymin']))
#------------------------------------------------------------------------------

# Shade from the northwest, with the sun 45 degrees from horizontal
ls = LightSource(azdeg=315, altdeg=45)
cmap = plt.cm.gist_earth

fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(8, 9))
plt.setp(axes.flat, xticks=[], yticks=[])

# Vary vertical exaggeration and blend mode and plot all combinations
for col, ve in zip(axes.T, [0.1, 1, 10]):
    # Show the hillshade intensity image in the first row
    col[0].imshow(ls.hillshade(z, vert_exag=ve, dx=dx, dy=dy), cmap='gray')

    # Place hillshaded plots with different blend modes in the rest of the rows
    for ax, mode in zip(col[1:], ['hsv', 'overlay', 'soft']):
        rgb = ls.shade(z, cmap=cmap, blend_mode=mode,
                       vert_exag=ve, dx=dx, dy=dy)
        ax.imshow(rgb)

# Label rows and columns
for ax, ve in zip(axes[0], [0.1, 1, 10]):
    ax.set_title('{0}'.format(ve), size=18)
for ax, mode in zip(axes[:, 0], ['Hillshade', 'hsv', 'overlay', 'soft']):
    ax.set_ylabel(mode, size=18)

# Group labels...
axes[0, 1].annotate('Vertical Exaggeration', (0.5, 1), xytext=(0, 30),
                    textcoords='offset points', xycoords='axes fraction',
                    ha='center', va='bottom', size=20)
axes[2, 0].annotate('Blend Mode', (0, 0.5), xytext=(-30, 0),
                    textcoords='offset points', xycoords='axes fraction',
                    ha='right', va='center', size=20, rotation=90)
fig.subplots_adjust(bottom=0.05, right=0.95)

plt.show()
```

What Python can do (5)?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- With NLTK, we can do many language processing operations.

```
>>> from nltk import sent_tokenize, word_tokenize, pos_tag
>>> text = "Machine learning is the science of getting computers to act without being
explicitly programmed. In the past decade, machine learning has given us self-driving cars,
practical speech recognition, effective web search, and a vastly improved understanding of the
human genome. Machine learning is so pervasive today that you probably use it dozens of times
a day without knowing it. Many researchers also think it is the best way to make progress
towards human-level AI. In this class, you will learn about the most effective machine
learning techniques, and gain practice implementing them and getting them to work for
yourself. More importantly, you'll learn about not only the theoretical underpinnings of
learning, but also gain the practical know-how needed to quickly and powerfully apply these
techniques to new problems. Finally, you'll learn about some of Silicon Valley's best
practices in innovation as it pertains to machine learning and AI."
>>> tokens = word_tokenize(text)
>>> tokens
['Machine', 'learning', 'is', 'the', 'science', 'of', 'getting', 'computers', 'to', 'act',
'without', 'being', 'explicitly', 'programmed.', 'In', 'the', 'past', 'decade', ',',
'machine', 'learning', 'has', 'given', 'us', 'self-driving', 'cars', ',', 'practical',
'speech', 'recognition', ',', 'effective', 'web', 'search', ',', 'and', 'a', 'vastly',
'improved', 'understanding', 'of', 'the', 'human', 'genome.', 'Machine', 'learning', 'is',
'so', 'pervasive', 'today', 'that', 'you', 'probably', 'use', 'it', 'dozens', 'of', 'times',
'a', 'day', 'without', 'knowing', 'it.', 'Many', 'researchers', 'also', 'think', 'it', 'is',
'the', 'best', 'way', 'to', 'make', 'progress', 'towards', 'human-level', 'AI.', 'In', 'this',
'class', ',', 'you', 'will', 'learn', 'about', 'the', 'most', 'effective', 'machine',
'learning', 'techniques', ',', 'and', 'gain', 'practice', 'implementing', 'them', 'and',
'getting', 'them', 'to', 'work', 'for', 'yourself.', 'More', 'importantly', ',', 'you', "'ll",
'learn', 'about', 'not', 'only', 'the', 'theoretical', 'underpinnings', 'of', 'learning', ',',
'but', 'also', 'gain', 'the', 'practical', 'know-how', 'needed', 'to', 'quickly', 'and',
'powerfully', 'apply', 'these', 'techniques', 'to', 'new', 'problems.', 'Finally', ',', 'you',
"'ll", 'learn', 'about', 'some', 'of', 'Silicon', 'Valley', "'s", 'best', 'practices', 'in',
'innovation', 'as', 'it', 'pertains', 'to', 'machine', 'learning', 'and', 'AI', '.']
```

```
>>> tagged_tokens = pos_tag(tokens)
>>> tagged_tokens
[('Machine', 'NN'), ('learning', 'NN'), ('is', 'VBZ'), ('the', 'DT'), ('science', 'NN'),
('of', 'IN'), ('getting', 'VBG'), ('computers', 'NNS'), ('to', 'TO'), ('act', 'VB'),
('without', 'IN'), ('being', 'VBG'), ('explicitly', 'RB'), ('programmed.', 'NNP'), ('In',
'NNP'), ('the', 'DT'), ('past', 'JJ'), ('decade', 'NN'), (',', ','), ('machine', 'NN'),
('learning', 'NN'), ('has', 'VBZ'), ('given', 'VBN'), ('us', 'PRP'), ('self-driving', 'JJ'),
('cars', 'NNS'), (',', ','), ('practical', 'JJ'), ('speech', 'NN'), ('recognition', 'NN'),
(',', ','), ('effective', 'JJ'), ('web', 'NN'), ('search', 'NN'), (',', ','), ('and', 'CC'),
('a', 'DT'), ('vastly', 'RB'), ('improved', 'VBN'), ('understanding', 'NN'), ('of', 'IN'),
('the', 'DT'), ('human', 'JJ'), ('genome.', 'NNP'), ('Machine', 'NNP'), ('learning', 'NN'),
('is', 'VBZ'), ('so', 'RB'), ('pervasive', 'JJ'), ('today', 'NN'), ('that', 'WDT'), ('you',
'PRP'), ('probably', 'RB'), ('use', 'VBP'), ('it', 'PRP'), ('dozens', 'VBZ'), ('of', 'IN'),
('times', 'NNS'), ('a', 'DT'), ('day', 'NN'), ('without', 'IN'), ('knowing', 'NN'), ('it.',
'NNP'), ('Many', 'NNP'), ('researchers', 'NNS'), ('also', 'RB'), ('think', 'VBP'), ('it',
'PRP'), ('is', 'VBZ'), ('the', 'DT'), ('best', 'JJS'), ('way', 'NN'), ('to', 'TO'), ('make',
'VB'), ('progress', 'NN'), ('towards', 'NNS'), ('human-level', 'JJ'), ('AI.', 'NNP'), ('In',
'NNP'), ('this', 'DT'), ('class', 'NN'), (',', ','), ('you', 'PRP'), ('will', 'MD'), ('learn',
'VB'), ('about', 'IN'), ('the', 'DT'), ('most', 'RBS'), ('effective', 'JJ'), ('machine',
'NN'), ('learning', 'NN'), ('techniques', 'NNS'), (',', ','), ('and', 'CC'), ('gain', 'NN'),
('practice', 'NN'), ('implementing', 'VBG'), ('them', 'PRP'), ('and', 'CC'), ('getting',
'VBG'), ('them', 'PRP'), ('to', 'TO'), ('work', 'VB'), ('for', 'IN'), ('yourself.', 'NNP'),
('More', 'NNP'), ('importantly', 'RB'), (',', ','), ('you', 'PRP'), ("'ll", 'MD'), ('learn',
'VB'), ('about', 'IN'), ('not', 'RB'), ('only', 'RB'), ('the', 'DT'), ('theoretical', 'JJ'),
('underpinnings', 'NNS'), ('of', 'IN'), ('learning', 'VBG'), (',', ','), ('but', 'CC'),
('also', 'RB'), ('gain', 'VBP'), ('the', 'DT'), ('practical', 'JJ'), ('know-how', 'NN'),
('needed', 'VBN'), ('to', 'TO'), ('quickly', 'RB'), ('and', 'CC'), ('powerfully', 'RB'),
('apply', 'RB'), ('these', 'DT'), ('techniques', 'NNS'), ('to', 'TO'), ('new', 'JJ'),
('problems.', 'NNP'), ('Finally', 'NNP'), (',', ','), ('you', 'PRP'), ("'ll", 'MD'), ('learn',
'VB'), ('about', 'IN'), ('some', 'DT'), ('of', 'IN'), ('Silicon', 'NNP'), ('Valley', 'NNP'),
("'s", 'POS'), ('best', 'JJS'), ('practices', 'NNS'), ('in', 'IN'), ('innovation', 'NN'),
('as', 'IN'), ('it', 'PRP'), ('pertains', 'VBZ'), ('to', 'TO'), ('machine', 'NN'),
('learning', 'NN'), ('and', 'CC'), ('AI', 'NNP'), ('.', '.')]
```

First Python program

- Interactive mode programming
  - When you enter into python IDE, you will see three greater signs. It means you are under the interactive mode.
  - Then you can write your code to let Python interpreter execute.
  - >>> print('Hello, Python!')

    Hello, Python! (this is what you see when you hit the enter)
  - This mode is useful for debugging.

First Python program

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Script mode programming (mostly used)
  - We will write a python program in a script (file). Python files have extension .py.
  - For example, test.py

```
#!/usr/bin/python
print "Hello, Python!"
x = 3
y = 5
sum = x+y
print('x+y=',sum )
```

  - We run this program under the command prompt as follows.

```
python test.py
You will see: Hello, Python
                    x+y= 8
```

The print statement

- String itself can also have ‘ or “
- We will use escape character (\)

Python escape characters:

For this	Use this	Setting x to:	Printing x will yield:
'	\'	'Don\'t do that'	Don't do that
"	\"	"She said \"hi\""	She said "hi"
\	\\	"Backslash: \\"	Backslash: \
[newline]	\n	"1\n2"	1 2
[carriage return]	\r	"1\r2"	2 overwrites the 1
[horizontal tab]	\t	"1\t2"	1 2
[backspace]	\b	"12\b3"	13
[16 bit unicode]	**u**xxxx	"Katakana a: \u30A1"	Katakana a: ア
[32 bit unicode]	**U**xxxxxxxx	"Katakana a: \u000030A1"	Katakana a: ア

The print statement

- print a string: string needs to be surrounded by double or single quotes
  - print("text") or print('text')
- print() (print a blank line)

The print statement

- print("African or 'European' swallows?")
  - African or 'European' swallows?

- print("Suppose two swallows \"carry\" it together.")
  - Suppose two swallows "carry" it together.

Comments

- Syntax:
  - # comment text (one line)
- test.py

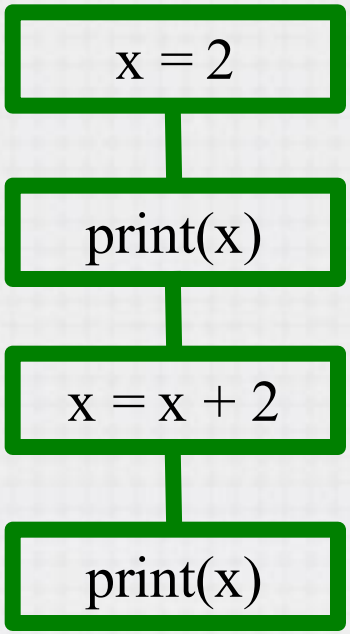
```
# Suzy Student, CSE 142, Fall 2097
# This program prints important messages.
print("Hello, world!")
Print("")                          # blank line
print("Suppose two swallows \"carry\" it together.")
print('African or "European" swallows?')
```

```
Output:
Hello, world!

Suppose two swallows "carry" it together.
African or European swallows?
```

Program steps or flow

- Like a recipe or installation instructions, a program is a sequence of steps to be done in order.
- Some steps are conditional - they may be skipped.
- Sometimes a step or group of steps are to be repeated.
- Sometimes we store a set of steps to be used over and over as needed several places throughout the program.

Sequential steps



Program:

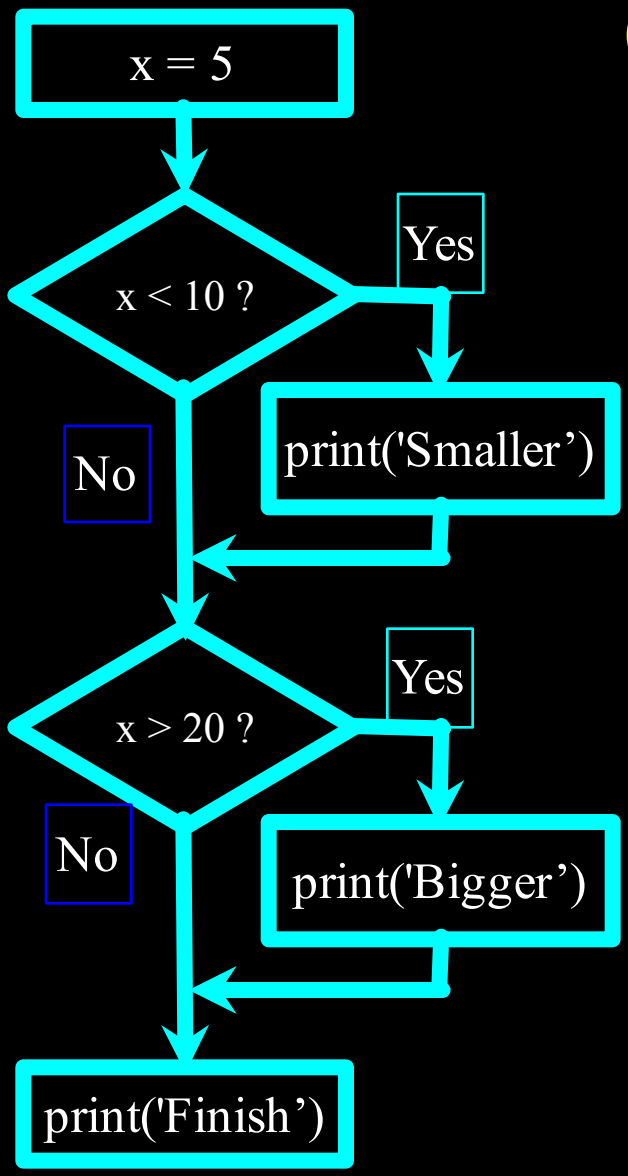
```
x = 2
print(x)
x = x + 2
print(x)
```

Output:

```
2
4
```

When a program is running, it flows from one step to the next. As programmers, we set up "paths" for the program to follow.

Conditional steps

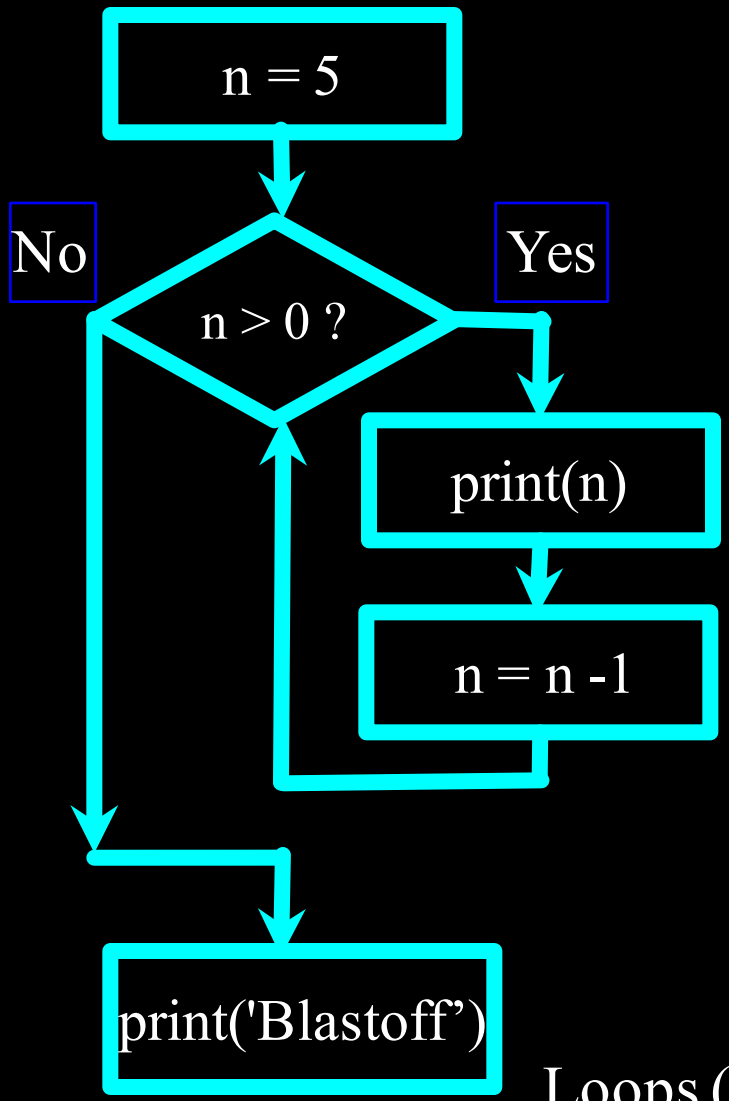


Program:

```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')

print('Finish')
```

Output:

```
Smaller
Finish
```



Repeated steps

Program:

```
n = 5
while n > 0 :
    print(n)
    n = n – 1
print('Blastoff!')
```

Output:

```
5
4
3
2
1
Blastoff!
```

Loops (repeated steps) have iteration variables that change each time through a loop. Often these iteration variables go through a sequence of numbers.

```
#!/usr/bin/python

name = input('Enter file:')
handle = open(name, 'r')
text = handle.read()
words = text.split()

counts = dict()
for word in words:
    counts[word] = counts.get(word,0) + 1
bigcount = None
bigword = None

for word,count in counts.items():
    if bigcount is None or count >
bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

Sequential

Repeated

Conditional

```
name = input('Enter file:')
handle = open(name, 'r')
text = handle.read()
words = text.split()
counts = dict()
for word in words:
    counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

A short Python "Story" about how to count words in a file

A word used to read data from a user

A sentence about updating one of the many counts

A paragraph about how to find the largest item in a list

Summary

- This is a quick overview of Introduction to Python
- We will revisit these concepts throughout the course
- Focus on the big picture