Requirements for Seminar

- Applications
  - Chrome (website: c9.io)

- GradQuant Resources
  - http://gradquant.ucr.edu/workshop-resources/

- Audience
  - No programing experience.
  - Never used Python.

Python Fundamentals

Part 1

Presented by GradQuant
Steven Jacobs

Acknowledgement:

Original Slides by Preston Carman

Based on:

Introduction to Python and programming

Michael Ernst

UW CSE 190p

Summer 2012

Who should attend?

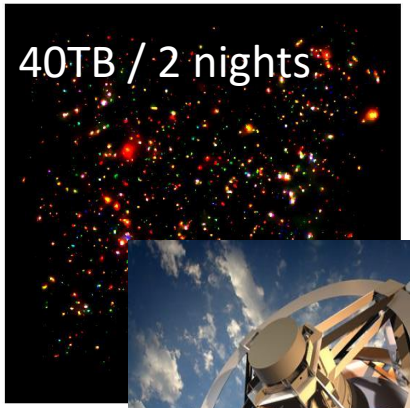
- No programing experience.
- Never used Python.

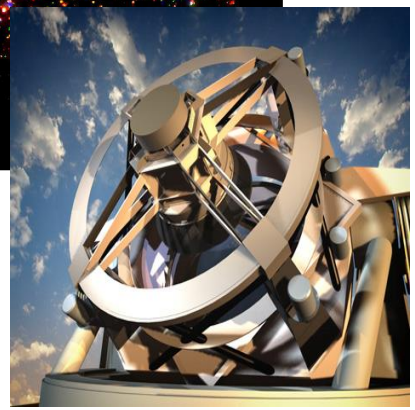
Objectives

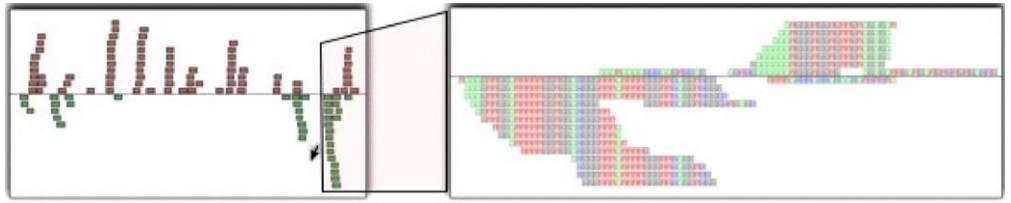
- Introduce Python programing concepts.
- Review Python syntax.
- Review available development tools.
- Create a python script.

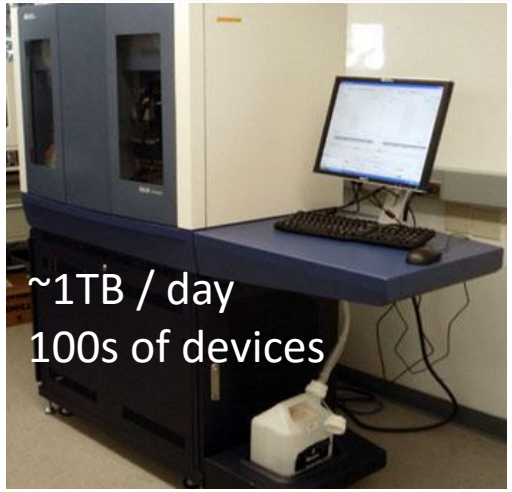
All of science is reducing to computational data manipulation

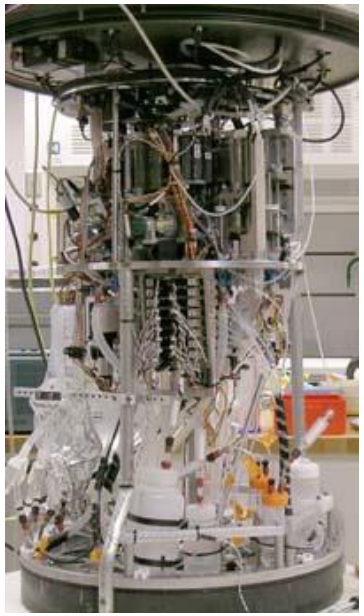
- Astronomy: High-resolution, high-frequency sky surveys (SDSS, LSST, PanSTARRS)
- Biology: lab automation, high-throughput sequencing,
- Oceanography: high-resolution models, cheap sensors, satellites

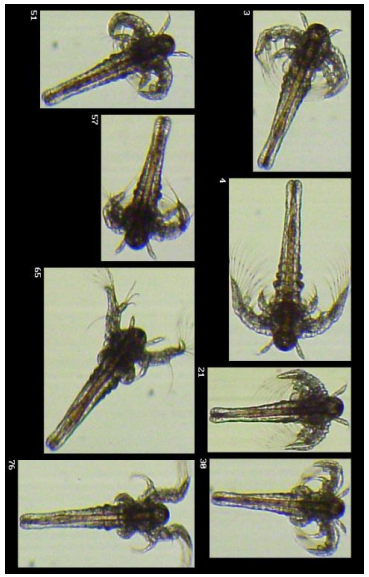












Example: Assessing treatment efficacy

	A	B	C	D	E	F	G	H	I	J
1	fu_2wk	fu_4wk	fu_8wk	fu_12wk	fu_16wk	fu_20wk	fu_24wk	total4type_fu	clinic_zip	pt_zip
2	1	3	4	7	9	9	9	12	98405	98405
3	2	4	6	7	8	8	8	8	98405	98403
4	0	0	[illegible]	[illegible]	[illegible]	0	0	[illegible]	98405	98445
5	3	4	[illegible]	[illegible]	[illegible]	5	5	[illegible]	98405	98332
6	0	0	[illegible]	[illegible]	[illegible]	0	0	[illegible]	[illegible]	[illegible]405
7	2	2	[illegible]	[illegible]	[illegible]	2	2	[illegible]	[illegible]	[illegible]402
8	1	2	5	6	8	10	10	14	98405	98418
9	1	1	2	2	2	2	2	2	98499	98406
10	0	0	1	2	2	2	2	6	98405	98404
11	0	0	0	0	0	0	0	0	98405	98402
12	1	1	2	2	4	4	4	4	98405	98405
13	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	98404	98404
14	2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	98499	98498
15	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	98499	98445
16	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	98499	98405
17	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	98499	98498
18	1	3	3	3	3	3	3	3	98499	98499
19	1	1	4	5	7	7	7	7	98499	98371

number of follow ups within 16 weeks after treatment enrollment.

Zip code of clinic

Zip code of patient

Question: Does the distance between the patient's home and clinic influence the number of follow ups, and therefore treatment efficacy?

Python program to assess treatment efficacy

```
# This program reads an Excel spreadsheet whose penultimate
# and antepenultimate columns are zip codes.
# It adds a new last column for the distance between those zip
# codes, and outputs in CSV (comma-separated values) format.
# Call the program with two numeric values:  the first and last
# row to include.
# The output contains the column headers and those rows.

# Libraries to use
import random
import sys
import xlrd        # library for working with Excel spreadsheets
import time
from gdapi import GoogleDirections

# No key needed if few queries
gd = GoogleDirections('dummy-Google-key')

wb = xlrd.open_workbook('mhip_zip_eScience_121611a.xls')
sheet = wb.sheet_by_index(0)

# User input:  first row to process, first row not to process
first_row = max(int(sys.argv[1]), 2)
row_limit = min(int(sys.argv[2]+1), sheet.nrows)

def comma_separated(lst):
  return ",".join([str(s) for s in lst])
```

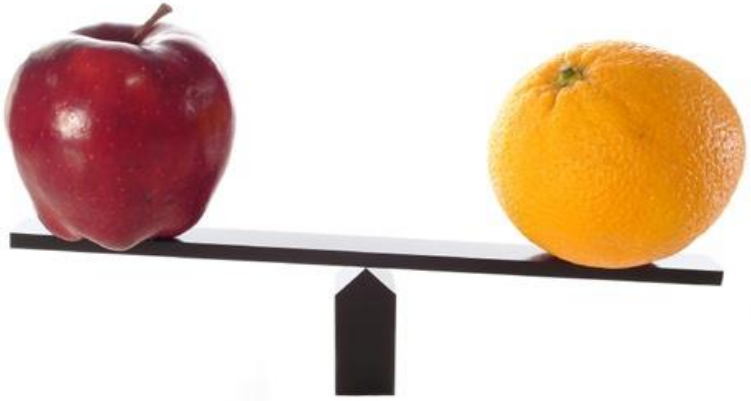
```
headers = sheet.row_values(0) + ["distance"]
print comma_separated(headers)

for rownum in range(first_row,row_limit):
   row = sheet.row_values(rownum)
   (zip1, zip2) = row[-3:-1]
   if zip1 and zip2:
      # Clean the data
      zip1 = str(int(zip1))
      zip2 = str(int(zip2))
      row[-3:-1] = [zip1, zip2]
      # Compute the distance via Google Maps
      try:
         distance = gd.query(zip1,zip2).distance
      except:
         print >> sys.stderr, "Error computing distance:", zip1,
zip2
         distance = ""
    # Print the row with the distance
    print comma_separated(row + [distance])
    # Avoid too many Google queries in rapid succession
    time.sleep(random.random()+0.5)
```

23 lines of code!

1. A variable contains a value

2. Python performs operations

3. Different types act differently



4. A program is a recipe

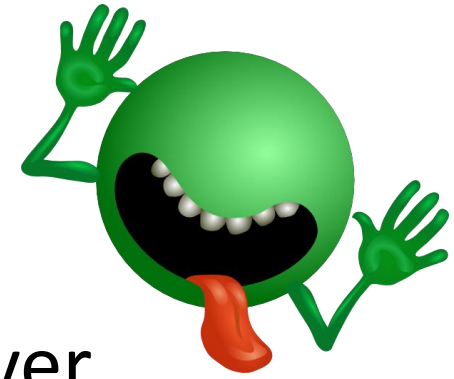
CORNBREAD

Colvin Run Mill Corn Bread

1 cup cornmeal
1 cup flour
½ teaspoon salt
4 teaspoons baking powder
3 tablespoons sugar
1 egg
1 cup milk
¼ cup shortening (soft) or vegetable oil

Mix together the dry ingredients. Beat together the egg, milk and shortening/oil. Add the liquids to the dry ingredients. Mix quickly by hand. Pour into greased 8x8 or 9x9 baking pan. Bake at 425 degrees for 20-25 minutes.

Don’t panic!

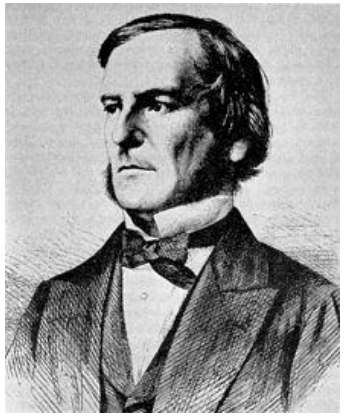


- This workshop is for people who have never programmed
  - (If you have programmed, you don’t belong here.)
- Ask questions!
  - This is the best way to learn

1. A variable contains a value

Types of values (4 basic types)

- Integers (**`int`**): **`-22, 0, 44`**
  - No decimal points
- Real numbers (**`float`**, for “floating point”): **`2.718, 3.1415`**
- Strings (**`str`**): **`"I love Python”`**
- Truth values (**`bool`**, for “Boolean”): **`True, False`**



George Boole

The Python Interpreter

- **Interactive interface to Python % python**

Python 2.5 (r25:51908, May 25 2007, 16:14:04)
[GCC 4.1.2 20061115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more
information. >>>

- **Python interpreter evaluates inputs: >>> 3*(7+2)**

27

Python prompts with ‘>>>’. To exit Python:

- CTRL-D

or type **exit()**

You type *expressions*. Python computes their *values*.

- 5
- 3+4
- 44/2
- 2**3 (what is a **?)
- 3*4+5*6
  - If precedence is unclear, use parentheses
- (72 – 32) / 9.0 * 5

Important: Integers vs Floats

- An operation on Integers will return an Integer
- An operation on Floats will return a Float
- What will each of these return?
- 12 / 4
- 13 / 4
- 13.0 / 4.0
- 13 / 4.0
- Modulo operator (for Integers)
- 13 % 4
- 12 % 4

Expressions

- **expression**: A data value or set of operations to compute a value.

  Examples: `1 + 4 * 3`

  `42`

- Arithmetic operators we will use:

`+ - * /`	addition, subtraction/negation, multiplication, division
`%`	modulus, a.k.a. remainder
`**`	exponentiation

- **precedence**: Order in which operations are computed.
  - `* / % **` have a higher precedence than `+ -`

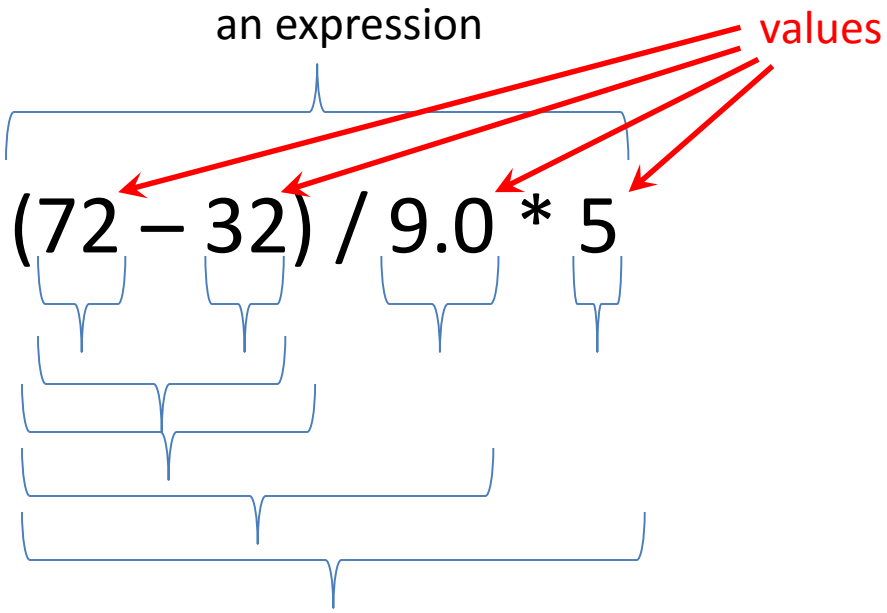
    `1 + 3 * 4` is `13`

  - Parentheses can be used to force a certain order of evaluation.

    `(1 + 3) * 4` is `16`

An expression is evaluated from the inside out

- How many expressions are in this Python code?



(**72** – **32**) / **9.0** * **5**

40 / **9.0** * **5**

4.44 * **5**

22.2

Assignment

- Now we have expressions that return values
- How do we store these values?
  - Variables
- Assignment Operator
  - X = 5
  - NOT an equality!
  - In Python, equality is represented as ==

Variables hold values

- To assign a variable, use "*variableName* = *expression*"

```
pi = 3.14
pi
Lost = 4815162342
Lost
22 = x                    # Error! Why?
```

- Not all variable names are permitted

Naming Rules

Names are case sensitive and cannot start with a number. They can contain letters, numbers, and underscores.

bob Bob _bob _2_bob_ bob_2 BOB

There are some reserved words:

and, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, **import**, in, is, lambda, not, or, pass, print, raise, return, try, while

Changing existing variables (“re-binding” or “re-assigning”)

```
x = 2 - 1
x
y = x
y
x = 5
x
y
```

Changing existing variables ("re-binding" or "re-assigning")

```
x = 2 - 1
x
y = x
y
x = 5
x
y
```

- "=" in an assignment is *not* a promise of eternal equality
- Evaluating an expression gives a new (copy of a) number, rather than changing an existing one

How an assignment is executed

1. Evaluate the right-hand side to a value
2. Store that value in the variable

```
x = 2
print x
y = x + 1
print y
x = 5
print y
z = x + 1
print x
print y
print z
```

State of the computer:

x: ~~2~~ 5

y: 3

z: 6

Printed output:

2
3
3
5
3
6

To visualize a program's execution:
http://people.csail.mit.edu/pgbovine/python/tutor.html

2. Python performs operations

Arithmetic Operations (Already seen)

```
22 * 10
22 / 10
22.0 / 10
3 ** 2
(5 +6) * (4 -3)

x = 3
y = x + 2
z = x + y

What about this?
z = 2
z - 5
z
```

More operations: Conditionals (return true/false)

Operator examples: **`not`**, **`and`**, **`or`**, **`<`**, **`>=`**, **`==`**, **`!=`**

```
22 > 4
22 < 4
22 == 4
x = 100                    # Assignment, not conditional!
x == 200
x == 100
22 = 4                     # Error!
x >= 5
not True
not (x >= 200)
3<4 and 7<6
4<3 or 5<6

temp = 72
is_liquid = temp > 32 and temp < 212
```

More operations: strings

A string represents text
Can use single or double quotes

```
'Python'
myName = "Steven"
"""
```

Operations:

- Length:

```
len(myName)
```

- Concatenation:

```
"Michael" + 'Ernst' #What will this do?
```

- More advanced: Containment/searching:

```
'ph' in myName #What do these return?
"v" in myName
```

Mathematical Operations

- Python has useful commands for performing calculations.

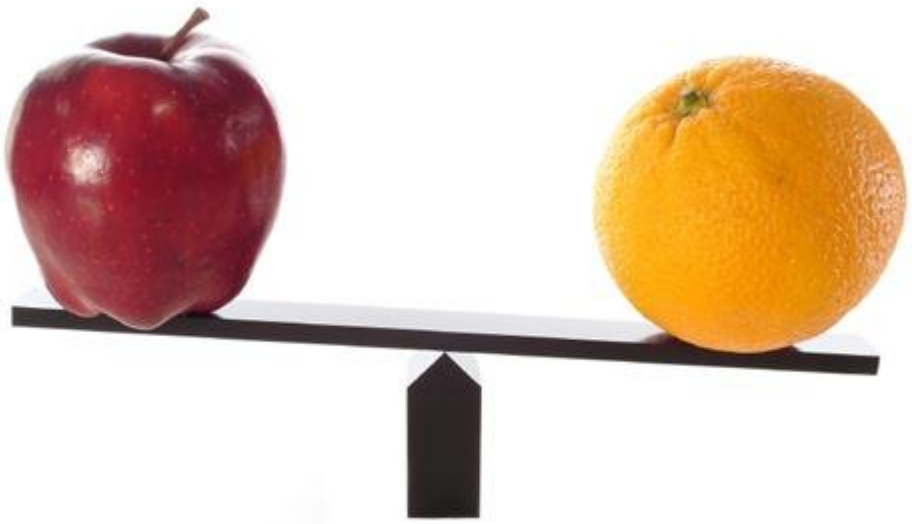
Command name	Description
`abs(`***value***`)`	absolute value
`ceil(`***value***`)`	rounds up
`cos(`***value***`)`	cosine, in radians
`floor(`***value***`)`	rounds down
`log(`***value***`)`	logarithm, base *e*
`log10(`***value***`)`	logarithm, base 10
`max(`***value1***`, `***value2***`)`	larger of two values
`min(`***value1***`, `***value2***`)`	smaller of two values
`round(`***value***`)`	nearest whole number
`sin(`***value***`)`	sine, in radians
`sqrt(`***value***`)`	square root

Constant	Description
`e`	2.7182818...
`pi`	3.1415926...

- To use many of these commands, you must write the following at the top of your Python program:

```
from math import *
```

3. Different types act differently



Operations behave differently on different types

```
3.0 + 4.0
3 + 4
3 + 4.0
"3" + "4"
3 + "4"                # Error
3 + True               # What will this do?
```

Moral: Python *sometimes* tells you when you do something that does not make sense.

Operations behave differently on different types

```
15.0 / 4.0
15 / 4
15.0 / 4
15 / 4.0
```

Type conversion:

```
float(15)
int(15.0)
int(15.5)
int("15")
str(15.5)
float(15) / 4
int(x)
```

4. A program is a recipe

CORNBREAD

Colvin Run Mill Corn Bread

1 cup cornmeal
1 cup flour
½ teaspoon salt
4 teaspoons baking powder
3 tablespoons sugar
1 egg
1 cup milk
¼ cup shortening (soft) or vegetable oil

Mix together the dry ingredients. Beat together the egg, milk and shortening/oil. Add the liquids to the dry ingredients. Mix quickly by hand. Pour into greased 8x8 or 9x9 baking pan. Bake at 425 degrees for 20-25 minutes.

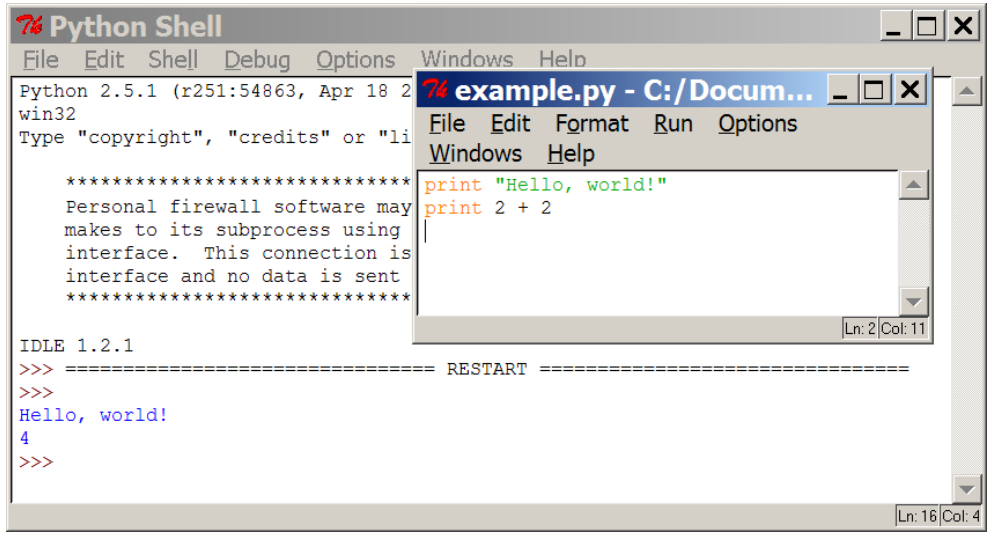
What is a program?

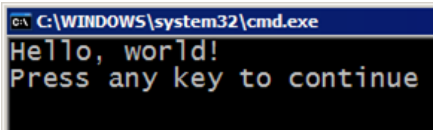
- A program is a sequence of instructions
- The computer executes one after the other, as if they had been typed to the interpreter
- Saving as a program is better than re-typing from scratch

```
x = input(‘Provide a value for x:’)
y = input (‘Provide a value for y:’)
z = x + y
print “x = “, x
print “y = “, y
print "The sum of", x, "and", y, "is", z
```

Programming basics

- **code** or **source code**: The sequence of instructions in a program.
- **syntax**: The set of legal structures and commands that can be used in a particular programming language.
- **output**: The messages printed to the user by a program.
- **console**: The place where the user interacts with the program
  - Some source code editors pop up the console as an external window, and others contain their own console window.

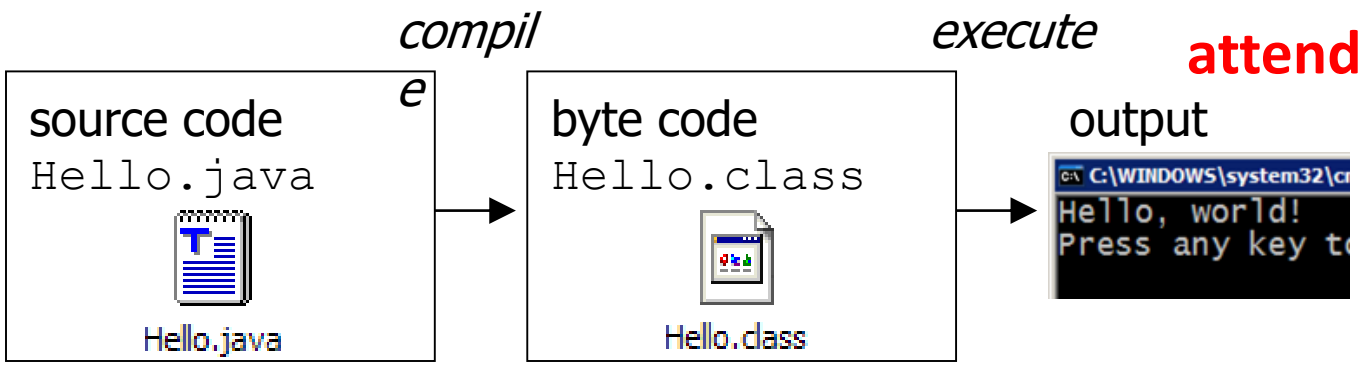




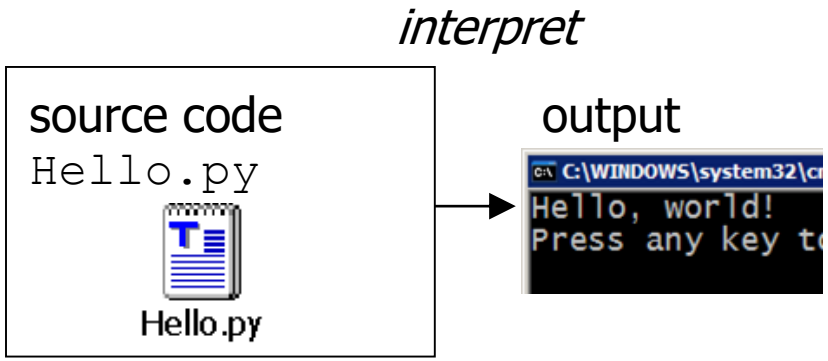
Compiling and interpreting

- Many languages require you to *compile* (translate) your program into a form that the machine understands.

For more on this, please attend the Java seminar!



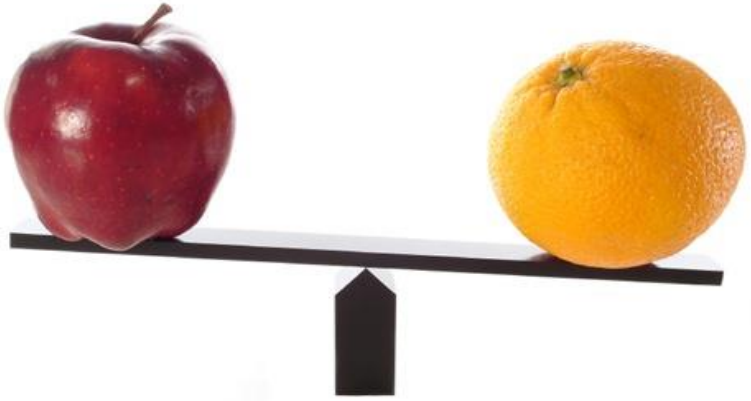
- Python is instead directly *interpreted* into machine instructions.



1. A variable contains a value

2. Python performs operations

3. Different types act differently



4. A program is a recipe

CORNBREAD

Colvin Run Mill Corn Bread

1 cup cornmeal
1 cup flour
½ teaspoon salt
4 teaspoons baking powder
3 tablespoons sugar
1 egg
1 cup milk
¼ cup shortening (soft) or vegetable oil

Mix together the dry ingredients. Beat together the egg, milk and shortening/oil. Add the liquids to the dry ingredients. Mix quickly by hand. Pour into greased 8x8 or 9x9 baking pan. Bake at 425 degrees for 20-25 minutes.

Half-time!

Running Programs on UNIX

% python filename.py

Comments

- Start comments with # – the rest of line is ignored.
- Can include a “documentation string” as the first line of any new function or class that you define.

```
# This is a comment
```

Import statement

- import allows a Python script to access additional modules
- Modules
  - sys: stdin, stderr, argv
  - os: system, path
  - string: split
  - re: match compile
  - math: exp, sin, sqrt, pow

Software

Online Development

- Cloud 9 (online editor)
  - http://c9.io
  - http://bit.ly/1KIJcEU

Local Development

- Python
  - http://www.python.org
- PyCharm (editor)
  - http://www.jetbrains.com/pycharm/

- Features
  - Free versions
  - Multiplatform

Exercise 1:

```
#get inputs from the user
x = input(‘Provide a value for x:’)
y = input (‘Provide a value for y:’)

#calculate output
z = x + y

#print results to the user
print “x = “, x
print “y = “, y
print "The sum of", x, "and", y, "is", z
```

Exercise 2: Fahrenheit to Celsius:

How could we take as input from the user a Fahrenheit temperature, and then convert it to Celsius?

Mathematical Equation for Celsius:

$(F - 32) \times 5/9$

Think about:

Input and output

Integers vs Floats

Exercise 2:

```
#get inputs from the user
F = input(‘Provide the temperature in
Fahrenheit:’)

#calculate output
#make sure you maintain floats!
#try C = (F-32) * 5 / 9
C = (F - 32) * 5.0 / 9.0

#print results to the user
print "The temperature in Celsius is", C
```

Exercise 3 (If Statement):

```
"if" provides a means of checking whether some condition is met.
Tabs are used to show what should run if the condition is met

if (5 < 6):
       print "five is less than six"

if (x == "banana"):
       print "x is banana"

if (y <= z):
       print "y is less than or equal to z"
       print "therefore I cannot choose the wine in front of me"
```

Exercise 3 (If Statement):

```
Have the user input a number. If this number is
greater than 1000, output a message “Wow that is
a big number!”
```

Exercise 3 (If Statement):

```
#get inputs from the user
x = input('Provide a value:')

#print results to the user
if (x > 1000):
      print "Wow that is a big number!"




*ALTERNATIVELY:
if (1000 < x):
      print "Wow that is a big number!"
```

Exercise 4 (else if):

```
else if provides a means to check alternate conditions:

Consider this code:

if (x < 5):
       print "x is pretty small"
if (x < 10):
       print "x is average"
if (x < 15):
       print "x is large"
if (x >= 15):
       print "x is huge"
```

Exercise 4 (else if):

```
else if provides a means to check alternate conditions:

Consider this code:

if (x < 5):
       print "x is pretty small"
elif (x < 10):
       print "x is average"
elif (x < 15):
       print "x is large"
else:
       print "x is huge"
```

Exercise 4 (else if):

```
Let's make a text-based adventure!

First line should be this:
x = raw_input('You are trapped with five dragons.(A)run (B)fight (C)make friends:')

You should output a unique message based on whether the user types A, B, or C

How do you handle when a user types something else?
```

Exercise 4 (else if):

```
#get inputs from the user
x = raw_input('You are trapped with five dragons.(A)run (B)fight (C)make friends:')

#print results to the user
if (x == "A"):
      print "You cannot escape. You die!"
elif (x == "B"):
      print "You cannot win. You die!"
elif (x == "C"):
      print "They do not want to be friends. You die!"
elif (x == "cheat"):
      print "You found the way to cheat. You win!"
else:
      print "Invalid choice. You die"
```

Moving Forward...

```
There are many more tools available in Python that we can't cover here.

If you want to move forward, the next things to look at would be:
While loops
Incrementing variables
For loops
Reading/Writing files
```

Python Editors

- Eclipse with PyDev
  - http://pydev.org/
- Sublime Text
  - http://www.sublimetext.com/
- PyCharm
  - http://www.jetbrains.com/pycharm/

- Why use a python editor
  - Syntax Highlighting
  - Error Detection
  - Auto-completion

Which Python?

- **Python 2.7**
- Current version on Cloud 9, so we'll use it
- Last commonly used release before version 3
- Implements some of the new features in version 3, but fully backwards compatible
- **Python 3**
- Released a few years ago
- Many changes (including incompatible changes)
- More existing third party software is compatible with Python 2 than Python 3 right now

Resources

- Python's website
  - http://www.python.org/
- Python Tutorial - Codecademy
  - http://www.codecademy.com/tracks/python
- GradQuant Resources
  - http://gradquant.ucr.edu/workshop-resources/
- Google
  - Search for "python …"
- Stack Overflow website
  - http://stackoverflow.com/

GradQuant

- One-on-one Consultations
  - Make appointment on the website
  - http://gradquant.ucr.edu
- More Seminars on Programming
  - Data Manipulation with Python
  - Advanced Python (Offered this quarter)
  - Intro to SQL (Offered this quarter)
  - Advanced Java (Offered next quarter)
    - http://gradquant.ucr.edu/workshop-resources/

Remember to fill out the seminar survey. Thank you!