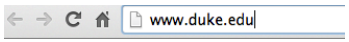
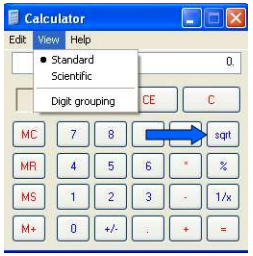
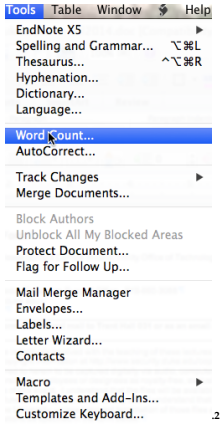
Plan for the week: Week 2, Sept 1-5

- Understanding program structure
  - Defining, testing, calling functions
  - How to run a program, how someone runs yours
- Understanding more of Python the language
  - Types: string, int, bool, float
  - Operations on these, e.g., +, %, [:], and
  - Control: conditionals and loops (Thursday)
- Course structure: APTs, labs, assignments
  - Tools for enabling structure

Examples of functions





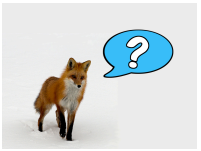


Functions explained

- In a calculator, sqrt: number in -> number out
  - What is domain, what is range?
- In MSWord, word count: document -> number
  - Domain is word doc, range is integer
- In browser, web: URL -> HTML formatted "page"
  - Domain is valid URL, range is HTML resources
- In Python we see similar structure!

Abstracting over code: functions

- http://goo.gl/DfcPgI
- See snarf for class work as well
- These functions do not return values, they print
  - Illustrates problem decomposition, but …
  - Normally have each function return a value
  - Normally use the return value in function call



Part of http://goo.gl/DfcPgl (and snarf)

```
def eieio():
  print "Ee-igh, Ee-igh, oh!"

def refrain():
  print "Old MacDonald had a farm,",
  eieio()

def had_a(animal):
  print "And on his farm he had a",animal,",",
  eieio()
```

Lots of commas

Anatomy and Dissection of Print

- Print generates output to a console, window, …
  - Depends on how program invoked
  - Basically used for: help with debugging and creating output for copy/paste, view

```
print "hello",x,"what's up",y
```

- Space inserted between comma-separated items
  - Can use string concatentation, `"hello"+str(x)`
  - If statement ends with comma, no newline
  - Print anything that has a string representation…

Tracing program execution

- The `def` statement defines a function
  - Creates a name that can be used in program
  - Name encapsulates program statements, creates its own environment for running code
    - Variables, parameters, *local* to the function
- Function name and statements part of Python execution environment
  - Can *call* or *invoke* the function
  - If *parameters* needed, must *pass* values for each
  - Visualize program execution: PythonTutor, brain

Abstraction over barnyards

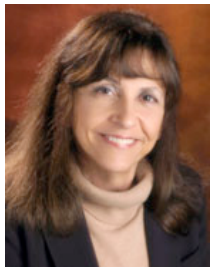
- In OldMacPrint we have `pig()` and `fox()` …
  - What's the same in these? What's different?
  - Capture differences in parameters/variables
- Create new function:
  - `def verse(animal, noise)`
- Look at pig() and fox() create new function
  - Call: `verse("horse", "neigh")`
  - Call: `verse("cow", "moo")`

http://bit.ly/101fall14-0902-1

Nancy Leveson: Software Safety

- Mathematical and engineering aspects, invented the discipline
  - Health care software
  - MS Word, Airbus 360, …

  "There will always be another software bug; never trust human life solely on software" huffington post?



- Therac 25: Radiation machine
  - http://en.wikipedia.org/wiki/Therac-25, http://bit.ly/5qOjoH
- Software and steam engines

Compsci 101: Running Python

- What does it mean to run a program?
  - What does clicking on app/program do?
  - How do you run/test APT code, other Python code
    - Where does program start to execute/run?
- Control flow in Python
  - Loops and if statements --- coming on Thursday
  - *Essential* for writing *real* programs
    - But function calls are part of control flow

Functions that return values

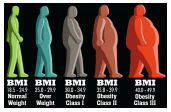
- Most functions return values
  - Example in Old MacDonald is "different"
  - Some claim: all functions return values

```
def inch2centi(inches):
    return 2.54*inches

xh = inch2centi(72)
```

```
def pluralize(word):
    return word + "es"

pf = pluralize("fish")
```


What is an APT? BMI APT

- Automated/Algorithmic Problem Testing
  - Write one function, 2-30 lines, solve a problem
  - Tested automagically in Eclipse or the browser
  - Test test test … Quality of code not an issue
- Start simple, build toward more complex
  - What is a function? A function call?
  - What is a parameter? Argument?
  - How do you run/execute a program



How to solve an APT

- Two very, very, very important steps
  1. How to solve the problem without computer
     Paper, Pencil, (Calculator)
  2. How to translate problem-solving to Python
- Both steps can be hard, vocabulary and language are initially a real barrier
  - More Python experience, easier step 2 becomes
  - With experience, step 2 can influence step 2
- Step 1 is key, without it you won't get anywhere

Anatomy of a Python function

```
def name(params):
    body
```

- Define a function: name, parameters, body
  - How do we decide on these?
  - Do we need parameters?
  - What does body of function do
- Functions provide a named abstraction over code
  - Huh? `math.factorial(5)` `"hello".upper()`

Functions: BMI (Body Mass Index)

- What is formula? How to use it?
  - For one person can simply print the BMI
    - Make sure units are correct, formula right
  - What if we want to validate data?
  - What if we want to notify folks who might need guidance?

call replaced by return value, why use function?

```
def bmi(weight, height):
    return 703.07 * weight/(height*height)
if bmi(170,72) < 18.5:
  print "underweight"
```


What does return statement do?

- Programs execute one line at a time
  - After one statement finishes, the next executes
  - Calling a function causes its code to execute
    - What happens in the code that calls the function?
- The value returned *replaces* the function call
  - `print math.sqrt(25.0)`
  - `if bmi(170,72) < 18.5: print "underweight"`
- What if nothing returned?
  - `None` by default in Python

Re-use: Counting words in file

```
def word_count(filename):
    f = open(filename)
    all = f.read()
    words = all.split()
    return len(words)
if __name__ == "__main__":
  name = "/data/romeo.txt"
  print "# words in",name,
  print "=",wordCount(filename)
```


Running Python Program/Module

- Python is an interpreter, platform specific
  - So is Java, so is Android, … contrast compilers
  - Python can execute a .py file, need a "launch point"
- Convention in Python and other languages
  - Start with section labeled __main__, that's run

```
if __name__ == "__main__":
   statements here
   statements here
```

- Boilerplate, don't memorize, let Eclipse do work!

Function calls: what is an API?

http://www.enotes.com/shakespeare-quotes/vasty-deep

Eclipse Particulars

- Supports many languages: we care about Python
  - PyDev perspective: Windows>Open Perspective>Other>…
  - Also use console: Windows>Show View>Console
  - Use PyDev console (right click console icon)
- Creating projects, Python Module
  - Illustrated with examples in class
- Submitting and check via Ambient
  - Illustrated with examples in class

A-Z, Soup to Nuts, APT all the way

- Where do we find what APTs are due this week?
  - Web pages, Sakai v Google v bookmark
- Testing code for APTs supplied by 101 staff
  - Snarf the project that provides testing harnass
  - Don't call us, ETester.py will call you (your code)
- Refresh to see results.html
  - Repeat until finished
- Submit using Ambient, Duke CS Eclipse plugin

Summary of Today

- Functions help in program/problem decomposition
  - Each function does one thing only
  - Functions typically return values
    - Song printing functions don't, they print
- Names, parameters, arguments, return values
  - Functions execute, return replaces call point
  - Calling code picks up and continues after call
- We'll see loops and conditionals on Thursday

Grace Murray Hopper (1906-1992)

- "third programmer on world's first large-scale digital computer"
  - US Navy: Admiral

"It's better to show that something can be done and apologize for not asking permission, than to try to persuade the powers that be at the beginning"

- ACM Hopper award given for contributions before 35

2010: Craig Gentry: http://www.youtube.com/watch?v=qe-zmHoPW30
2011: Luis von Ahn
2013: Pedro Felzenszwalb

Duke Compsci: Grace Hopper 2013