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**

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 Tuesday

- **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"
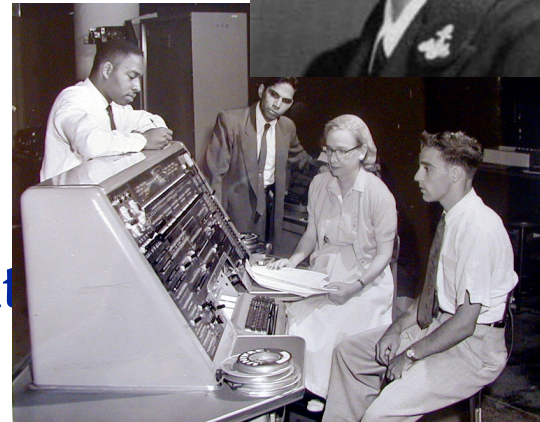
- ACM Hopper award given for contributions before 35

2010: Craig Gentry: http://www.youtube.com/watch?v=qe-zmHoPW30

2011: Luis von Ahn

2013: Pedro Felzenszwalb

Duke Compsci: Grace Hopper 2013

Python review

- **We have several types to store data/values**
  - ➢ Different types for different purposes
  - ➢ Still need to explore how to use these types, what operations can be used with each
  - ➢ Types: int, float, string, bool, list, file

- **We need to learn how to put types/values together into programs/code:**
  - ➢ Function was first step toward doing this
  - ➢ Need more

Anatomy of a Python float

- **A float is a floating point number**
  - ➢ Internally doesn't have infinite precision,
  - ➢ Floats have arithmetic operations: *, /, +, -, **
- **Floats**
  - ➢ There are largest, smallest floats, expressed in terms of exponents, e.g., 1.79e+308, 2.22e-308
    - Typically not an issue in Compsci 101
  - ➢ Don't compare f == g with floats
    - Precision issues

Anatomy of a Python String

- **String is a sequence of characters**
  - ➢ **Functions apply to sequences: len, slice [:], others**
  - ➢ **Methods applied to strings [specific to strings]**
    - • st.split(), st.startswith(), st.strip(), st.lower(), …
    - • st.find(), st.count()
- **Strings are *immutable* sequences**
  - ➢ **Characters are actually length-one strings**
  - ➢ **Cannot change a string, can only create new one**
    - • What does .upper() do?
  - ➢ **See resources for functions/methods on strings**
- ***Iterable*: Can loop over it, *Indexable*: can slice it**

Anatomy of Python List

- **String is a sequence of characters**
  - ➢ **Immutable, cannot change, but can copy**
  - ➢ **Lists are mutable**
- **List is a sequence of values/objects**
  - ➢ **['apple', 3.145, 45, True]**
  - ➢ **Indexable, like a string, using [:] and []**
  - ➢ **We'll see it's iterable too – loop over**

- **Simple, but powerful way to structure data**
  - ➢ **Internal to a program, not like a file: external**

Indexable summary

- [0] is first element of string or list or indexable
  - If length of string is 7: 0,1,2,3,4,5,6 for indexes
  - History of zero-indexing in computer science
  - String access/read, List access/read/write
  - [-1] is the last element

- [:] is a slice, returns a new sequence
  - [a:b] is start at a, up-to but not including b
  - [:x] starts at 0 and [x:] goes to end
  - [a:b:c] has a stride/step of c

APT Interlude

http://bit.ly/101fall14-0902-2

Some simple computational problems

- **How does calendar know it's a leap year?**
  - ➢ **Are all leap years hard-wired in?**
  - ➢ **Does each February determine "am I leap year"?**

- **Readability metric: what level is this story?**
  - ➢ **Syllables, words, sentences, …**
  - ➢ **http://en.wikipedia.org/wiki/Readability_test**

- **Student home-town data: where do you live?**
  - ➢ **Who is close, far, more**

What years are leap years?

- **2000, 2004, 2008, ...**
  - ➤ But not 1900, not 2100, yes 2400!
  - ➤ Yes if divisible by 4, but not if divisible by 100 unless divisible by 400! (what?)

```
def is_leap_year(year):
    if year % 400 == 0:
        return True
    if year % 100 == 0:
        return False
    if year % 4 == 0:
        return True
    return False
```

- **There is more than one way to skin a cat, but we need at least one way**

Python if statements and Booleans

- **In python we have if: else: elif:**
  - ➢ **Used to guard or select block of code**
  - ➢ **If guard is True then, else other**

- **What type of expression used in if/elif tests?**
  - ➢ **==, <=, <, >, >=, !=, and, or, not, in**
  - ➢ **Value of expression must be either True or False**
  - ➢ **Type == bool, George Boole, Boolean,**

- **Look at more examples**

Three versions of is_vowel

```python
def is_vowel(ch):
    if ch =='e':
        return True
    if ch == 'a':
        return True
    if ch == 'i':
        return True
    if ch == 'o':
        return True
    if ch == 'u':
        return True
    return False
```

```python
def is_vowel(ch):
    if ch in "aeiou":
        return True
    else:
        return False
```

```python
def is_vowel(ch):
    return "aeiou".count(ch) > 0
```

Lynn Conway

See Wikipedia and lynnconway.com
- **Joined Xerox Parc in 1973**
  - ➢ **Revolutionized VLSI design with Carver Mead**

- **Joined U. Michigan 1985**
  - ➢ **Professor and Dean, retired '98**

- **NAE '89, IEEE Pioneer '09**

- **Helped invent dynamic scheduling early '60s IBM**
  - ➢ **Transgender, fired in '68**

Data interlude

- **Exploring what we can do with latitude and longitude, websites, APIs, simple Python scripts**
  - ➢ **Sometimes when data is about us it's …**

- **We'll use batchgeo.com to create a visual**
  - ➢ **Copy/paste, see what happens?**
  - ➢ **Download into Excel and repeat?**

- **Who travels the greatest distance to Duke?**
  - ➢ **At least where are they from, if not who**

Visualizing and Analyzing Data

- **Sometimes data is dirty**
  - ➢ We clean it. By hand, or with scripts/programs
  - ➢ There are data cleaning libraries (what's that?)

- **For more in-depth analysis need other tools**
  - ➢ Compsci course Everything Data
  - ➢ Develop your own, use Python!
  - ➢ Sometimes need statistics, sometimes need artistic/aesthetic skills

Data analyzed with Python

- **Open file of data in csv format**
  - ➢ **Where do we get this? Why edit first?**
- **Loop over file, separate each line**
  - ➢ **Convert string to list, index to get parts**

- **Find code to determine distance using (lat, long)**
  - ➢ **Google is your friend, what's the query?**

http://bit.ly/101fall14-0904-data

Simple loops, more later

```
for x in "abcdefg":
    code
for ch in ['a','b','c']:
    code
for line in file:
    code
```

- **As with if, def, the : separates body**
  - ➢ **In Python indentation is important**
  - ➢ **Loop repeats body once for each IN element**