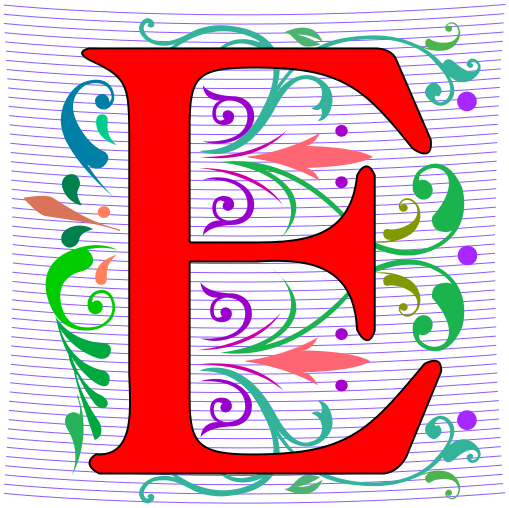
Compsci 101, Indexing, Slicing, Selection

Owen Astrachan
Kristin Stephens-Martinez
January 30, 2018

`E` is for …

- **Escape Sequence**
  - Why `\n` is newline and `\\s` is whitespace
- **Encryption**
  - From Caesar Ciphers to SSL and beyond
- **Enumerate**
  - Adding counters to iterable
- **Emoticon**
  -  



PFTW

- **Review Pancake, Selection, Problem Solving**
  - From idea to all-green
- **Review Totem Assignment**
  - Function definitions, calls, parameters
- **Sequences: Strings and Lists**
  - Indexing, Slicing, Combining
- **Lists and Loops (Thursday)**
  - Iteration and Appending

Pancake Concepts/Tools

- **Solve by hand – required before programming!**
  - What are edge cases? Identify and solve
    - Zero pancakes, # pancakes < size of pan, …
- **Write down steps? Maybe good idea for many**
  - Translate ideas into code? Or words into code!
- **New and Review Python**
  - // and % and if …

Three versions of is_vowel

```
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
```

```
def is_vowel(ch):
    c = "aeiou".count(ch)
    if c > 0:
        return True
    else:
        return False
```

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

Finishing the APT Pancake Problem

- **Use the algorithm developed AFTER verifying with pencil and paper aka by hand**
  - Test first in our Eclipse (VM or Local)
  - Create main program block
  - Call **`minutesNeeded`** with several examples

- **Use APT testing system, submit, reflect**
  - Can we have more than one return statement?

Pancake Code/Algorithm

- There are some special cases not shown below
  - Understand `//` and `%` operators for int values
- Testing locally and in APT tester

```
fullPans = numCakes//capacity
leftOver = numCakes % capacity
if leftOver ==0:
    return fullPans*10
if leftOver <= capacity/2:
    return fullPans*10+5
return(fullPans+1)*10
```

How to teach pancake flipping

- http://www.youtube.com/watch?v=W_gxLKSsSIE
  - Is this computer science? http://bit.ly/zykOrh
  - For longer, more complex robotic tasks
    - http://www.youtube.com/watch?v=4usoE981e7I

- **Do robots matter?**
  - Do they dream?
  - Self-driving cars?
  - Machine learning?

Selection Summarized

- We can use selection: if statement
  - **`if boolean_condition: block`**
  - What can change? Boolean and block

- What is a boolean condition? True/False
  - See **`type(3 < 5)`**
  - Relational operators: **`< <= > >= == !=`**
  - Boolean operators: **`and or not`**

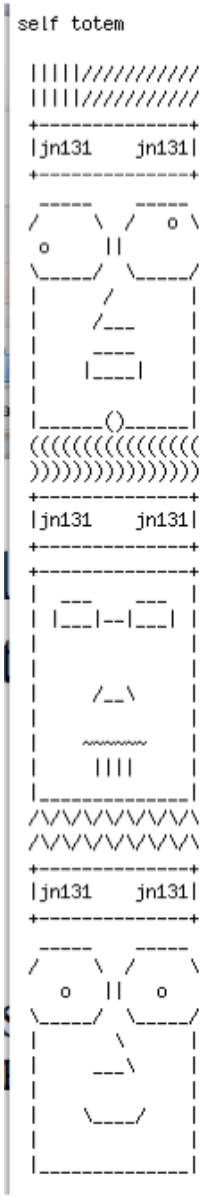
Sir Tim Berners-Lee

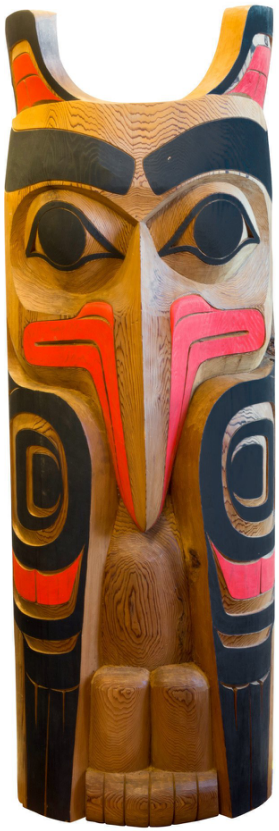
- Turing award 2016
  - World Wide Web
- HTTP vs. TCP/IP
  - Just protocols?

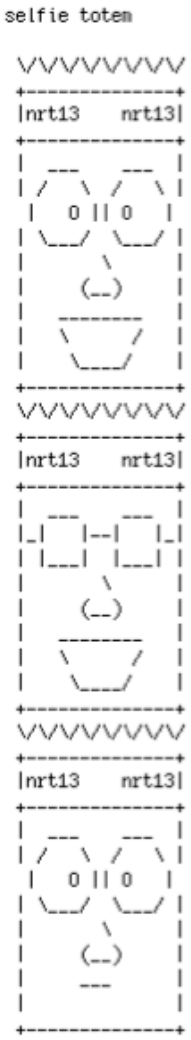
I want you to realize that, if you can imagine a computer doing something, you can program a computer to do that.

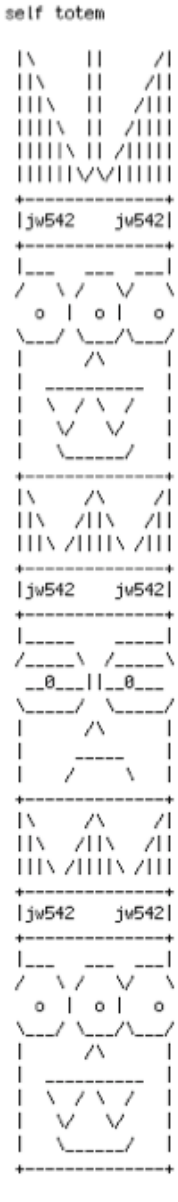
Unbounded opportunity... limited only by your imagination. And a couple of laws of physics.

Creating Selfies









Learning Goals: Totem

- **Understand differences and similarities:**
  - Function definitions, function calls
  - Functions that return values, those that don't
  - Functions with parameters, those without

- **Be creative and learn lesson(s) about software design and engineering**
  - Create a small, working program, make incremental improvements.
  - Read the directions and understand specifications!

Terminology and Progress

- What is **`def eyes_crossed():`**
  - Does this return a value? Type?
  - How do you call the function?

- What is **`def head_happy():`**
  - Where do you call it? What does it return?

- Let's examine a whole head function from write-up

Anatomy of a whole head

- What does doc-string mean/do? What is it?
  - What are the function calls here?
  - What about with a different nose?

```
def head_funny():
    """
    Print a head that looks a little funny,
    With surprised mouth and eyebrows
    """
    print(hair_plain())
    print(eyes_withbrows())
    print(nose_big())
    print(mouth_surprised())
    print(chin_plain())
```

Anatomy of a whole head

- How similar is this to **`head_funny`?**
  - What are differences? Find and parameterize!

```
def head_sad():
    """
    Print a head that looks a little sad,
    With sad mouth and eyebrows
    """
    print(hair_plain())
    print(eyes_withbrows())
    print(nose_big())
    print(mouth_sad())
    print(chin_plain())
```

Parameterize Whole Head

- Call **`head_funny(mouth_surprised)`**
  - What is call? What is parameter?
- Can't write call below, why? Small difference!!
  - **`head_funny(mouth_surprised())`**

```
def head_funny(mouth_func):
    """
    Print a head that looks a little funny,
    With surprised mouth and eyebrows
    """
    print(hair_plain())
    print(eyes_withbrows())
    print(nose_big())
    print(mouth_func())
    print(chin_plain())
```

Parameterize Whole Head

- What does a *totem function* look like?
  - Why does **`head_funny`** not have return value?
  - What type is parameter to **`head_funny`**?
  - Names, types, values!

```
def totem_fixed():
    """
    totem poles with different mouths
     """
    head_funny(mouth_surprised)
    head_funny(mouth_angry)
    head_funny(mouth_tired)
```

WOTOTOTEM

http://bit.ly/101spring18-jan30-1

- Read carefully, build slowly
- Create a running program and ensure it does run

Additional Tools for Building

- **We need sequences to access and create data**
  - Documents rather than words
  - Spread sheets rather than single data/formula

- **Operations on these sequences**
  - Access all elements: loops
  - Access elements selectively: if statements

Python Sequences

- Types String and List share characteristics
  - Both are sequences, have a length
  - Both support indexing and slicing
  - Conversion functions help connect them

- **`x="hello world" y=["hello", "world"]`**
  - What is **`len(x)`**?
  - String is immutable, list is mutable

Indexing Python Sequences

- **`x="hello world" y=["hello", "world"]`**
- Indexing provides access to individual elements
  - Compare **`x[0]`** and **`y[0]`**
  - Start with 0, what is last valid positive index?

  - Compare **`x[-1]`** and **`y[-1]`**
  - What is negative index of second to last element?

  - Index **`-n`** is the same as index **`len(seq) - n`**

Slicing Python Sequences

- **`x="hello world"`**
- **`y=["my", "big", "beautiful", "world"]`**
- Slicing provides sub-sequence (string or list)
  - Compare **`x[0:3]`** and **`y[0:3]`**
  - What is length of subsequence? **`seq[2:4]`**

  - Compare **`x[4:-1]`** and **`y[2:-1]`**
  - Is last index part of subsequence?
- We can omit value, e.g., x[2:] or x[:4], good shortcut!

One more APT

- Let's have Brunch with the Digerati
- https://www2.cs.duke.edu/csed/pythonapt/portmanteau.html

- Read the problem statement carefully
  - Why can't we create netizen in function? Combination of "internet" and "citizen"

- How do we solve with slicing?

Anatomy of Python String

- String is a sequence of characters
  - Strings cannot be changed: *immutable*
  - Strings parameters to built-in functions: **`len`**
  - Operators can be applied: **`[n]`** and **`[m:n]`**

- Methods/functions applied ON strings
  - **`"HELLO".lower()`**
  - **`"the duke way".split()`**

Split and List Preview

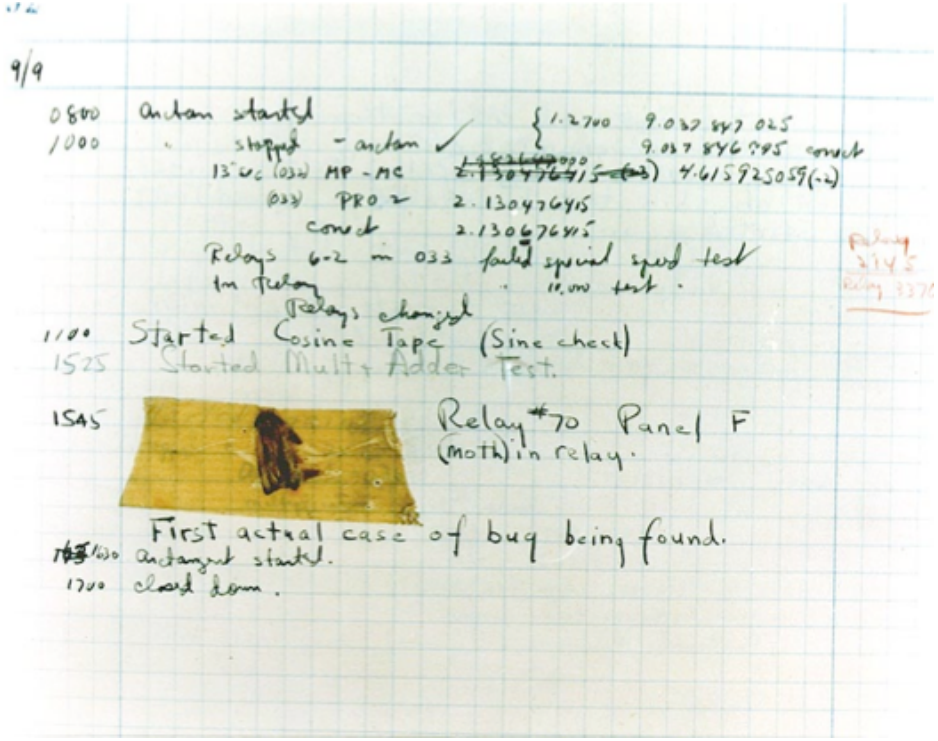
- **Lists can be heterogenous *sequence***
  - Strings, ints, lists, anything
  - Contrast String: *sequence* of characters

- **Lists can grow: read from file for example**
  - Contrast String: immutable/cannot change

- **`"one fish two fish".split()` is a list**
  - How many elements? See Lab this week!

Bug and Debug

- software 'bug'
- Start small
  - Easier to cope
- Judicious 'print'
  - Debugger too



- Verify the approach being taken, test small, test frequently, add to working code
  - What does “working code” mean?

if, else, elif, oh my!

- Leap years, ugh, but …
  - Show how statement order matters, trace it!
  - https://en.wikipedia.org/wiki/Leap_year

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

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

Wikipedia Leap Algorithm

- See algorithm section 
  - https://en.wikipedia.org/wiki/Leap_year

```
def is_leap_(year):
    if year % 4 != 0:
        return False          # not leap
    elif year % 100 != 0:     # 1968
        return True
    elif year % 400 != 0:
        return False          #1968
    else
      return True              #2000
```

Slicing WOTO

http://bit.ly/101spring18-jan25-2

- Correctness counts!