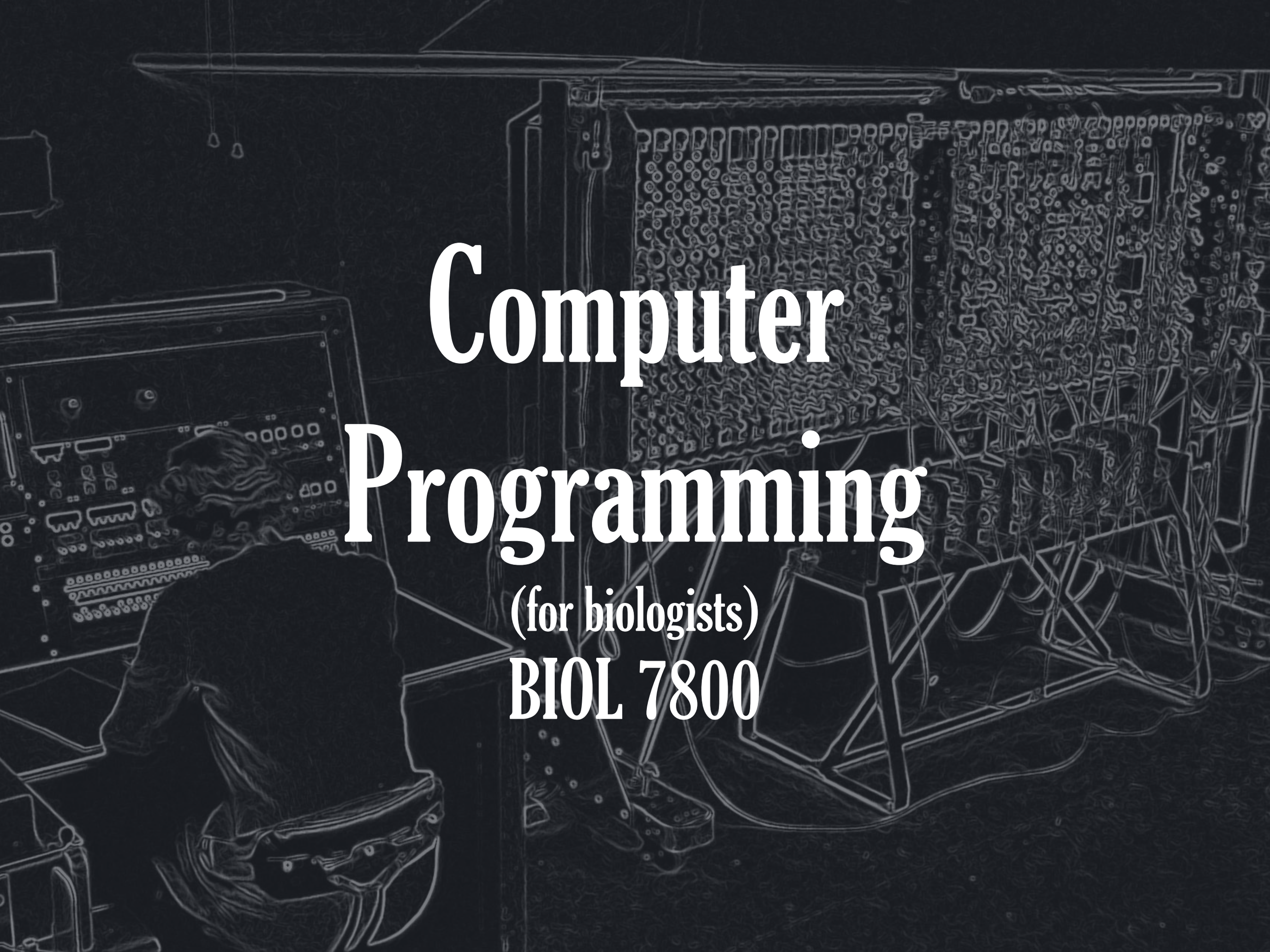


Computer Programming

(for biologists)

BIOL 7800

What is computer programming?

1. zsh

```
bcf at brant-4 in ~
$ cowsay -f turtle "programming?" | lolcat

 ______________
< programming? >
 --------------
    \
     \
      \
                    ___-------___
                _-~~             ~~-_
             _-~                    /~-_
          /^\__/^\         /~  \   /    \
        /|  O|| O|        /      \_______________/        \
        | |___||__|      /       /                \          \
        |          \    /      /                    \          \
        |   (______) /______/                        \_________ \
        |         / /         \                      /           \
         \         \^\\         \                  /             \     /
           \         ||           \______________/      _-_       //\__//
             \       ||------_-~~-_ -------------- \ --/~   ~\    || __/
               ~-----||====/~     |==================|       |/~~~~~
                (_(__/  ./     /                    \_\      \.
                       (_(___/                         \_____)_)

bcf at brant-4 in ~
$
```

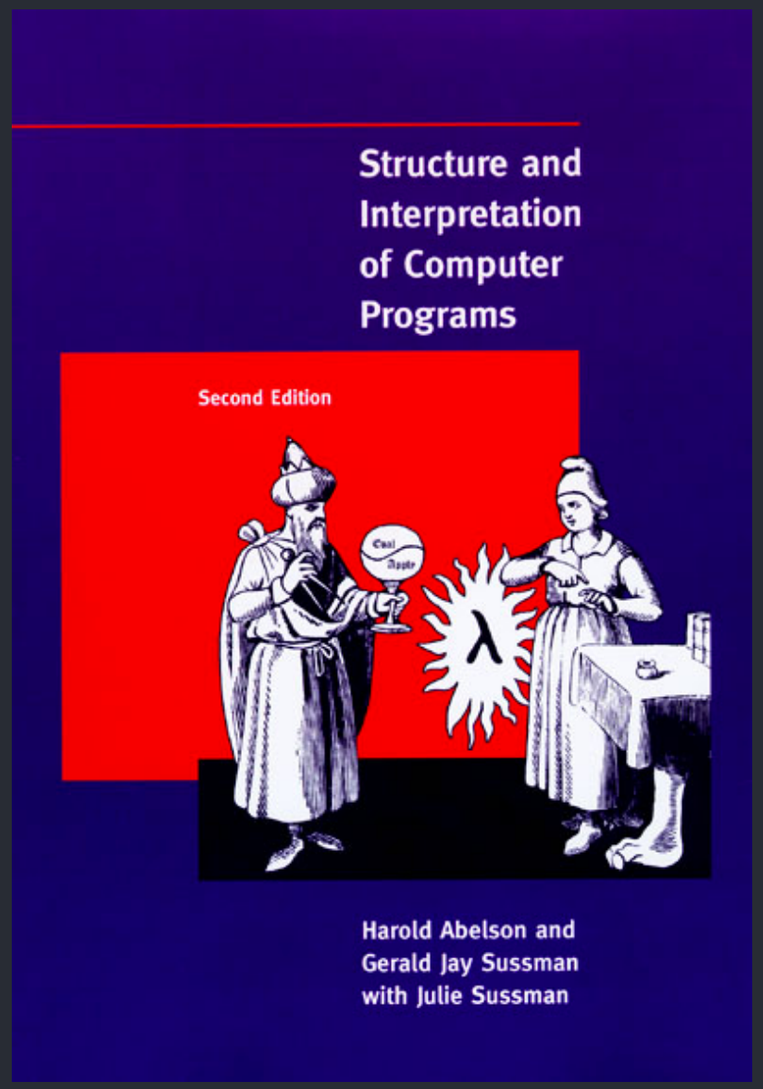
What is computer programming?

Computer programming
(often shortened to `*programming*`)
is a process that leads from an original formulation of a computing problem to executable computer programs.

- Wikipedia

Ada Lovelace
(1815 - 1852)

What is computer programming?



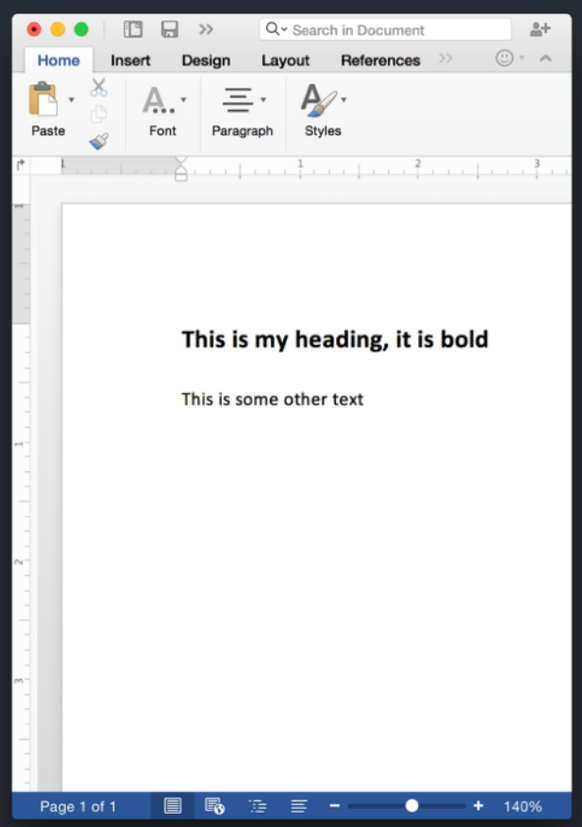
"We are about to study the idea of a **computational process**. Computational processes are abstract beings that inhabit computers. As they evolve, processes manipulate other abstract things called data. The evolution of a process is directed by a pattern of rules called a program. People create programs to direct processes. In effect, we conjure the spirits of the computer with our spells."

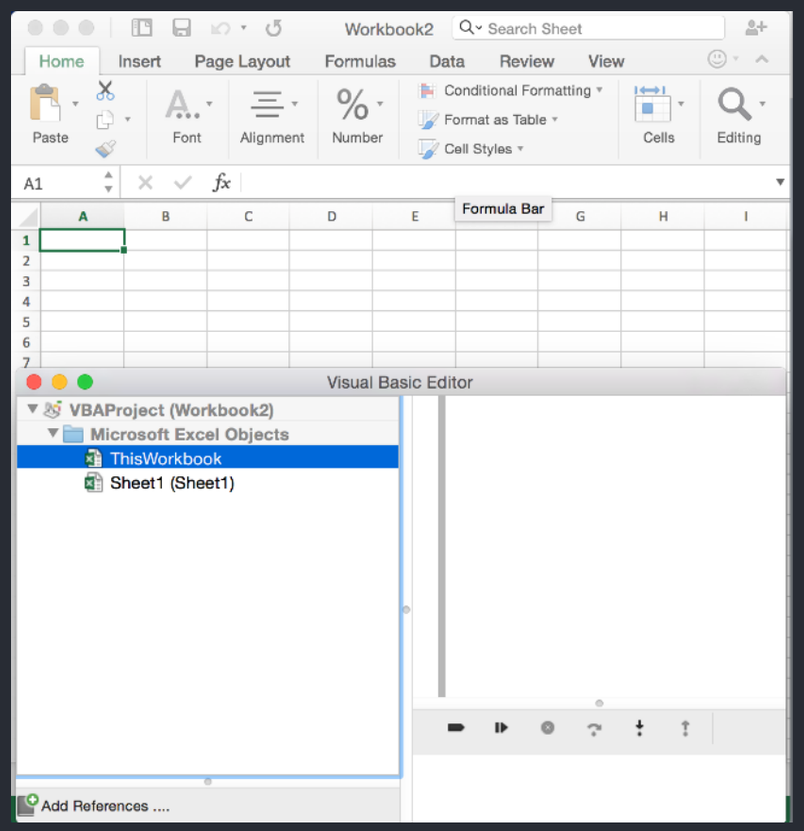
- Intro to Chapter 1

What is computer programming?

Providing a logical sequence of arguments to a computer so that it can perform a desired task

<u>Several "levels"</u>






```
27  def _mean_and_variance_pop_n(values):
28      n = 0
29      s = 0.0
30      ss = 0.0
31      for v in values:
32          n += 1
33          s += v
34          ss += v*v
35      if n == 0:
36          raise IndexError("values in mean_and_
37      mean = float(s)/n
38      var = (ss - mean*s)/n
39      return mean, var, n
40
41  def mean_and_population_variance(values):
42      """Returns the mean and population varia
43      elements in values once."""
44      return _mean_and_variance_pop_n(values)[
45
46  def mean_and_sample_variance(values):
47      """Returns the mean and sample variance
48      elements in values once."""
49      mean, pop_var, n = _mean_and_variance_po
```

Application level

Macro-level

General purpose level

What is computer programming?

2. exit (zsh)

```
bcf at brant-4 in ~
$ cowsay -f stegosaurus "Is using the command line programming?" | lolcat
 ___________________________________________
< Is using the command line programming? >
 -------------------------------------------
\                             .       .
 \                           / `.   .' "
  \                  .---.  <    > <    >  .---.
   \                 |    \  \ - ~ ~ - /  /    |
         _____          ..-~             ~-..-~
        |     |   \~~~\.'                    `./~~~/
       ---------   \__/                         \__/
      .'  O    \     /               /       \  "
     (_____,    `._.'               |         } \/~~~/
      `----.         /      }       |        /    \__/
            `-.      |      /       |       /      `. ,~~|
                ~-.__|     /_ - ~ ^|      /- _      `..-'
                     |    /         |     /     ~-.     `-. _  _  _
                     |____|         |____|         ~ - . _ _ _ _ _>

bcf at brant-4 in ~
$
```

General purpose level

Programming Language

The **“dialect”** that we use to provide our instructions to the computer

TCSH

C++

COBOL

ZSH

R

C

BASH

Ruby

C#

Python

D

Perl

FORTRAN

JAVA

Mother Tongues

Tracing the roots of computer languages through the ages

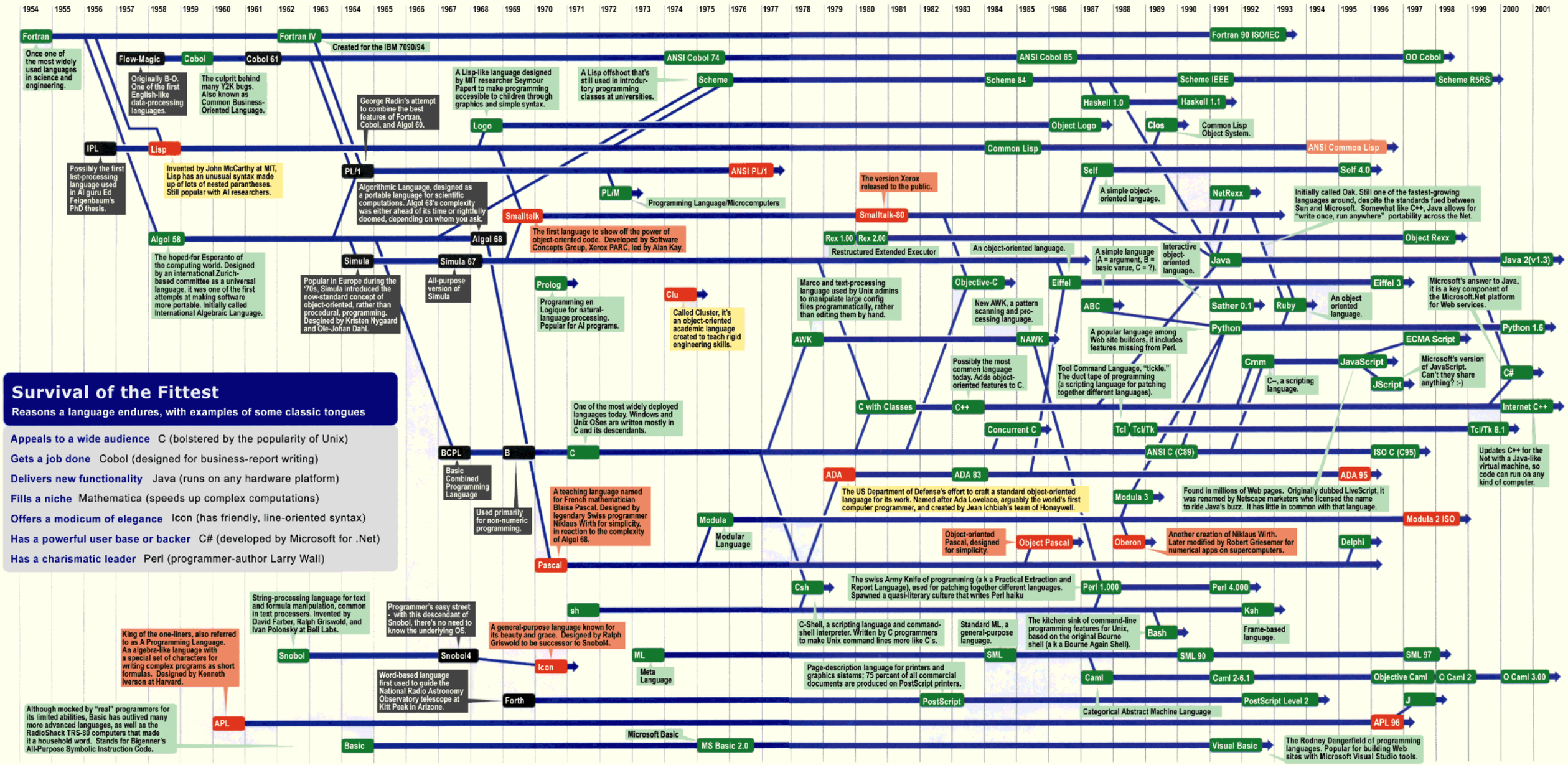
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers-electronic lexicographers, if you will-aim to save, or at least document the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua frangas. Among the most endangered are Ada, APL, B (the predecessor of C), Lsp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working with the Computer History Musuem in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social, and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an utter failure." Here's a peek at the strongest branches of programming's family tree. For a nearly exhaustive rundown, check out the Language List at *HTTP://www.informatik.uni-freiburg.de/Java/misc/lang_list.html*. - Michael Mendeno

Key

- 1954 Year Introduced
- **Active:** thousands of users
- **Protected:** taught at universities; compilers available
- **Endangered:** usage dropping off
- **Extinct:** no known active users or up-to-date compilers
- **Lineage continues**



Survival of the Fittest

Reasons a language endures, with examples of some classic tongues

- **Appeals to a wide audience** C (bolstered by the popularity of Unix)
- **Gets a job done** Cobol (designed for business-report writing)
- **Delivers new functionality** Java (runs on any hardware platform)
- **Fills a niche** Mathematica (speeds up complex computations)
- **Offers a modicum of elegance** Icon (has friendly, line-oriented syntax)
- **Has a powerful user base or backer** C# (developed by Microsoft for .Net)
- **Has a charismatic leader** Perl (programmer-author Larry Wall)

Sources: Paul Boutin; Brent Hailpern, associate director of computer science at IBM Research; The Retrocomputing Museum; Todd Proebsting, senior researcher at Microsoft; Gio Wiederhold, computer scientist, Stanford University

General purpose level

Programming Language

The **“dialect”** that we use to provide our instructions to the computer

Four classes of Programming Language

machine code

assembly language

high-level language

very high-level language

```
/usr/bin/less:     file format elf64-x86-64


Disassembly of section .init:

0000000000401590 <_init>:
  401590:       48 83 ec 08             sub    $0x8,%rsp
  401594:       e8 b3 04 00 00          callq  401a4c <time@plt+0x44>
  401599:       e8 42 05 00 00          callq  401ae0 <time@plt+0xd8>
  40159e:       e8 fd 42 01 00          callq  4158a0 <error+0x3720>
  4015a3:       48 83 c4 08             add    $0x8,%rsp
  4015a7:       c3                      retq

Disassembly of section .plt:

00000000004015a8 <tcsetattr@plt-0x10>:
  4015a8:       ff 35 3a ec 21 00       pushq  0x21ec3a(%rip)        # 6201e8 <_fini+0x20a910>
  4015ae:       ff 25 3c ec 21 00       jmpq   *0x21ec3c(%rip)        # 6201f0 <_fini+0x20a918>
  4015b4:       0f 1f 40 00             nopl   0x0(%rax)

00000000004015b8 <tcsetattr@plt>:
  4015b8:       ff 25 3a ec 21 00       jmpq   *0x21ec3a(%rip)        # 6201f8 <_fini+0x20a920>
  4015be:       68 00 00 00 00          pushq  $0x0
  4015c3:       e9 e0 ff ff ff          jmpq   4015a8 <_init+0x18>

00000000004015c8 <fileno@plt>:
  4015c8:       ff 25 32 ec 21 00       jmpq   *0x21ec32(%rip)        # 620200 <_fini+0x20a928>
  4015ce:       68 01 00 00 00          pushq  $0x1
  4015d3:       e9 d0 ff ff ff          jmpq   4015a8 <_init+0x18>

00000000004015d8 <close@plt>:
  4015d8:       ff 25 2a ec 21 00       jmpq   *0x21ec2a(%rip)        # 620208 <_fini+0x20a930>
  4015de:       68 02 00 00 00          pushq  $0x2
  4015e3:       e9 c0 ff ff ff          jmpq   4015a8 <_init+0x18>

00000000004015e8 <__longjmp_chk@plt>:
  4015e8:       ff 25 22 ec 21 00       jmpq   *0x21ec22(%rip)        # 620210 <_fini+0x20a938>
  4015ee:       68 03 00 00 00          pushq  $0x3
  4015f3:       e9 b0 ff ff ff          jmpq   4015a8 <_init+0x18>

00000000004015f8 <ioctl@plt>:
  4015f8:       ff 25 1a ec 21 00       jmpq   *0x21ec1a(%rip)        # 620218 <_fini+0x20a940>
  4015fe:       68 04 00 00 00          pushq  $0x4
  401603:       e9 a0 ff ff ff          jmpq   4015a8 <_init+0x18>

0000000000401608 <tgetflag@plt>:
  401608:       ff 25 12 ec 21 00       jmpq   *0x21ec12(%rip)        # 620220 <_fini+0x20a948>
  40160e:       68 05 00 00 00          pushq  $0x5
  401613:       e9 90 ff ff ff          jmpq   4015a8 <_init+0x18>

0000000000401618 <nl_langinfo@plt>:
  401618:       ff 25 0a ec 21 00       jmpq   *0x21ec0a(%rip)        # 620228 <_fini+0x20a950>
  40161e:       68 06 00 00 00          pushq  $0x6
  401623:       e9 80 ff ff ff          jmpq   4015a8 <_init+0x18>

0000000000401628 <__fprintf_chk@plt>:
  401628:       ff 25 02 ec 21 00       jmpq   *0x21ec02(%rip)        # 620230 <_fini+0x20a958>
  40162e:       68 07 00 00 00          pushq  $0x7
  401633:       e9 70 ff ff ff          jmpq   4015a8 <_init+0x18>

0000000000401638 <isatty@plt>:
  401638:       ff 25 fa eb 21 00       jmpq   *0x21ebfa(%rip)        # 620238 <_fini+0x20a960>
  40163e:       68 08 00 00 00          pushq  $0x8
```

machine code

set of instructions that are run directly by the central processing unit

* architecture specific

objdump -d /usr/bin/less | less

```
/usr/bin/less:     file format elf64-x86-64


Disassembly of section .init:

0000000000401590 <_init>:
  401590:       48 83 ec 08             sub    $0x8,%rsp
  401594:       e8 b3 04 00 00          callq  401a4c <time@plt+0x44>
  401599:       e8 42 05 00 00          callq  401ae0 <time@plt+0xd3>
  40159e:       e8 fd 42 01 00          callq  4158a0 <error+0x3720>
  4015a3:       48 83 c4 08             add    $0x8,%rsp
  4015a7:       c3                      retq

Disassembly of section .plt:

00000000004015a8 <tcsetattr@plt-0x10>:
  4015a8:       ff 35 3a ec 21 00       pushq  0x21ec3a(%rip)        # 6201e8 <_fini+0x20a910>
  4015ae:       ff 25 3c ec 21 00       jmpq   *0x21ec3c(%rip)        # 6201f0 <_fini+0x20a918>
  4015b4:       0f 1f 40 00             nopl   0x0(%rax)

00000000004015b8 <tcsetattr@plt>:
  4015b8:       ff 25 3a ec 21 00       jmpq   *0x21ec3a(%rip)        # 6201f8 <_fini+0x20a920>
  4015be:       68 00 00 00 00          pushq  $0x0
  4015c3:       e9 e0 ff ff ff          jmpq   4015a8 <_init+0x18>

00000000004015c8 <fileno@plt>:
  4015c8:       ff 25 32 ec 21 00       jmpq   *0x21ec32(%rip)        # 620200 <_fini+0x20a928>
  4015ce:       68 01 00 00 00          pushq  $0x1
  4015d3:       e9 d0 ff ff ff          jmpq   4015a8 <_init+0x18>

00000000004015d8 <close@plt>:
  4015d8:       ff 25 2a ec 21 00       jmpq   *0x21ec2a(%rip)        # 620208 <_fini+0x20a930>
  4015de:       68 02 00 00 00          pushq  $0x2
  4015e3:       e9 c0 ff ff ff          jmpq   4015a8 <_init+0x18>

00000000004015e8 <__longjmp_chk@plt>:
  4015e8:       ff 25 22 ec 21 00       jmpq   *0x21ec22(%rip)        # 620210 <_fini+0x20a938>
  4015ee:       68 03 00 00 00          pushq  $0x3
  4015f3:       e9 b0 ff ff ff          jmpq   4015a8 <_init+0x18>

00000000004015f8 <ioctl@plt>:
  4015f8:       ff 25 1a ec 21 00       jmpq   *0x21ec1a(%rip)        # 620218 <_fini+0x20a940>
  4015fe:       68 04 00 00 00          pushq  $0x4
  401603:       e9 a0 ff ff ff          jmpq   4015a8 <_init+0x18>

0000000000401608 <tgetflag@plt>:
  401608:       ff 25 12 ec 21 00       jmpq   *0x21ec12(%rip)        # 620220 <_fini+0x20a948>
  40160e:       68 05 00 00 00          pushq  $0x5
  401613:       e9 90 ff ff ff          jmpq   4015a8 <_init+0x18>

0000000000401618 <nl_langinfo@plt>:
  401618:       ff 25 0a ec 21 00       jmpq   *0x21ec0a(%rip)        # 620228 <_fini+0x20a950>
  40161e:       68 06 00 00 00          pushq  $0x6
  401623:       e9 80 ff ff ff          jmpq   4015a8 <_init+0x18>

0000000000401628 <__fprintf_chk@plt>:
  401628:       ff 25 02 ec 21 00       jmpq   *0x21ec02(%rip)        # 620230 <_fini+0x20a958>
  40162e:       68 07 00 00 00          pushq  $0x7
  401633:       e9 70 ff ff ff          jmpq   4015a8 <_init+0x18>

0000000000401638 <isatty@plt>:
  401638:       ff 25 fa eb 21 00       jmpq   *0x21ebfa(%rip)        # 620238 <_fini+0x20a960>
  40163e:       68 08 00 00 00          pushq  $0x8
```

assembly language

a low-level abstraction of machine code that corresponds very strongly to machine code instructions

architecture specific

objdump -d /usr/bin/less | less

```c
#include <unistd.h>

#include <math.h>
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdint.h>
#include <limits.h>
#include "axml.h"
#include <stdint.h>
#include <xmmintrin.h>
#include <pmmintrin.h>
#include <immintrin.h>

#ifdef _FMA
#include <x86intrin.h>
#define FMAMACC(a,b,c) _mm256_fmadd_pd(b,c,a)
#endif

extern const unsigned int mask32[32];

const union __attribute__ ((aligned (BYTE_ALIGNMENT)))
{
  uint64_t i[4];
  __m256d m;

} absMask_AVX = {{0x7fffffffffffffffULL, 0x7fffffffffffffffULL, 0x7f



static inline __m256d hadd4(__m256d v, __m256d u)
{
  __m256d
    a, b;

  v = _mm256_hadd_pd(v, v);
  a = _mm256_permute2f128_pd(v, v, 1);
  v = _mm256_add_pd(a, v);

  u = _mm256_hadd_pd(u, u);
  b = _mm256_permute2f128_pd(u, u, 1);
  u = _mm256_add_pd(b, u);

  v = _mm256_mul_pd(v, u);

  return v;
}
```

high-level language

An (strong) abstraction from machine code or assembly language that is more similar to “natural” language

raxml: avxLikelihood.c

```python
#! /usr/bin/env python

##############################################################################
##  DendroPy Phylogenetic Computing Library.
##
##  Copyright 2010-2015 Jeet Sukumaran and Mark T. Holder.
##  All rights reserved.
##
##  See "LICENSE.rst" for terms and conditions of usage.
##
##  If you use this work or any portion thereof in published work,
##  please cite it as:
##
##     Sukumaran, J. and M. T. Holder. 2010. DendroPy: a Python library
##     for phylogenetic computing. Bioinformatics 26: 1569-1571.
##
##############################################################################

"""
Functions to calculate some general statistics.
"""

import math
from dendropy.calculate import probability
from operator import itemgetter

def _mean_and_variance_pop_n(values):
    n = 0
    s = 0.0
    ss = 0.0
    for v in values:
        n += 1
        s += v
        ss += v*v
    if n == 0:
        raise IndexError("values in mean_and_variance cannot be empty")
    mean = float(s)/n
    var = (ss - mean*s)/n
    return mean, var, n

def mean_and_population_variance(values):
    """Returns the mean and population variance while only passing over the
    elements in values once."""
    return _mean_and_variance_pop_n(values)[:2]

def mean_and_sample_variance(values):
    """Returns the mean and sample variance while only passing over the
    elements in values once."""
    mean, pop_var, n = _mean_and_variance_pop_n(values)
    if n == 1:
        return mean, float('inf')
    samp_var = n*pop_var/(n-1)
    return mean, samp_var

```

very high-level language

An (even stronger) abstraction from machine code or assembly language that is very similar to natural language

dendropy: statistics.py

```c
static inline __m256d hadd4(__m256d v, __m256d u)
{
  __m256d
    a, b;

  v = _mm256_hadd_pd(v, v);
  a = _mm256_permute2f128_pd(v, v, 1);
  v = _mm256_add_pd(a, v);

  u = _mm256_hadd_pd(u, u);
  b = _mm256_permute2f128_pd(u, u, 1);
  u = _mm256_add_pd(b, u);
```

```python
def _mean_and_variance_pop_n(values):
    n = 0
    s = 0.0
    ss = 0.0
    for v in values:
        n += 1
        s += v
        ss += v*v
    if n == 0:
        raise IndexError("values in mean_and_variance cannot be empty")
    mean = float(s)/n
    var = (ss - mean*s)/n
    return mean, var, n
```

High-level language

compiled

code is translated to machine code before execution

memory management

statically typed

faster (10 - 100x speedup)

harder

C, C++, BASIC, pascal

Very high-level language

interpreted

code is “interpreted” and translated on-the-fly

garbage collection

dynamically typed

slower

easier

Python, Perl, R, Ruby

Why Python?

object-oriented, interpreted, garbage-collected language

"general-purpose" programming language

syntax is more like natural language

formatting rules (PEP8) help legibility

can use many programming paradigms
(procedural, functional, object-oriented)

```python
import math
from dendropy.calculate import probability
from operator import itemgetter

def _mean_and_variance_pop_n(values):
    n = 0
    s = 0.0
    ss = 0.0
    for v in values:
        n += 1
        s += v
        ss += v*v
    if n == 0:
        raise IndexError("values in mean_and_variance cannot be empty")
    mean = float(s)/n
    var = (ss - mean*s)/n
    return mean, var, n

def mean_and_population_variance(values):
    """Returns the mean and population variance while only passing over the
    elements in values once."""
    return _mean_and_variance_pop_n(values)[:2]

def mean_and_sample_variance(values):
    """Returns the mean and sample variance while only passing over the
    elements in values once."""
```

Why not R?

R is a domain specific language (specialized for a particular problem)

R is (debatably) ugly

R is slow (again, debatably)

R is (more) confusing…

But which language you prefer is really up to you!

That said, we're using Python…

```python
#! /usr/bin/env python

##############################################################################
##  DendroPy Phylogenetic Computing Library.
##
##  Copyright 2010-2015 Jeet Sukumaran and Mark T. Holder.
##  All rights reserved.
##
##  See "LICENSE.rst" for terms and conditions of usage.
##
##  If you use this work or any portion thereof in published work,
##  please cite it as:
##
##     Sukumaran, J. and M. T. Holder. 2010. DendroPy: a Python library
##     for phylogenetic computing. Bioinformatics 26: 1569-1571.
##
##############################################################################

"""
Functions to calculate some general statistics.
"""

import math
from dendropy.calculate import probability
from operator import itemgetter

def _mean_and_variance_pop_n(values):
    n = 0
    s = 0.0
    ss = 0.0
    for v in values:
        n += 1
        s += v
        ss += v*v
    if n == 0:
        raise IndexError("values in mean_and_variance cannot be empty")
    mean = float(s)/n
    var = (ss - mean*s)/n
    return mean, var, n

def mean_and_population_variance(values):
    """Returns the mean and population variance while only passing over the
    elements in values once."""
    return _mean_and_variance_pop_n(values)[:2]

def mean_and_sample_variance(values):
    """Returns the mean and sample variance while only passing over the
    elements in values once."""
    mean, pop_var, n = _mean_and_variance_pop_n(values)
    if n == 1:
        return mean, float('inf')
    samp_var = n*pop_var/(n-1)
    return mean, samp_var
```

Python is not perfect

packaging is still awful

Python 2 to Python 3

slow

"global interpreter lock"

one thread runs a time

python™ 2

October 16, 2000

python™ 3

December 3, 2008

print is a statement

2. ipython (ssh)

```
bcf at ketchup in ~
$ ipython
Python 2.7.9 |Continuum Analytics, Inc.| (default, Mar  9 2015, 16:20:48)
Type "copyright", "credits" or "license" for more information.

IPython 1.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: print "Hellow world"
Hellow world

In [2]:
```

print is a function

3. bash

```
bcf at brant-4 in ~
$ ipython
Python 3.5.1 |Anaconda 2.4.1 (x86_64)| (default, Dec  7 2015, 11:24:55)
Type "copyright", "credits" or "license" for more information.

IPython 4.0.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: print "Hellow world"
  File "<ipython-input-1-8feb44e30b9b>", line 1
    print "Hellow world"
                        ^
SyntaxError: Missing parentheses in call to 'print'
```

python™ 2

October 16, 2000

print is a statement

integer division

2. ipython (ssh)

```
bcf at ketchup in ~
$ ipython
Python 2.7.9 |Continuum Analytics, Inc.| (default, Mar  9 2015, 16:20
Type "copyright", "credits" or "license" for more information.

IPython 1.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details

In [1]: 3/2
Out[1]: 1

In [2]:
```

python™ 3

December 3, 2008

print is a function

float division

3. bash

```
(py35)
bcf at brant-4 in ~
$ ipython
Python 3.5.1 |Anaconda 2.4.1 (x86_64)| (default, Dec  7 2015, 11:24:55)
Type "copyright", "credits" or "license" for more information.

IPython 4.0.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: 3/2
Out[1]: 1.5

In [2]:
```

python™ 2

October 16, 2000

print is a statement

integer division

ASCII strings

xrange()

Support stops in 2020

python™ 3

December 3, 2008

print is a function

float division

Unicode strings

range()

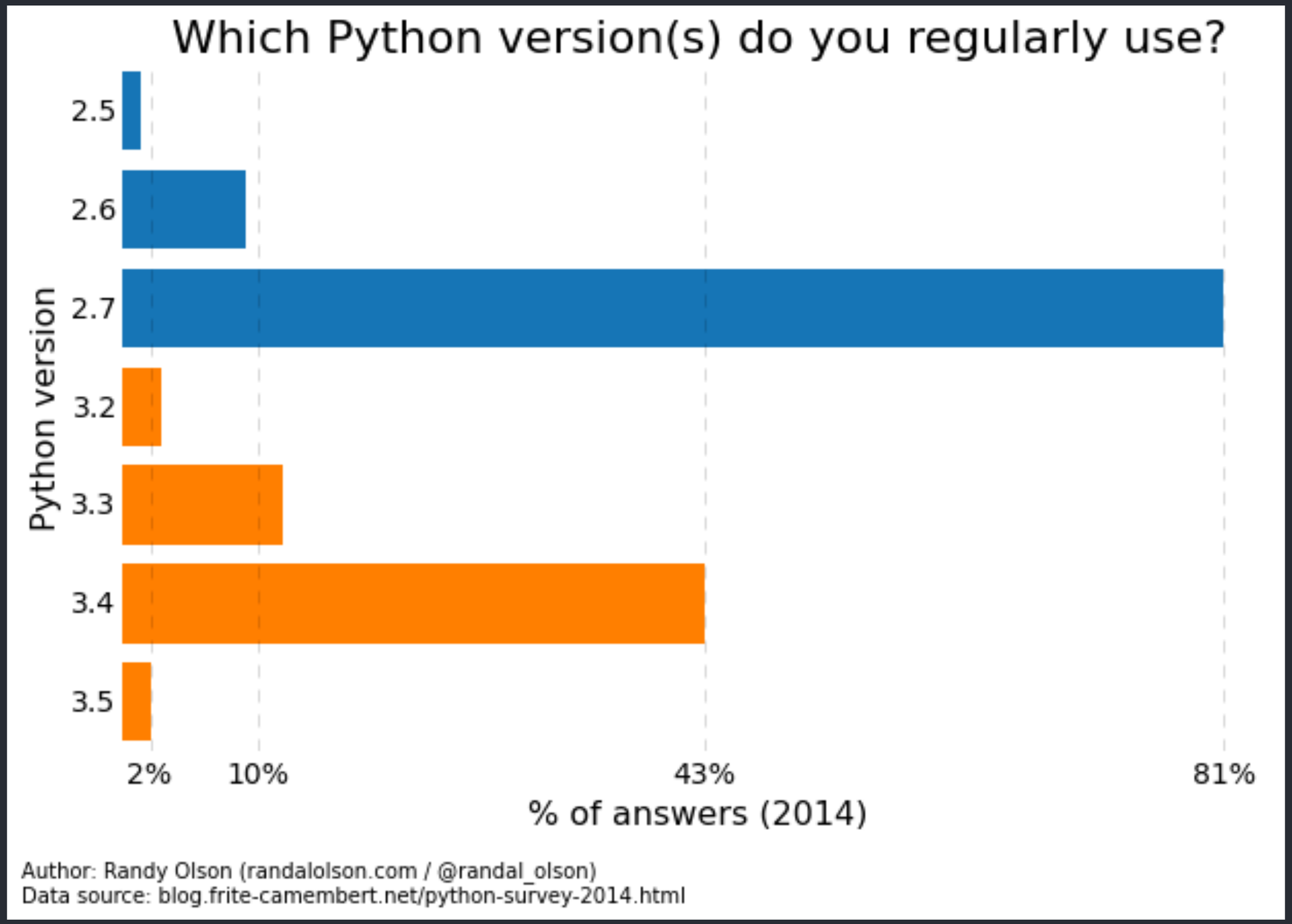
many “under-the-hood” changes…

python™ 2

October 16, 2000

python™ 3

December 3, 2008

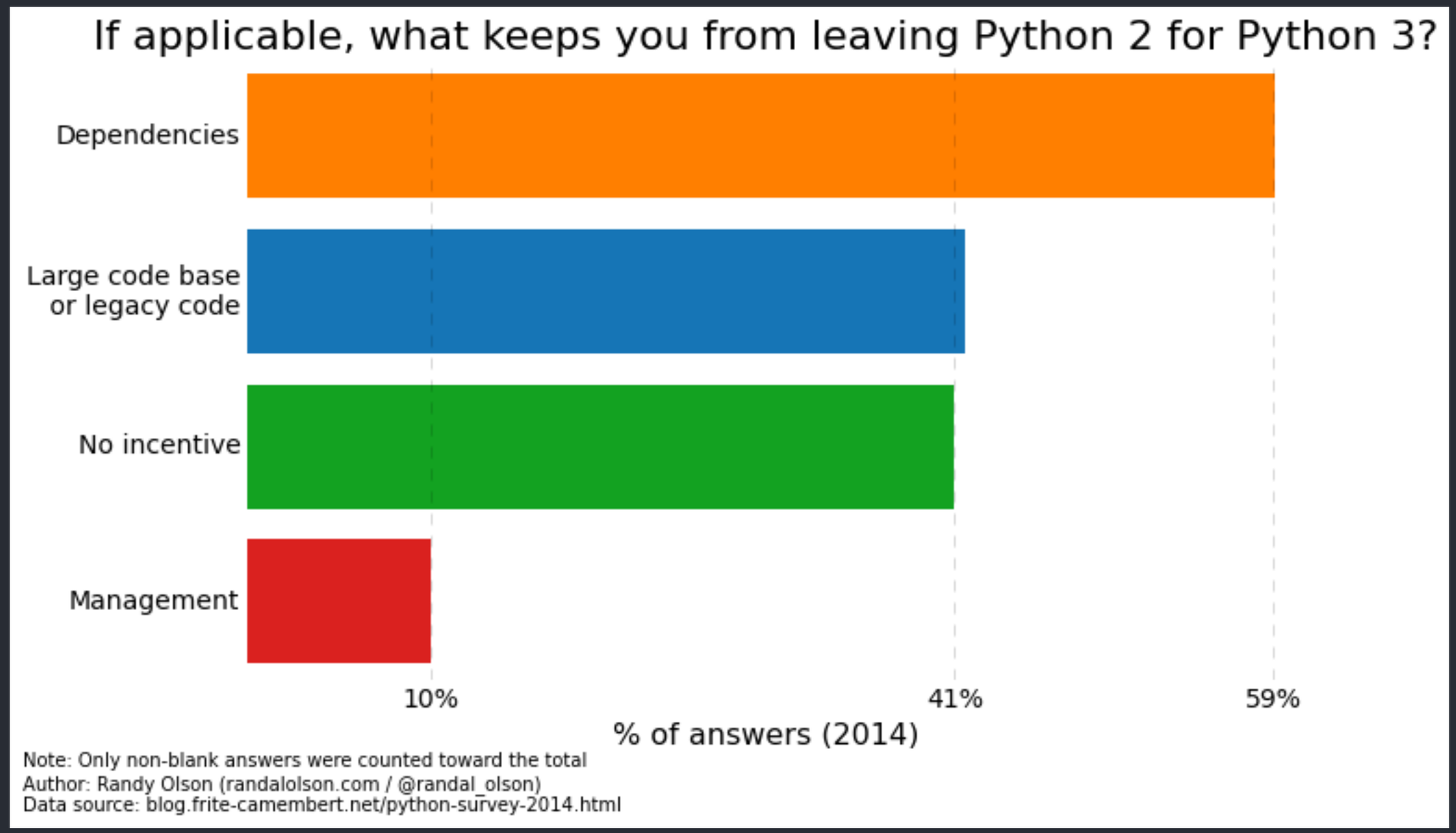


python™ 2

October 16, 2000

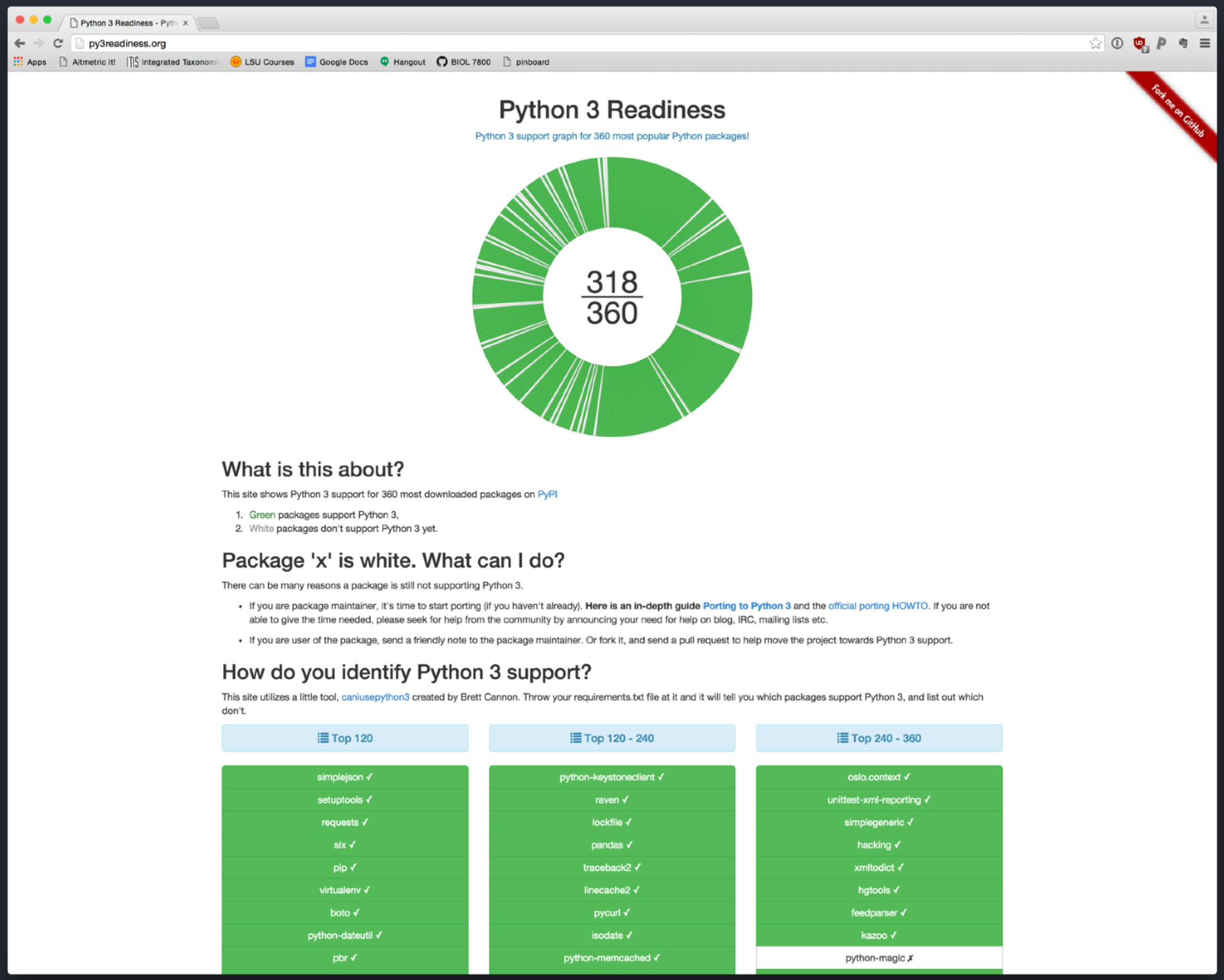
python™ 3

December 3, 2008



python™ 3 → python™ 3.6.x

December 3, 2008 → December 12, 2016

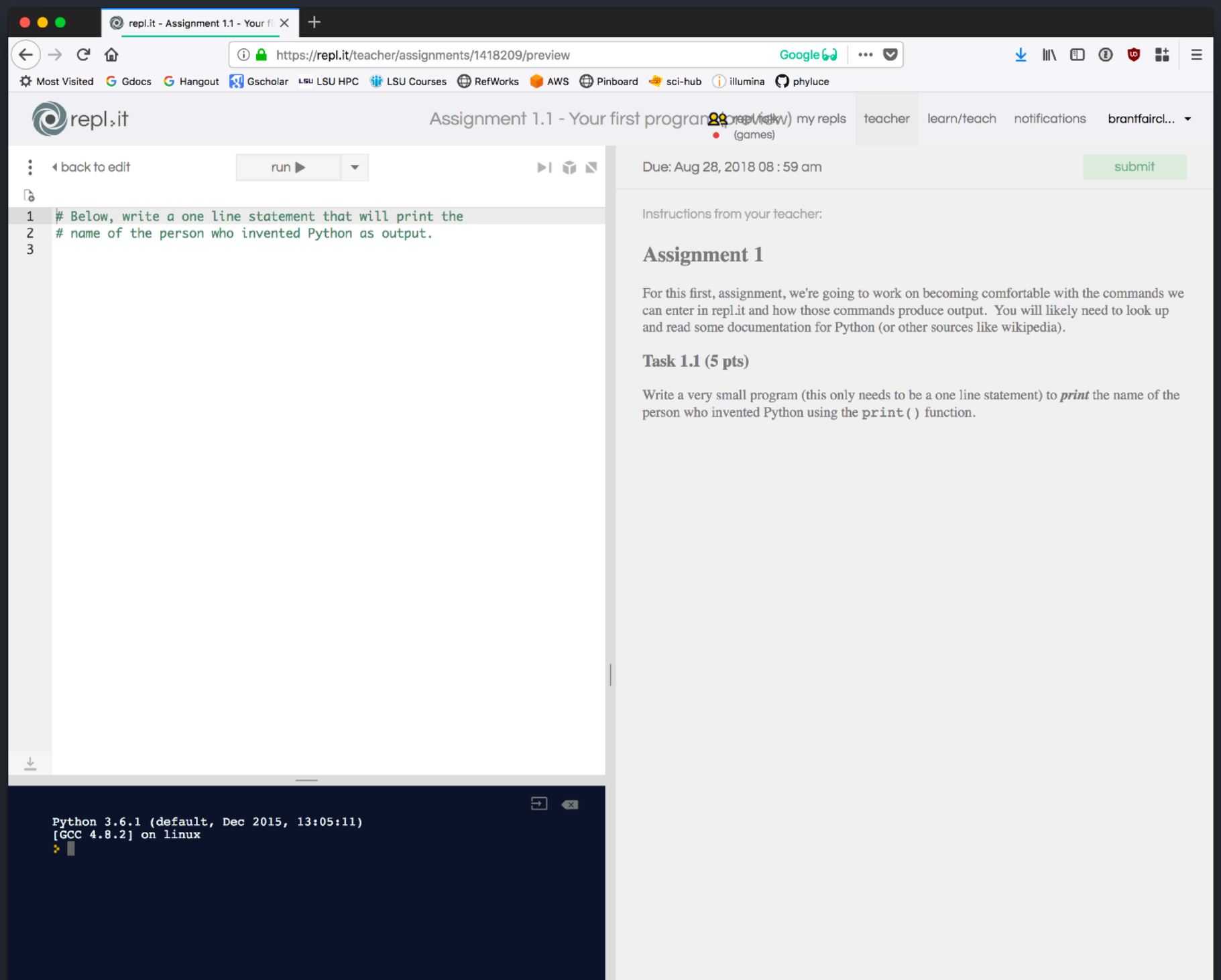


python™ 3.6.x

December 12, 2016

python™ 3.6.x

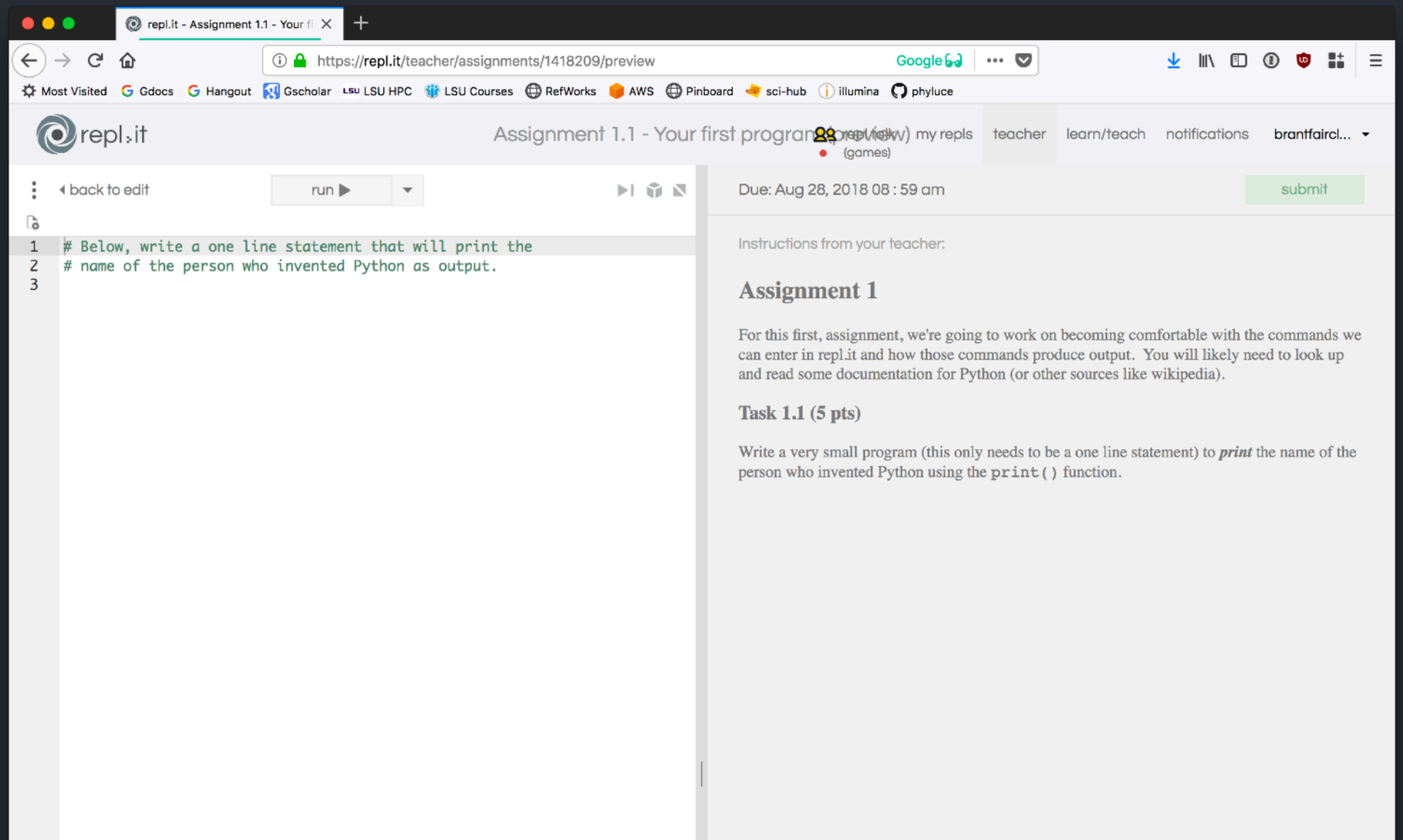
At least in beginning, we are using **repl.it**



python™ 3.5.x

But, what is **repl.it**?

It's basically a *web interface to* Python that helps us work around different issues on different platforms (meaning it runs the same on windows and mac os)



python™ 3.5.x

Python has **modules**,
and in Python, you can import things…

3. python

```
(py35)
bcf at brant-4 in ~
$ ipython
Python 3.5.1 |Anaconda 2.4.1 (x86_64)| (default, Dec  7 2015, 11:24:55)
Type "copyright", "credits" or "license" for more information.

IPython 4.0.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import random

In [2]:
```

python™ 3.5.x

But, **what the hell** are **modules?**

python™ 3.5.x

Python **modules**

But, **what the hell** are **modules?**

They are basically a way to make functions and modules you write persist (stick around) so you can use them again.

python™ 3.5.x

Python **modules**

Many modules are part of the **stdlib**

os
sys
string
sqlite3
random
collections

What's the **difference**?

While others are 3rd-party modules

scipy
PyMC
numpy
pandas
biopython
statsmodels

python™ 3.6.x

You can import things different ways…

repl.it

Assignment 1.1 - Your first program (review) my repls teacher learn/teach notifications brantfaircl...

(games)

back to edit run

```
# Below, write a one line statement that will print the
# name of the person who invented Python as output.

import numpy

from numpy import *

from numpy import random
```

Due: Aug 28, 2018 08 : 59 am submit

Instructions from your teacher:

Assignment 1

For this first, assignment, we're going to work on becoming comfortable with the commands we can enter in repl.it and how those commands produce output. You will likely need to look up and read some documentation for Python (or other sources like wikipedia).

Task 1.1 (5 pts)

Write a very small program (this only needs to be a one line statement) to ***print*** the name of the person who invented Python using the `print()` function.

(and we'll get to these)