C → Python

CS50 Seminar

Ross Rheingans-Yoo

November 5, 2015

This seminar will not teach you Python *from scratch*

— Python is *very* similar to C

— I’ll give you the major differences, and most of the important “Python magic”

This seminar will teach you to write Python programs

— We will implement psets 0-6 in Python

— ...and provide pointers to official documentation for your future reference

This video will be online, courtesy of CS50

python 3 is not python 2

use <$ python --version> to check
try python3 if python defaults to 2.X.Y

official resources — docs.python.org/3/

Tutorial — docs.python.org/3/tutorial/

Language Reference — docs.python.org/3/reference/

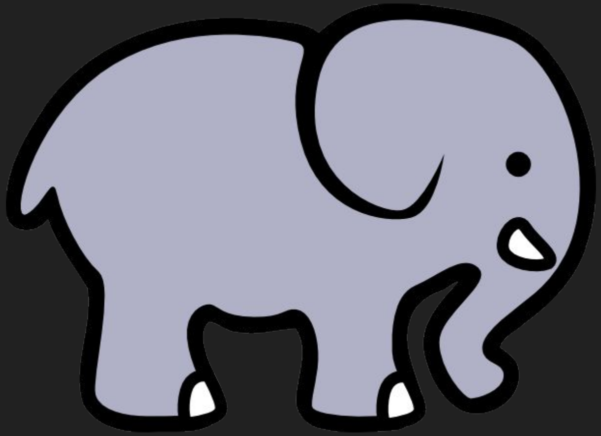
Standard Library Reference — docs.python.org/3/library/

```c
#include <stdio.h>

int main(void)
{
    int n;
    n = 10;

    for (int row=0; row<n; row++)
    {
        for (int col=0; col<n; col++)
        {
            if (row+col < n-1)
            {
                printf(" ");
            }
            else
            {
                printf("#");
            }
        }
        printf("#\n");
    }
}
```

```python
n = 10

for row in range(n):
    for col in range(n):
        if row+col < n-1:
            print(' ', end="")
        else:
            print("#", end='')
    print('#')
```

```
$ make mario
clang -ggdb3 -O0 -std=c11 -Wall -Werror mario.c -lcs50 -lm -o mario
$ ./mario




$ python mario.py
```

Rules for Python



— Try natural syntax before looking anything up.

— Standard library functions / methods are your friends!

— Most standard functions support most logically sensible inputs.

— Somebody else has probably already written a library for that.

```c
#include <stdio.h>

int main(void) /**/
{ /**/
    int n;
    n = 10;

    for (int row=0; row<n; row++)
    {
        for (int col=0; col<n; col++)
        {
            if (row+col < n-1)
            {
                printf(" ");
            }
            else
            {
                printf("#");
            }
        }
        printf("#\n");
    }
} /**/
```

```python
def mario(): #
    n = 10

    for row in range(n):
        for col in range(n):
            if row+col < n-1:
                print(' ', end="")
            else:
                print("#", end='')
        print('#')

if __name__ == '__main__': #
    mario() #
```

+ - * /

=

C:	&&	||	!
py:	and	or	not

```
 C: if (foo==bar) {
... }else if(){ ... }else{
py: if foo==bar:
...  elif():      ... else:
 C:  for (int i=0; i<n; i++) {

py:  for i in range(n):
```

```
C:  for (int i=0; i<n; i++) {
        printf(“%i\n”,ary[i]);

py: for a in ary:
        print(a)
```

```
# python magic

print(‘#’*5)
> #####
```

```c
#include <stdio.h>

int main(void)
{
    int n;
    n = 10;

    for (int row=0; row<n; row++)
    {
        for (int col=0; col<n; col++)
        {
            if (row+col < n-1)
            {
                printf(" ");
            }
            else
            {
                printf("#");
            }
        }
        printf("#\n");
    }
}
```

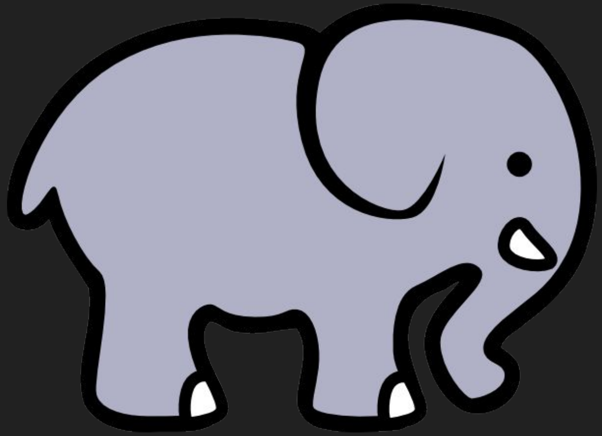
```python
def mario(n):
    for row in range(n):
        print((' '*(n-row-1))+
              ('#'*row) + '##')

if __name__ == '__main__':
    mario(10)
```

```c
#include <stdio.h>

int main(void)
{
    int n;
    n = 10;

    for (int row=0; row<n; row++)
    {
        for (int col=0; col<n; col++)
        {
            if (row+col < n-1)
            {
                printf(“ “);
            }
            else
            {
                printf(“#”);
            }
        }
        printf(“#\n”);
    }
}
```

```python
def mario(n):
    for row in range(n):
        print((‘#’*(row+2)).rjust(n+1))


if __name__ == ‘__main__’:
    mario(10)
```

```
object.method(argument, arg2)

function(object, argument, arg2)
```

Rules for Python



— Try natural syntax before looking anything up.

— Standard library functions / methods are your friends!

— Most standard functions support most logically sensible inputs.

— Somebody else has probably already written a library for that.

py: 64

```python
n = 10

for row in range(n):
    print((‘#’*(row+2)).rjust(n+1))
```

```c
int main(){int h=10,r=0,x;for(;r-h;++r)for(x=h+2;x;)printf(--x?r-x>-3?"#":" ":"\n");
```

C: 84

lists

```
squares = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]


squares = []
for i in range(1,10+1):
    squares.append(i**2) # ** is the exponentiation operator


squares = [ i**2 for i in range(1,10+1) ]
```

lists / printing

```
for i in range(len(squares)):
    print(squares[i], end=‘ ’)
```

```
> 1 4 9 16 25 36 49 64 81 100
```

```
for square in squares:
    print(square, end=‘ ’)
```

```
> 1 4 9 16 25 36 49 64 81 100
```

```
print(squares)
```

```
> [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

fifteen.py

```
def init(d):
    board = [[d**2-r-c-1
        for c in range(d)]
        for r in range(d)]

    if d>2 and d%2==0:
        board[d-1][d-3] = 1
        board[d-1][d-2] = 2

    return board
```

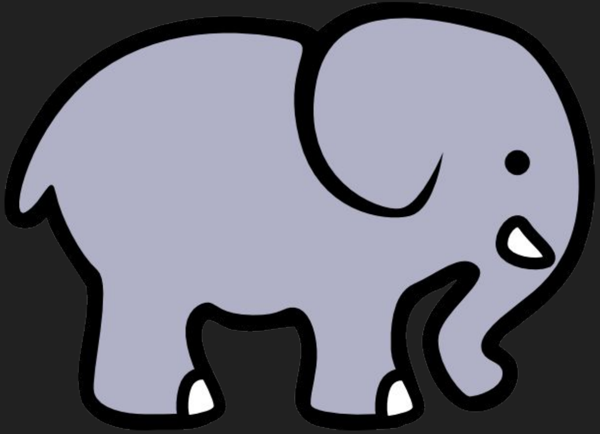
```
def draw(board):
    for row in board:
        for tile in row:
            if tile==0:
                print('   ', end='')
            else:
                print(' '+str(tile), end='')
        print('')

if __name__ == '__main__':
    board = init(4)
    draw(board)
    # etc...
```

find.py / sort.py

```
needle = 42
haystack = [41+i for i in range(5)]


if needle in haystack:
    return True
return False
```

```
ary = [1, 7, 5, 4, 0, 6, 3, 2]
print(sorted(ary))
> [0, 1, 2, 3, 4, 5, 6, 7]

print(sorted(ary, reverse=True))
> [7, 6, 5, 4, 3, 2, 1, 0]

ary = [‘cs50!’, ‘cats,’, ‘ban’]
print(sorted(ary))
> [‘ban’, ‘cats,’, ‘cs50!’]
```

Rules for Python



— Try natural syntax before looking anything up.

— Standard library functions / methods are your friends!

— Most standard functions support most logically sensible inputs.

— Somebody else has probably already written a library for that.

file I/O — https://docs.python.org/3/tutorial/inputoutput.html

```
f = open(‘filename.txt’, ‘r’)
# r(ead); r+(=read&write);
# w(rite); a(ppend/write)

print(f.read(6))

> Hello!

print(f.read())

> This is the rest of the file.
>

print(f.read())

>

f.close()
```

```
f2 = open(‘otherfile.txt’, ‘r’)
f2.readline()

> This is the first line.

f2.close()

f3 = open(‘newfile.txt’, ‘w’)
f3.write(‘youtu.be/dQw4w9WgXcQ’)
# f3.write(b‘ffd8ffe0’) for raw bytes
f3.close()

f4 = open(‘thirdfile.txt’, ‘r’)
for line in f4:
    print(line, end=‘’)

> This is the answer key to Quiz 1.
> Please don’t share it with students!
>
> The answers to the first page are
```

dictionaries — hashtables in disguise!

```
dict = {}

dict[‘Rob’] = ‘Bowden’
dict[‘Rick’] = ‘Astley’
dict[‘David’] = [‘Malan’, ‘Hughes’]

print(dict[‘Rob’])

> Bowden

print(dict[‘Ross’])

> KeyError: ‘Ross’
$

for key, value in dict:
    print(key+‘ ’+str(value))

> Rob Bowden
> David [‘Malan’, ‘Hughes’]
> Rick Astley
```

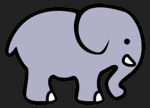
```
def load(dictfile):
    dict = {}
    df = open(dictfile, ‘r’)

    for line in df:
        word = line[:-1]
        dict[word] = 1 # dummy value

    return dict



def check(word, dict):
    if word in dict:
        return True
    return False
```

modules / command-line arguments

echo-args.py

```
import sys
# docs.python.org/3/library/sys.html

argc = len(sys.argv)
print(sys.argv)
```

```
$ python echo-args.py
['echo-args.py']
$ python echo-args.py cs50
['echo-args.py', 'cs50']
```



```
# use argparse for options / flags!
```

modules / json, pickle

```
import json

f = open('thing.json', 'r')
thing = json.load(f)
f.close()

print(thing)
> ["...json contents here..."]

f = open('new.json', 'w')
json.dump(thing, f)
f.close()

# json.loads() and json.dumps()
# [load from / dump to] strings,
# rather than files
```

```
import pickle

dict = {'Rob':'Bowden', 'David':'Malan'}

f = open('thing.p', 'w')
pickle.dump(thing, f)
f.close()

f = open('thing.p', 'r')
thing = pickle.load(f)
f.close()

print(thing)
> {'David': 'Malan', 'Rob': 'Bowden'}
```

modules / *

```
sys             # system utility functions
os              # misc. operating system interfaces
multiprocessing # multi-threading utilities

json   # csv reader/writer
pickle # pickle (de)serializer
csv    # csv reader/writer

urllib # http client
bs4    # (BeautifulSoup) html parser

numpy   # numeric matrix operations
scipy   # science / engineering utilities
sklearn # machine learning
nltk    # (Natural Language Toolkit) natural language processing
```

modules /

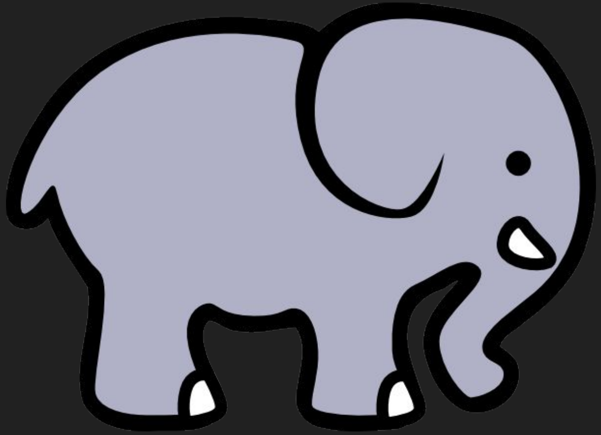
```
import sys

import multiprocessing as mp

from urllib import request
# invoke as just request(), not urllib.request

from urllib import *
# beware namespace collisions!
```

python design cycle

— Check if someone has already written it

— Think about how you would design it...

Rules for Python



— Try natural syntax before looking it up.

— Standard library functions / methods are your friends!

— Most standard functions support most logically sensible inputs.

— Somebody else has probably already written a library for that.

black boxes / white lies

- A lot of “magic” is “object-oriented programming”
  - You can make your own “magic” objects… though it takes some work
- `for thing in list:` is subtly different from C-style `for()` loops
- There is a dark side to “magic”
- Garbage collection
- Python treats tabs and spaces differently, so watch out!
  - Either is acceptable, but they are not interchangeable

the biggest hole in this talk

- tuples

pset6

```
$ python -m SimpleHTTPServer 8080
```