Imperative programming with Python

January 2012 project: Class #4

Facundo Carreiro

ILLC, University of Amsterdam

January 11th, 2012

Strings: the real thing

- We saw that strings are sequences of letters

```
s = 'this is a string'
```

Strings: the real thing

- We saw that strings are sequences of letters

```
s = 'this is a string'
```

- They can be *indexed* by integers

```
>>> s[2]
'i'
```

. . . starting from 0, and up to length-1

```
>>> len(s)
16
>>> s[16]
IndexError: string index out of range
```

Strings: the real thing

- We saw that strings are sequences of letters

```
s = 'this is a string'
```

- They can be *indexed* by integers

```
>>> s[2]
'i'
```

...starting from 0, and up to length-1

```
>>> len(s)
16
>>> s[16]
IndexError: string index out of range
```

- Although...

```
>>> s[-1]
'g'
```

you can count *backwards* using negative numbers!
Warning: this is highly *Python*-specific.

Strings: the real thing

- Strings are immutable: you can't modify them

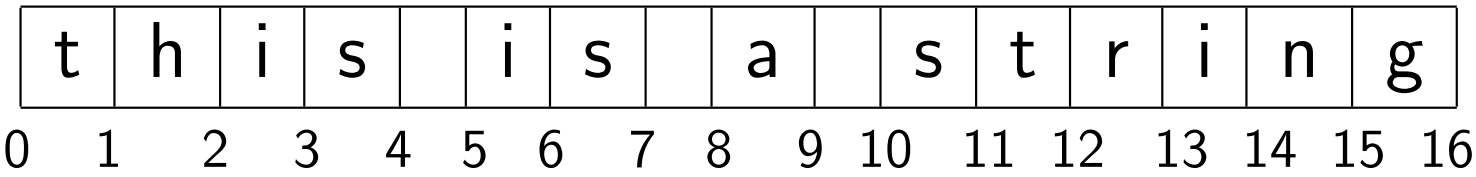
```
>>> s[4] = 'L'
TypeError: 'str' object does not support item assignment
```

Strings: the real thing

- Strings are immutable: you can't modify them

```
>>> s[4] = 'L'
TypeError: 'str' object does not support item assignment
```

- But you can make new strings out of its *slices*




```
>>> s[0:7] + ' not' + s[7:]
'this is not a string'
```

Strings methods

- *Methods* are functions associated with an object.
- They are called using the 'dot notation'.

Strings methods

- *Methods* are functions associated with an object.
- They are called using the 'dot notation'.
- For example, strings have a method called `upper`

```
>>> s.upper()
'THIS␣IS␣A␣STRING'
>>> s
'this␣is␣a␣string'
```

it returns an uppercased version of the string *without* modifying it.

Strings methods

- *Methods* are functions associated with an object.
- They are called using the 'dot notation'.
- For example, strings have a method called `upper`

```
>>> s.upper()
'THIS␣IS␣A␣STRING'
>>> s
'this␣is␣a␣string'
```

  it returns an uppercased version of the string *without* modifying it.

- The `find` method looks for a substring and returns its index

```
>>> 'Python'.find('thon')
2
>>> 'Python'.find('tuna')
-1
```

Strings methods

- *Methods* are functions associated with an object.
- They are called using the 'dot notation'.
- For example, strings have a method called `upper`

```
>>> s.upper()
'THIS␣IS␣A␣STRING'
>>> s
'this␣is␣a␣string'
```

  it returns an uppercased version of the string *without* modifying it.

- The `find` method looks for a substring and returns its index

```
>>> 'Python'.find('thon')
2
>>> 'Python'.find('tuna')
-1
```

- Suggested HW: Check all of them in the Python documentation.

Basic File I/O

Reading

- *File objects* are values representing files.
- We use the `open` function to read/write from files

```
>>> f = open('file.txt')
>>> print f
<open file 'file.txt', mode 'r' at 0x630b0>
```

mode `'r'` means that the file is open for *reading*.

Basic File I/O

Reading

- *File objects* are values representing files.
- We use the `open` function to read/write from files

```
>>> f = open('file.txt')
>>> print f
<open file 'file.txt', mode 'r' at 0x630b0>
```

mode `'r'` means that the file is open for *reading*.

- We can read the file line by line using the `readline` method.

```
>>> f.readline()
'First line\n'
```

- The object remembers our position in the file, if you call `readline` again you get the following result

```
>>> f.readline()
'Second line\n'
```

Basic File I/O

Writing

- We use `open` with the `'w'` mode to write to files

```
>>> f = open('out.txt', 'w')
```

Basic File I/O

Writing

- We use `open` with the `'w'` mode to write to files

```
>>> f = open('out.txt', 'w')
```

- The `write` method... well, writes to the file

```
>>> f.write('first thing')
>>> f.write('second thing')
```

Basic File I/O

Writing

- We use `open` with the `'w'` mode to write to files

```
>>> f = open('out.txt', 'w')
```

- The `write` method... well, writes to the file

```
>>> f.write('first thing')
>>> f.write('second thing')
```

- Always close a file when you are done with it. *Both* if you were reading or writing.

```
>>> f.close()
>>> print f
<closed file 'out.txt', mode 'w' at 0x632f0>
```

Basic File I/O

Writing

- We use `open` with the `'w'` mode to write to files

```
>>> f = open('out.txt', 'w')
```

- The `write` method. . . well, writes to the file

```
>>> f.write('first thing')
>>> f.write('second thing')
```

- Always close a file when you are done with it. *Both* if you were reading or writing.

```
>>> f.close()
>>> print f
<closed file 'out.txt', mode 'w' at 0x632f0>
```

- *Note*: if we check out newly created file we see

```
first thingsecond thing
```

Newlines should be added explicitly! (with `\n`)

System arguments

- When executing a program you can also pass arguments to it

```
python program.py argument1 argument2
```

System arguments

- When executing a program you can also pass arguments to it

```
python program.py argument1 argument2
```

- You can use the `sys` module to access them

```
import sys
print sys.argv
```

save that as 'program.py' and execute

```
python program.py argument1 argument2
```

System arguments

- You will get the complete list of arguments

```
['program.py', 'argument1', 'argument2']
```

- The first one is always the name of the program
- The rest, if they exist, are the other arguments

System arguments

- You will get the complete list of arguments

```
['program.py', 'argument1', 'argument2']
```

- The first one is always the name of the program
- The rest, if they exist, are the other arguments
- You can get the number of arguments using

```
len(sys.argv)
```

and access each one using

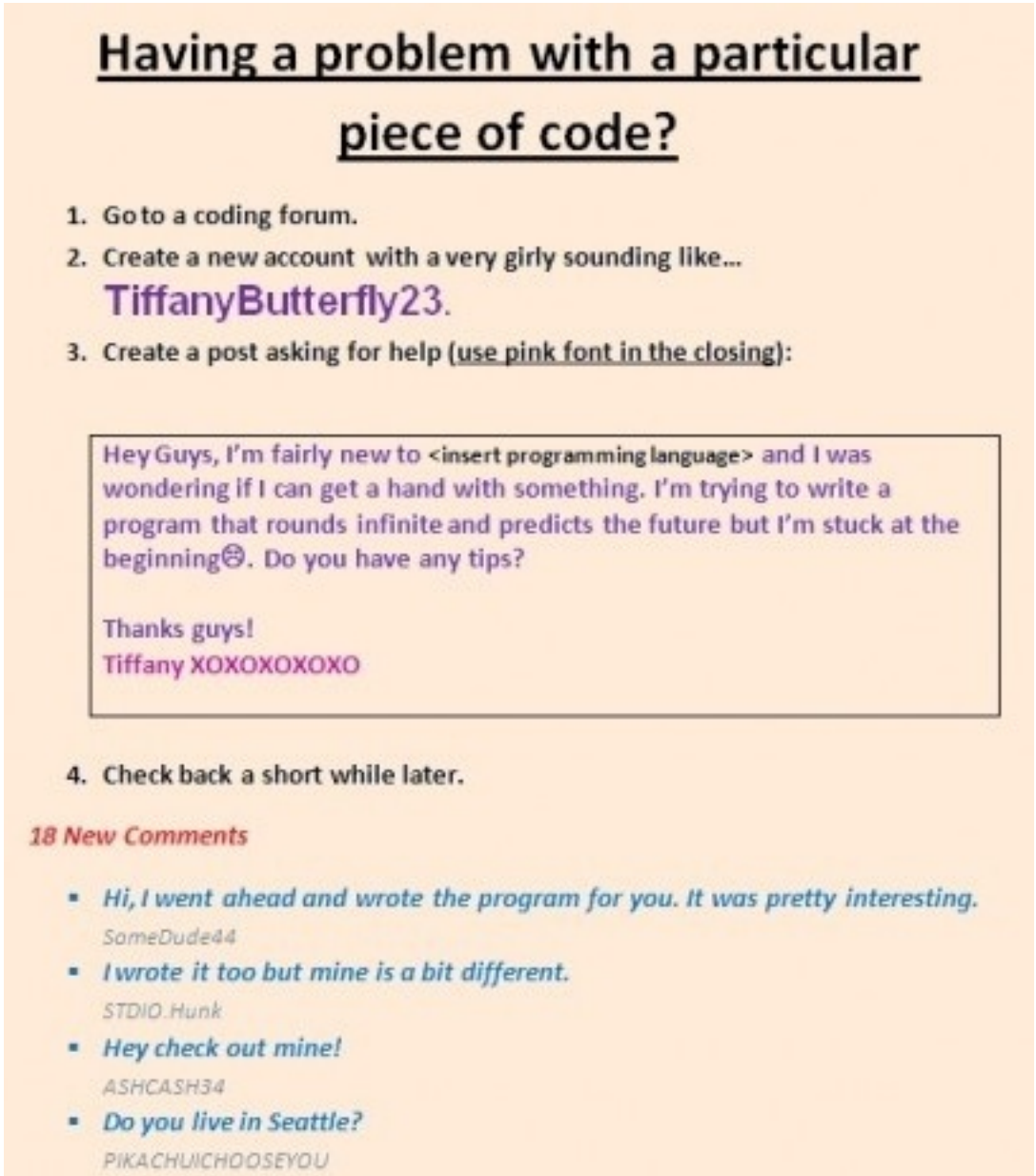
```
sys.argv[0]
sys.argv[1]
...
```

Where to find help?

- Google
- Python documentation
- Ask Fabio

Where to find help?

- Google
- Python documentation
- Ask Fabio
- You can try this →



Having a problem with a particular piece of code?

1. Go to a coding forum.
2. Create a new account with a very girly sounding like...
   TiffanyButterfly23.
3. Create a post asking for help (use pink font in the closing):

> Hey Guys, I'm fairly new to <insert programming language> and I was wondering if I can get a hand with something. I'm trying to write a program that rounds infinite and predicts the future but I'm stuck at the beginning☹. Do you have any tips?
>
> Thanks guys!
> Tiffany XOXOXOXOXO

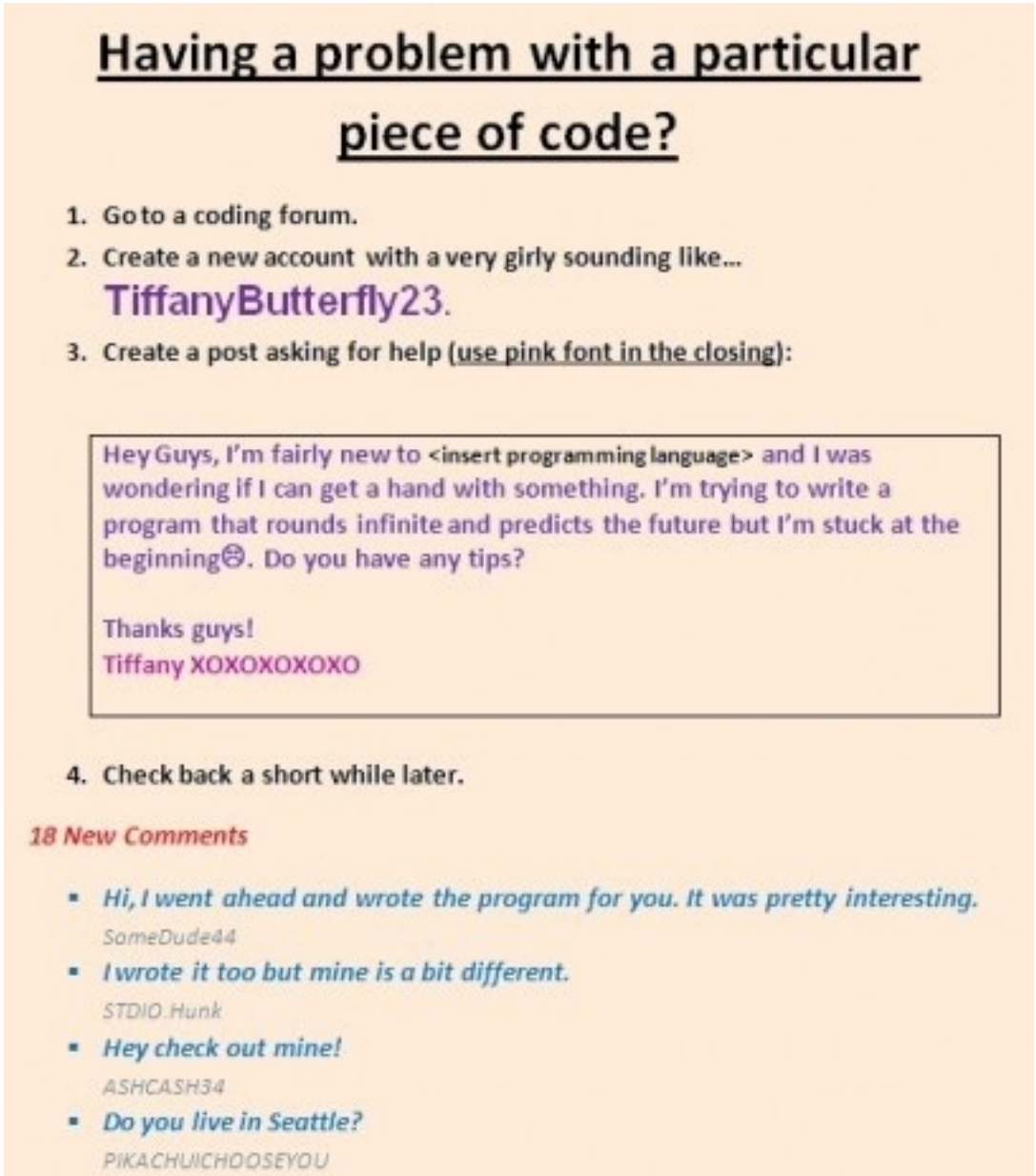
4. Check back a short while later.

18 New Comments

- *Hi, I went ahead and wrote the program for you. It was pretty interesting.*
  SomeDude44
- *I wrote it too but mine is a bit different.*
  STDIO.Hunk
- *Hey check out mine!*
  ASHCASH34
- *Do you live in Seattle?*
  PIKACHUICHOOSEYOU

Where to find help?

- Google
- Python documentation
- Ask Fabio
- You can try this $\rightarrow$
- Ask Me



Having a problem with a particular piece of code?

1. Go to a coding forum.
2. Create a new account with a very girly sounding like...
   TiffanyButterfly23.
3. Create a post asking for help (use pink font in the closing):

> Hey Guys, I'm fairly new to <insert programming language> and I was wondering if I can get a hand with something. I'm trying to write a program that rounds infinite and predicts the future but I'm stuck at the beginning☹. Do you have any tips?
>
> Thanks guys!
> Tiffany XOXOXOXOXO

4. Check back a short while later.

18 New Comments

- *Hi, I went ahead and wrote the program for you. It was pretty interesting.*
  SomeDude44
- *I wrote it too but mine is a bit different.*
  STDIO.Hunk
- *Hey check out mine!*
  ASHCASH34
- *Do you live in Seattle?*
  PIKACHUICHOOSEYOU

References

- Chapters 8 and 9 of the book
  http://greenteapress.com/thinkpython/thinkpython.html
- String methods
  http://docs.python.org/library/stdtypes.html#string-methods
- File objects
  http://docs.python.org/library/stdtypes.html#file-objects