

Lecture 3

**Strings,
Functions, & Modules**

Labs this Week

- Lab 1 is due at the **beginning** of your lab
 - If it is not yet by then, you cannot get credit
 - Only exception is for students who added late (Those students should talk to me)
- Should spend time *entirely* on Lab 2
 - Similar format to last week
 - Next weeks lab is substantially longer

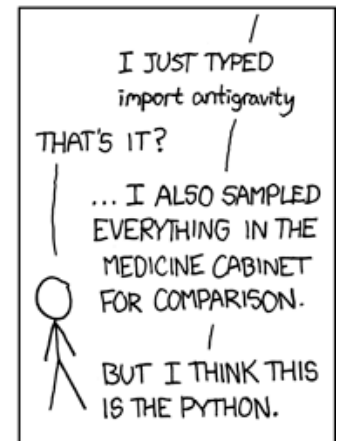
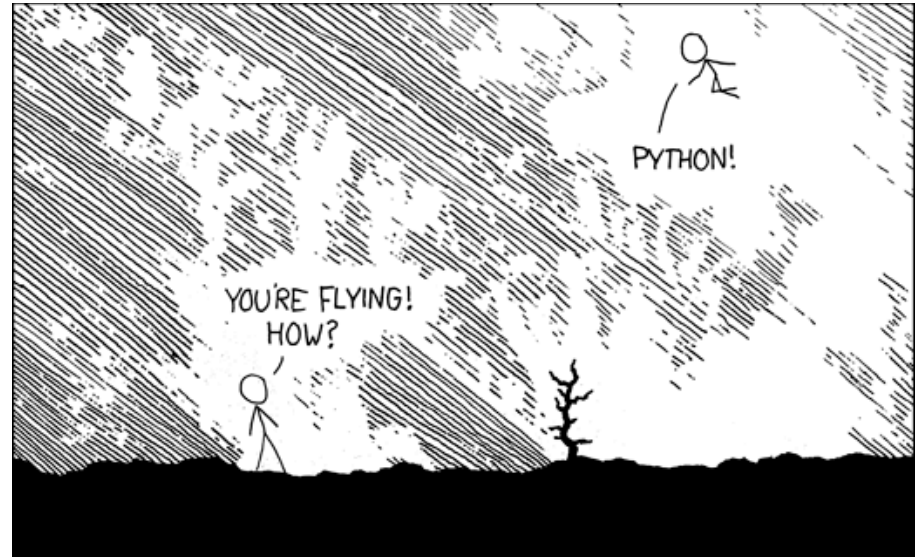
Readings for Next Two Lectures

This Lecture

- Sections 3.1-3.4
- Sections 8.1, 8.2, 8.4, 8.5
- Browse the Python API
 - Do not need to read all of it
 - Look over built-in functions

Thursday

- Complete Chapter 3



[xkcd.com]

String: Text as a Value

- String are quoted characters
 - 'abc d' (Python prefers)
 - "abc d" (most languages)
- How to write quotes in quotes?
 - Delineate with “other quote”
 - **Example:** " ' " or ' " '
 - What if need both " and ' ?
- **Solution:** escape characters
 - Format: \ + letter
 - Special or invisible chars

Type: str

Char	Meaning
\'	single quote
\"	double quote
\n	new line
\t	tab
\\	backslash

String are Indexed

- `s = 'abc d'`

0	1	2	3	4
a	b	c		d

- `s = 'Hello all'`

0	1	2	3	4	5	6	7	8
H	e	l	l	o		a	l	l

- Access characters with `[]`

- `s[0]` is 'a'
- `s[4]` is 'd'
- `s[5]` **causes an error**
- `s[0:2]` is 'ab' (excludes c)
- `s[2:]` is 'c d'

- What is `s[3:6]`?

A: 'lo a'
B: 'lo'
C: 'lo '
D: 'o '
E: I do not know

- Called “string slicing”

String are Indexed

- `s = 'abc d'`

0	1	2	3	4
a	b	c		d

- `s = 'Hello all'`

0	1	2	3	4	5	6	7	8
H	e	l	l	o		a	l	l

- Access characters with `[]`

- `s[0]` is 'a'
- `s[4]` is 'd'
- `s[5]` **causes an error**
- `s[0:2]` is 'ab' (excludes c)
- `s[2:]` is 'c d'

- What is `s[3:6]`?

A: 'lo a'
B: 'lo'
C: 'lo ' **CORRECT**
D: 'o '
E: I do not know

- Called “string slicing”

String are Indexed

- `s = 'abc d'`

0	1	2	3	4
a	b	c		d

- `s = 'Hello all'`

0	1	2	3	4	5	6	7	8
H	e	l	l	o		a	l	l

- Access characters with `[]`

- `s[0]` is 'a'
- `s[4]` is 'd'
- `s[5]` **causes an error**
- `s[0:2]` is 'ab' (excludes c)
- `s[2:]` is 'c d'

- What is `s[:4]`?

A: 'o all'
B: 'Hello'
C: 'Hell'
D: **Error!**
E: I do not know

- Called “string slicing”

String are Indexed

- `s = 'abc d'`

0	1	2	3	4
a	b	c		d

- `s = 'Hello all'`

0	1	2	3	4	5	6	7	8
H	e	l	l	o		a	l	l

- Access characters with `[]`

- `s[0]` is 'a'
- `s[4]` is 'd'
- `s[5]` **causes an error**
- `s[0:2]` is 'ab' (excludes c)
- `s[2:]` is 'c d'

- What is `s[:4]`?

A: 'o all'
B: 'Hello'
C: 'Hell' **CORRECT**
D: **Error!**
E: I do not know

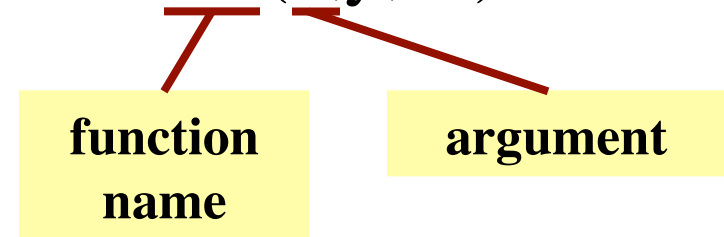
- Called “string slicing”

Other Things We Can Do With Strings

- **Operation** `in`: `s1 in s2`
 - Tests if `s1` “a part of” `s2`
 - Say `s1` a *substring* of `s2`
 - Evaluates to a `bool`
- **Function** `len`: `len(s)`
 - Value is # of chars in `s`
 - Evaluates to an `int`
- **Examples:**
 - `s = 'abracadabra'`
 - `'a' in s == True`
 - `'cad' in s == True`
 - `'foo' in s == False`
- **Examples:**
 - `s = 'abracadabra'`
 - `len(s) == 11`
 - `len(s[1:5]) == 4`
 - `s[1:len(s)-1] == 'bracadabr'`

Function Calls

- Python supports expressions with math-like functions
 - A function in an expression is a **function call**
 - Will explain the meaning of this later
- Function expressions have the form **fun**(x,y,...)



- **Examples** (math functions that work in Python):
 - `round(2.34)`
 - `max(a+3,24)`

Arguments can be any **expression**

Built-In Functions

- You have seen many functions already
 - Type casting functions: `int()`, `float()`, `bool()`
 - Dynamically type an expression: `type()`
 - Help function: `help()`
- Getting user input: `raw_input()`
- `print <string>` is **not** a function call
 - It is simply a statement (like assignment)
 - But it is in Python 3.x: `print(<string>)`

Arguments go in (),
but `name()` refers to
function in general

Method: A Special Type of Function

- Methods are unique (right now) to strings
- Like a function call with a “string in front”
 - Usage: *string.method*(x,y...)
 - The string is an *implicit argument*
- Example: upper()
 - `s = 'Hello World'`
 - `s.upper() == 'HELLO WORLD'`
 - `s[1:5].upper() == 'ELLO'`
 - `'abc'.upper() == 'ABC'`

Will see why we do it this way later in course

Examples of String Methods

- `s1.index(s2)`
 - Position of the first instance of `s2` in `s1`
 - `s1.count(s2)`
 - Number of times `s2` appears inside of `s1`
 - `s.strip()`
 - A copy of `s` with white-space removed at ends
- `s = 'abracadabra'`
 - `s.index('a') == 0`
 - `s.index('rac') == 2`
 - `s.count('a') == 5`
 - `' a b '.strip() == 'a b'`

See Python
Docs for more

Built-in Functions vs Modules

- The number of built-in functions is small
 - <http://docs.python.org/2/library/functions.html>
- Missing a lot of functions you would expect
 - **Example:** `cos()`, `sqrt()`
- **Module:** file that contains Python code
 - A way for Python to provide optional functions
 - To access a module, the `import` command
 - Access the functions using module as a *prefix*

Example: Module math

```
>>> import math
```

To access math functions

```
>>> math.cos(0)
```

```
1.0
```

Functions require math prefix!

```
>>> cos(0)
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
NameError: name 'cos' is not defined
```

```
>>> math.pi
```

Module has variables too!

```
3.141592653589793
```

```
>>> math.cos(math.pi)
```

```
-1.0
```

Example: Module math

```
>>> import math
```

To access math functions

```
>>> math.cos(0)
```

```
1.0
```

```
>>> cos(0)
```

Functions require math prefix!

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
NameError: name 'cos' is not defined
```

```
>>> math.pi
```

Module has variables too!

```
3.141592653589793
```

```
>>> math.cos(math.pi)
```

```
-1.0
```

Other Modules

- `io`
 - Read/write from files
- `random`
 - Generate random numbers
 - Can pick any distribution
- `string`
 - Useful string functions
- `sys`
 - Information about your OS

Reading the Python Documentation

The screenshot shows the Python documentation page for the `math` module. The page title is "9.2. math — Mathematical functions". The left sidebar contains a "Table Of Contents" with links to various sub-sections. The main content area includes a description of the module, a list of functions provided, and a search bar at the bottom. A search bar is located at the bottom left of the page with the text "Enter search terms or a module, class or function name." and a "Go" button.

Function name (points to `math.ceil(x)`)

Possible arguments (points to `x` in `math.ceil(x)`)

Module (points to `math` in `math.ceil(x)`)

What the function evaluates to (points to the description of `math.ceil(x)`)

Return the ceiling of x as a float, the smallest integer value greater than or equal to x .

Return x with the sign of y . On a platform that supports signed zeros, `copysign(1.0, -0.0)` returns `-1.0`.

Return the absolute value of x .

Return x factorial. Raises `ValueError` if x is not integral or is negative.

New in version 2.6.

Return the floor of x as a float, the largest integer value less than or equal to x .

<http://docs.python.org/library>

Using the from Keyword

```
>>> import math
```

```
>>> math.pi
```

Must prefix with
module name

```
3.141592653589793
```

```
>>> from math import pi
```

```
>>> pi
```

No prefix needed
for variable pi

```
3.141592653589793
```

```
>>> from math import *
```

```
>>> cos(pi)
```

```
-1.0
```

No prefix needed
for anything in math

- Be careful using from!
- Using import is *safer*
 - Modules might conflict (functions w/ same name)
 - What if import both?
- **Example:** Turtles
 - Older version of A4
 - 2 modules: turtle, tkturtle
 - Both have func. Turtle()

A String Puzzle (Extraction Practice)

- **Given:** a string with a parenthesis pair inside
`s = 'labs are (usually) every week'`
- **Goal:** expression for substring inside parentheses
 - **Step 1:** Find the open parenthesis
`start = s.index('(')`
 - **Step 2:** Store part of string **after** parenthesis in **tail**
`tail = s[start+1:]`
 - **Step 3:** Get the part of the tail **before** close parenthesis
`tail[:tail.index(')')]`

- **Given:** A string that is a list of words separated by commas, and spaces in between each comma:

```
pets = 'cat, dog, mouse, lion'
```



- **Goal:** Want second element with no spaces or commas.
Put result inside of variable `answer`

Where, in the following sequence of commands, is there a (conceptual) error that prevents our goal?

A: `startcomma = info.index(',')`

B: `tail = info[startcomma+1:]`

C: `endcomma = tail.index(',')`

D: `df = tail[:endcomma]`

E: this sequence achieves the goal

- **Given:** A string that is a list of words separated by commas, and spaces in between each comma:

```
pets = 'cat, dog, mouse, lion'
```



- **Goal:** Want second element with no spaces or commas.
Put result inside of variable `answer`

Where, in the following sequence of commands, is there a (conceptual) error that prevents our goal?

A: `startcomma = info.index(',')`

B: `tail = info[startcomma+1:]` +2 instead, or use

C: `endcomma = tail.index(',')`

D: `df = tail[:endcomma]` `tail[:endcomma].strip()`

E: this sequence achieves the goal

