UC Berkeley EECS
Head TA
Michael Ball

The Beauty and Joy of Computing

Lecture #18
Besides Blocks I:
Intro to Python

Amazon Dash, Not a joke!

Just before April 1st Amazon introduced the Amazon Dash, a "free" hardware button which orders a specific product when you click it. Amazon trademarked 1-click ordering in the 90's now they've created a literal clicker!

`https://www.amazon.com/oc/dash-button`

Admin Notes

- Reminder: Explore Project Part 2 is due Friday
  - Comments
  - Artifact + Explanation
- Final Projects start next week!
  - Use Snap! or Python
  - Individual
  - 3-weeks to work
- Demo: (review)) Exporting files from Snap!

Why Learn Python?

The Goals of BJC

- BJC's goal is not to teach you Snap*!*
- Teach you critical thinking about societal implications of computing
- Teach you how to program (Snap*!* is the best intro language we know) and help you succeed in the future
- More importantly: Teach you how to think like a computer scientist in life, called "computational thinking"

What is Computational Thinking?

- Using abstraction (removing detail and generalization with parameters)
- It's understanding the value of a "spec" that specifies a contract
- The iterative design cycle: design, prototype, implement, evaluate (loop)
- Thinking about how solutions scale, parallelize, generalize, and trying to foresee the unintended consequences!

Why Learn Python?

- You already know it!
  - Syntax is very similar to Snap!, especially like writing Snap! on paper.
- Python (also) runs everywhere
  - OS X and Windows, and now there are browser apps (Cloud9.io) and even iOS apps
- Lots of online support
- Plenty of advanced libraries
  - Everything from graphics processing to AI to games!
- Used in industry and academia

What You'll Learn

- New Syntax
  - Different way to write programs you already know how to write (in Snap!)
- A little bit about the command line
  - A text-based interface that exposes you to the internals of how a computer works
- How to find help online
- A little more about Object Oriented Programming
- Programming is really *not* about the language or environment

Demo

Intro To Python

Getting Python 3

- We'll be using Python 3 for this class.
  - It does have some minor changes from Python 2, but you don't need to worry about what they are.
- Download Python 3 from
  - https://python.org/downloads
  - Run the graphical installer
- All official Python documentation is at
  - https://docs.python.org

Intro to the Command Line

- "Terminal" on OS X (and Linux) and "Command Prompt" on Windows
  - a.k.a. "Unix shell" on Mac and Linux
- Does the same things as the rest of your computer (browse and edit files, even browse the web!)
- OS X:
  - Open: /Applications/Utilities/Terminal.app
- Windows:
  - Use the search bar for "cmd"

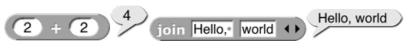
Python Programs

- Python programs are just a text file with Python syntax.
- To run a program you type:
  - `python  file_name.py`
  - (Sometimes this is `python3`)
  - Aside: a `moonscape font` indicates a command to run.
- Python has two modes – "normal" and "interactive"
  - Interactive mode happens if you don't provide a file to run.
  - After each command Python evaluates your code and returns the response. (Kind of like clicking a block in Snap*!*)
  - Use `python -i file_name.py`
    - Force a file to be run in interactive mode.

SNAP! ⟷ Python

Text and Numbers

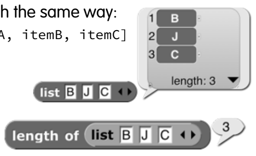
- Numbers in Python are called
  - `ints` (numbers w/o decimals)
  - `floats` (numbers with decimals)
- Strings:
  - Some text in between quotes "" or ' '



```
>>> 2 + 2
4
>>> "Hello, " + "world"
'Hello, world'
```

Lists

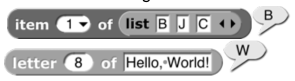
- Lists Work in much the same way:
  - Syntax: `[itemA, itemB, itemC]`



```
>>> ['B', 'J', 'C']
['B', 'J', 'C']
>>> len(['B', 'J', 'C'])
3
```

BEWARE: Item 0!

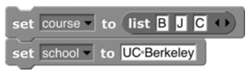
- However, there is one big difference!
- The first item in Python is item 0.
  - This also applies for strings as well.
- Access items using



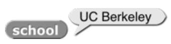
```
>>> letters = ['B', 'J', 'C']
>>> letters[0]
'B'
>>> 'Hello, World!'[7]
'W'
```

Variables

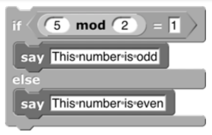
- No need to "declare" variable in Python,
  - Just use =
  - To access a variable, type it's name



```
>>> course = ['B', 'J', 'C']
>>> school = 'UC Berkeley'
>>> course
['B', 'J', 'C']
>>> school
'UC Berkeley'
```



Conditionals

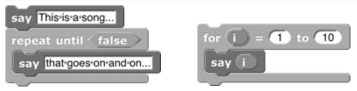


```
>>> if (5 % 2) == 1:
...     print('This number is odd')
... else:
...     print('This number is even')
...
This number is odd
```

Conditionals

- Conditionals Work the same way.
  - End the condition with `:`
    - Parentheses are optional around the condition.
  - Indent the body 1 "level" (usually 4 spaces
    - Indentation matters in Python!
  - To end a condition, just un-indent your code
- You can also see that mod in Python is a
- Note that the equals check is
- Python also supports an if (without the else) just like Snap*!*

Loops



```
>>> print('this is a song...')
this is a song...
>>> while(True):
...     print('that goes on and on...')
...
that goes on and on...
[omitted]
>>> for i in range(1, 11):
...     print(i)
...
1 [omitted 2…10[
```

Loops

- Loops are similar to conditionals.
- Instead of an "   " loop, Python has a "   " loop.
- Python is missing the   loop and the `forever` loop, but you can make these with whole and for loops.
- Note: `range()` is a function which is includes the first item, but not the last!
  - `range(1,10)` counts from 1 to 9!
  - You can use this function anywhere in Python.

Functions

```
>>> def factorial(n):
...     if n < 1:
...         return 1
...     else:
...         return n * factorial(n - 1)
...
>>> factorial(4)
24
```

Functions

- There is no distinction between a command, reporter or predicate.
  - You can simply use `return None` or just `return`
- Python uses the word `def`
- The body of function is indented
- All arguments are specified in `()` and must come at the end of the function name
- `report` → `return`
- Recursion works exactly the same as in Snap*!*
- Call a function like this: `name(arg1, arg2..)`

Summary

- Lots of little syntax differences!
  - The Python documentation is your friend
- Don't get too hung up on the differences and don't get discouraged when you get an error!
- There's so much more to Python in the coming weeks:
  - Python has thousands of additional, useful built in tools
  - Python supports HOFs and lambdas
  - Lots of cool libraries to explore (including turtle graphics)