A Lightning Introduction to Python

Will Monroe
CS 109 tutorial
13 April 2017

xkcd comic by Randall Munroe (no relation): https://xkcd.com/353/
def fizzbuzz(n):
    for i in range(1, n + 1):
        if i % 3 == 0 and i % 5 == 0:
            print('fizzbuzz')
        elif i % 3 == 0:
            print('fizz')
        elif i % 5 == 0:
            print('buzz')
        else:
            print(i)

fizzbuzz(100)
def fizzbuzz(n):
    for i in range(1, n + 1):
        if i % 3 == 0 and i % 5 == 0:
            print('fizzbuzz')
        elif i % 3 == 0:
            print('fizz')
        elif i % 5 == 0:
            print('buzz')
        else:
            print(i)

fizzbuzz(100)
Whitespace

• Whitespace matters in Python! Sometimes.

```python
if i % 3 == 0 and i % 5 == 0:
    print('fizzbuzz')
elif i % 3 == 0:
    print('fizz')
elif i % 5 == 0:
    print('buzz')
else:
    print(i)
```
Whitespace

- Whitespace matters in Python! Sometimes.

```python
if i % 3 == 0 and i % 5 == 0:
    print('fizzbuzz')
elif i % 3 == 0:
    print('fizz')
elif i % 5 == 0:
    print('buzz')
else:
    print(i)
```

- 4 spaces is a common convention.
- Don't mix spaces and tabs.
Whitespace

- Whitespace on otherwise blank lines is ignored.

```python
if i % 3 == 0 and i % 5 == 0:
    print('fizzbuzz')
elif i % 3 == 0:
    print('fizz')
elif i % 5 == 0:
    print('buzz')
else:
    print(i)
```
Whitespace

- Whitespace is ignored inside all brackets/parens.

```python
if (i % 3 == 0 and i % 5 == 0):
    print('fizzbuzz')
elif i % 3 == 0:
    print('fizz')
elif i % 5 == 0:
    print('buzz')
else:
    print(i)
```
Whitespace

- Whitespace is ignored inside all brackets/parens.

```python
if (i % 3 == 0 and i % 5 == 0):
    print('fizzbuzz')
elif i % 3 == 0:
    print('fizz')
elif i % 5 == 0:
    print('buzz')
else:
    print(i)
```
Whitespace

- Newline (+whitespace) is ignored after backslash.

```python
if (i % 3 == 0 and
    i % 5 == 0):
    print('fizzbuzz')
elif i % 3 == 0:
    print('fizz')
elif i % 5 == 0:
    print('buzz')
else:
    print(i)
```
Dynamic typing

Variables don't have types. **Values** do.
Dynamic typing

• A variable is created when you assign to it:

\[
x = 3
\]

• 3 is an integer. But \texttt{x} doesn't have to be—you can later give it a string value:

\[
\text{if } x \% 3 == 0:
    x = \text{'fizz'}
\]
Functions

- Arguments and return values can also be any type:

```python
def fizzbuzzify(i):
    if i % 3 == 0 and i % 5 == 0:
        return 'fizzbuzz'
    elif i % 3 == 0:
        return 'fizz'
    elif i % 5 == 0:
        return 'buzz'
    else:
        return i
```

WE HOPE THIS IS AN int. BUT IT CAN BE ANYTHING!

FUNCTION DEFINITIONS START WITH `def`. THAT'S ALL YOU NEED!
A few small spelling differences

<table>
<thead>
<tr>
<th>Java</th>
<th>C++</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>else if</td>
<td>else if</td>
<td>elif</td>
</tr>
<tr>
<td>true, false</td>
<td>true, false</td>
<td>True, False</td>
</tr>
<tr>
<td>&quot;string&quot;</td>
<td>&quot;string&quot;</td>
<td>'string',</td>
</tr>
<tr>
<td>// comment</td>
<td>// comment</td>
<td># comment</td>
</tr>
<tr>
<td>null</td>
<td>nullptr, NULL</td>
<td>None</td>
</tr>
</tbody>
</table>
The interactive interpreter
Python 2 or Python 3?

- Python 3:
  - fixes some annoying design decisions
  - has a bunch of awesome new features

- but
  - some libraries might not support it

The differences aren't large, but Python 3 is not backwards compatible!

Code in this tutorial should work in both.
String operations

- Make a string:
  ```python
  first_name = 'Will'
  ```

- Concatenate two strings:
  ```python
  last_name = 'Monroe'
  full_name = first_name + ' ' + last_name
  ```

- Get one character of a string
  ```python
  first_name[2]  # 'n'
  ```

- Including numbers in strings
  ```python
  age = 'I am {} years old'.format(6 * 4)
  ```
String operations: Slicing

• “Slice” = “substring”:

  ```
  >>> full_name[3:9]
  "l Monr"
  ```

  

  ```
  "Wil"
  ```

• Grab the first n characters:

  ```
  >>> full_name[:3]
  "Wil"
  ```

• ...or the last:

  ```
  >>> full_name[-3:]
  "roe"
  ```
## Containers

<table>
<thead>
<tr>
<th>Java</th>
<th>C++ <code>std::</code></th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayList</td>
<td>vector</td>
<td>list [1, 2]</td>
</tr>
<tr>
<td>HashMap</td>
<td>map</td>
<td>dict {'a': 1, 'b': 2}</td>
</tr>
<tr>
<td>HashSet</td>
<td>set</td>
<td>set {1, 2}</td>
</tr>
<tr>
<td>(n/a)</td>
<td>tuple</td>
<td>tuple (1, 2)</td>
</tr>
</tbody>
</table>
List operations: Building

- Make a list:
  ```python
  numbers = [1, 2, 3]
  ```
- Add a single value to the end:
  ```python
  numbers.append(4)
  ```
- Tack another list onto the end:
  ```python
  numbers.extend([5, 6])
  ```
- Concatenate two lists:
  ```python
  big_numbers = [7, 8, 9, 10]
  lots_of_numbers = numbers + big_numbers
  ```
List operations: Slicing

- Works the same way as strings:

  ```python
  >>> fruit = ['apple', 'banana', 'peach']
  >>> fruit[0]
  'apple'  # NO COLON: GET A SINGLE ELEMENT
  >>> fruit[1:3]
  ['banana', 'peach']
  >>> fruit[-1:]
  ['peach']  # WITH A COLON: GET A LIST
  ```
List operations: Slicing

- Make a list:
  ```python
  numbers = [1, 2, 3]
  ```
- Add a single value to the end:
  ```python
  numbers.append(4)
  ```
- Tack another list onto the end:
  ```python
  numbers.extend([5, 6])
  ```
- Concatenate two lists:
  ```python
  big_numbers = [7, 8, 9, 10]
  lots_of_numbers = numbers + big_numbers
  ```
def possible_passwords(length, smudges):
    if len(smudges) > length:
        return []
    if length == 0:
        return ['']

    passwords = []

    for i in range(0, len(smudges)):
        first = smudges[i]
        for suffix in possible_passwords(length - 1, smudges[:i] + smudges[i + 1:]):
            passwords.append(first + suffix)
        # Consider duplicates
        for suffix in possible_passwords(length - 1, smudges):
            passwords.append(first + suffix)

    return passwords
Set operations

• Make a set:
  >>> cats = {'Phoebe', 'Annie'}

• Add a single value to the set:
  >>> cats.add('Sylvester')

• Check if a value is in the set:
  >>> 'Tweety' in cats
  False

• Get the union of two sets:
  >>> cats.union({'Buster', 'Fido'})
  {'Buster', 'Annie', 'Phoebe', 'Fido'}
## Set operations

<table>
<thead>
<tr>
<th>Method</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a.union(b)</code></td>
<td>$a \cup b$</td>
</tr>
<tr>
<td><code>a.intersection(b)</code></td>
<td>$a \cap b$</td>
</tr>
<tr>
<td><code>a.difference(b)</code></td>
<td>$a - b$</td>
</tr>
<tr>
<td><code>a.symmetric_difference(b)</code></td>
<td>$a \cup b - a \cap b$</td>
</tr>
<tr>
<td><code>a.issubset(b)</code></td>
<td>$a \subseteq b$</td>
</tr>
<tr>
<td><code>a.isdisjoint(b)</code></td>
<td>$a \cap b = \emptyset$ ?</td>
</tr>
</tbody>
</table>

>>> `help(set)`
File reading

```python
with open('datafile.csv') as infile:
    for line in infile:
        print(line)
```
Appendix: Reading a CSV file

True,True,True,False,True,False
False,True,False,True,True,True

.
Appendix: Reading a CSV file

```python
with open('data.csv') as infile:
    data = []
    for line in infile:
        line = line[:-1]  # Strip off the final \n
        # line is 'True,True,False,...'
        # split(x) returns a list of substrings, separated by x
        row = line.split(',')
        data.append(row)

    # data is now a list of lists:
    #    [['True', 'True', 'False', ...],
    #     ['False', 'True', 'False', ...],
    #     ...]
```
The random library

```
random.randint(a, b)
# integer between a and b, inclusive
random.choice(seq)
# pick one element of seq, equally likely
random.shuffle(seq)
# shuffle seq, in-place
random.sample(seq, k)
# draw k elements of seq without replacement
random.random()
# uniform float in [0, 1)
```
Appendix: How to Defend Yourself Against Fresh Fruit

DO PROBABILISTIC SIMULATIONS WITH Monty Python's Flying Circus © Python (Monty) Pictures, Ltd.

GRATUITOUS MONTY PYTHON REFERENCE. PYTHON IS NAMED AFTER THEM!

He was attacking me with a banana.

Monty Python's Flying Circus © Python (Monty) Pictures, Ltd.
Appendix: Fresh fruit

From lecture Fri 4/7: 4 mandarins, 3 grapefruits in a bag. Draw 3. P(2 grapefruits, 1 mandarin)?

```python
import random

def grapefruit(num_draws):
    bag = [ 'mandarin', 'mandarin', 'mandarin', 'mandarin',
            'grapefruit', 'grapefruit', 'grapefruit']
    total_successes = 0
    for i in range(num_draws):
        draw = random.sample(bag, 3)
        if sorted(draw) == [ 'grapefruit', 'grapefruit', 'mandarin']:
            total_successes += 1
    return total_successes * 1.0 / num_draws

# Note: * 1.0 not necessary on Python 3
```

Answer: \( \frac{12}{35} = 0.3428... \)

Running with 10M draws gave me 0.3425...
class Mandarin(Fruit):
    def __init__(self, juiciness):
        self.juiciness = juiciness

    def peel(self, care):
        if self.juiciness - care > 9000:
            print('Squirt!')

extra_juicy = Mandarin(1000000)
exra_juicy.peel(0)