Plan

Python Basics
Data Structures
Regular Expressions
Practice!!
Intro to Python
>>> import this

The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Programmers are more important than programs
"Hello World" in Java

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```
"Hello World" in C++

```cpp
#include <iostream>

using namespace std;

int main() {
    cout << "Hello World!" << endl;
}
```

Double Yuck
“Hello World” in Python

print("Hello world!")
Running Python
Interactive Interpreter

(cs124-env) sredmond$ python3
Python 3.5.2 (default, Nov 12 2018, 13:43:14)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.

>>> You can write Python code right here!
Running Python Scripts

(cs124-env) sredmond$ python3 my_script.py
<output from the script>

(cs124-env) sredmond$ python3 hello.py
What is your name? Sam
Hey Sam, I'm Python!

(cs124-env) sredmond$
What is your name? Sam
Hey Sam, I'm Python!

>>> greet("Dan")
Hey Dan, I'm Python!

>>>
Python Basics
Comments

# Single line comments start with a '#'

......

Multiline comments can be written between three "s and are often used as function and module comments.

......
Numbers and Math

Python has two numeric types:

- **int**
- **float**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td># =&gt; 3 (int)</td>
</tr>
<tr>
<td>3.0</td>
<td># =&gt; 3.0 (float)</td>
</tr>
<tr>
<td>1 + 1</td>
<td># =&gt; 2</td>
</tr>
<tr>
<td>8 - 1</td>
<td># =&gt; 7</td>
</tr>
<tr>
<td>10 * 2</td>
<td># =&gt; 20</td>
</tr>
<tr>
<td>5 / 2</td>
<td># =&gt; 2.5</td>
</tr>
<tr>
<td>13 / 4</td>
<td># =&gt; 3.25</td>
</tr>
<tr>
<td>9 / 3</td>
<td># =&gt; 3.0</td>
</tr>
<tr>
<td>7 / 1.4</td>
<td># =&gt; 5.0</td>
</tr>
<tr>
<td>7 // 3</td>
<td># =&gt; 2 (integer division)</td>
</tr>
<tr>
<td>7 % 3</td>
<td># =&gt; 1 (integer modulus)</td>
</tr>
<tr>
<td>2 ** 4</td>
<td># =&gt; 16 (exponentiation)</td>
</tr>
</tbody>
</table>
### Booleans

<table>
<thead>
<tr>
<th>Condition</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td># =&gt; True</td>
</tr>
<tr>
<td>False</td>
<td># =&gt; False</td>
</tr>
<tr>
<td><code>not</code> True</td>
<td># =&gt; False</td>
</tr>
<tr>
<td>True and False</td>
<td># =&gt; False</td>
</tr>
<tr>
<td>True or False</td>
<td># =&gt; True (short-circuits)</td>
</tr>
</tbody>
</table>

- `1 == 1` # => True
- `2 * 3 == 5` # => False
- `1 != 1` # => False
- `2 * 3 != 5` # => True
- `1 < 10` # => True
- `2 >= 0` # => True
- `1 < 2 < 3` # => True (1 < 2 and 2 < 3)
- `1 < 2 >= 3` # => False (1 < 2 and 2 >= 3)

Booleans are a subtype of int, where `true == 1` and `false == 0`
# Unicode by default

```python
greeting = 'Hello'
group = "wørld"
```

# Concatenate

```python
combo = greeting + ' ' + group + '!'  # combo == "Hello wørld!!"
```

```
combo == "Hello wørld!!"  # => True
```

---

No char in Python! Both ' and " make string literals
If Statements

```
if the_world_is_flat:
    print("Don't fall off!")
```

- No parentheses needed
- Colon
- No curly braces!
- Use 4 spaces for indentation
elif and else

if name == "jurafsky":
    print("dan is lecturing")
elif name == "sredmond":
    print("sam is lecturing")
else:
    print("someone else is lecturing")

else is optional

Python has no switch statement, opting for if/elif/else chains

zero or more elifs
For Loops

Loop explicitly over data

Strings, lists, etc.

```
for item in iterable:
    process(item)
```

No loop counter!
# Loop over words in a list.
for color in ["red", "green", "blue", "yellow"]:  
    print(color)

# => "red"
# => "green"
# => "blue"
# => "yellow"
Writing Functions

The `def` keyword defines a function

```python
def fn_name(param1, param2):
    value = do_something()
    return value
```

Parameters have no explicit types

Return is optional
If either return or its value are omitted, implicitly returns None
Data Structures
easy_as = [1, 2, 3]

Lists

Square brackets delimit lists

Commas separate elements
Inspecting List Elements

letters = ['a', 'b', 'c', 'd']
numbers = [2, 3, 5, 7, 11]

# Access elements at a particular index
numbers[0]  # => 2
numbers[-1]  # => 11

# You can also slice lists - the same rules apply
letters[:3]  # => ['a', 'b', 'c']
numbers[2:-1]  # => [5, 7]
# General Queries

# Length (len)
len([])  # => 0
len("python")  # => 6
len([4, 5, "seconds"])  # => 3

# Membership (in)
0 in []  # => False
'y' in "python"  # => True
"minutes" in [4, 5, "seconds"]  # => False
empty = {}
d = {"one": 1, "two": 2, "three": 3}

# Get
d['one']  # => 1
d['five']  # raises KeyError

# Set
d['two'] = 22  # Modify an existing key
d['four'] = 4  # Add a new key
re
Regular expression operations

"regular expression" == "search pattern" for strings
# Search for pattern match anywhere in string; return None if not found
m = re.search(r"(\w+) (\w+)", "Isaac Newton, Physicist")

m.group(0)  # "Isaac Newton" — the entire match
m.group(1)  # "Isaac" — first parenthesized subgroup
m.group(2)  # "Newton" — second parenthesized subgroup

# Match pattern against start of string; return None if not found
m = re.match(r"(?P<fname>\w+) (?P<lname>\w+)", "Malcolm Reynolds")

m.group('fname')  # => 'Malcolm'
m.group('lname')   # => 'Reynolds'
```python
# Substitute occurrences of one pattern with another
re.sub(r'@\w+\.com', '@stanford.edu', 'sam@cal.com jurafsky@bears.com')
# => sam@stanford.edu jurafsky@stanford.edu

pattern = re.compile(r'[a-z]+[0-9]{3}')  # compile pattern for fast ops
match = re.search(pattern, '@@@abc123')  # pattern is first argument
match.span()  # (3, 9)
```
def is_phone(num):
    return bool(re.match(r'\d{3} \d{3}-\d{4}', num))

def get_area_code(num):
    m = re.match(r'(\?P<areacode>\d{3}) \d{3}-\d{4}', num)
    if not m:
        return None
    return m.group('areacode')
Virtual Environments Primer

Local, isolated Python environment for interpreter, third-party libraries, and scripts
Practical Commands

sredmond$ source ~/cs124-env/bin/activate
(cs124-env) sredmond$ python3 --version
Python 3.5.2

(cs124-env) sredmond$ deactivate
sredmond$

Always activate your virtual environment before beginning to code!
Your Turn!
Hello, World!
Variables and Types
Lists
Basic Operators
String Formatting
Basic String Operations
Conditions
Loops
Functions
Dictionaries

Modules and Packages
Numpy Arrays
Generators
List Comprehensions
Regular Expressions
Sets
Decorators

https://www.learnpython.org/