



April 6, 2019

Urbana, IL

# ACI-UIUC Python Workshop: Basic Python

# Before we begin...

- If you haven't already, go to <https://github.com/sushobhansen/aci-python-workshop> for course materials
- Download and install Anaconda:  
<https://www.anaconda.com/distribution/>
- Or use an online IDE: <https://repl.it/languages/python3>

# Who am I?



- Sushobhan Sen
- Doctoral candidate in Civil Engineering (Transportation) – I study stratified urban boundary layer flows
- Languages I know: FORTARN, C/C++/C#, Python, Visual Basic, HTML, CSS, JavaScript, Latex, Bash
  - (I also speak a few human languages)

# Workshop Website:

<https://github.com/sushobhansen/aci-python-workshop>

sushobhansen / aci-python-workshop

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

A repo for the Python workshop for ACI-UIUC (April 6, 2019) Edit

Manage topics

22 commits 1 branch 0 releases 1 contributor GPL-3.0

Branch: master New pull request Create new file Upload files Find File Clone or download

sushobhansen Updated objectives and cleared output from notebooks Latest commit c4b94f 3 days ago

data	Added Pandas and conclusion	3 days ago
notebooks	Updated objectives and cleared output from notebooks	3 days ago
scripts	Added Pandas and conclusion	3 days ago
.gitignore	Created initial notes file	6 days ago
LICENSE	Initial commit	10 days ago
README.md	Added installation and online IDE info	6 days ago
notes.pdf	Updated objectives and cleared output from notebooks	3 days ago

See this

# Learning Objectives: Basic Python

- At the end of this workshop, participants will be able to:
  - List the types of Python variables and define them
  - Add control statements and loops to their programs
  - Define and use functions
  - Define and use object oriented programming (if we have time)
- We'll also solve a problem using all these skills

# Why Python?

- According to IEEE, it's the most popular programming language today
- Simple, extensible, powerful
- Latest version: Python 3.7 (Python 3.x is not backwards compatible with Python 2.x, so stick to 3.x)

# Variables

Data Type	Syntax	Description	Comments
Integer	<code>x = 5</code>	Signed integer	Use <code>int</code> to typecast, if valid
Float	<code>x = 5.</code>	IEEE floating point number	Use <code>float</code> to typecast, if valid
Complex	<code>x = 3.1+4.6j</code>	Complex number	Use <code>complex</code> to typecast, if valid
String	<code>x = 'Hello World!'</code>	Strings are always enclosed within quotation marks	Use <code>str</code> to typecast, slicing operator valid
List	<code>x = [1, 2.2, 'otter']</code>	Mutable list of variables of any type	Use <code>list()</code> to typecast, if valid
Tuple	<code>x = (1, 2.2, 'otter')</code>	Immutable tuple of variables of any tupe	Use <code>tuple()</code> to typecast, if valid
Dictionary	<code>x = {'one':1, 'two':2}</code>	Key-value pairs	Use <code>get()</code> to get value from key, use <code>dict()</code> to typecast, if valid
Bool	<code>x = True</code>	Boolean value	Python uses keywords <code>True</code> and <code>False</code> , not 1 and 0

# Variable Names

1. The name **cannot** contain spaces - consider using an underscore character or camelCase instead for readability
2. The name cannot start with a number, but can contain a number anywhere
3. The name is case-sensitive, so `temp` and `Temp` are different variables
4. Python has a number of reserved keywords (see documentation, of just follow along and you'll get the hang of it), which cannot be used as names

*Go to Notebook* 



# Mutability

- Immutable: Most objects (int, float, complex, str, bool, tuple)
- Mutable: Lists and dictionaries

*Go to Notebook* 

# Operators

- **Arithmetic:** +, -, \*, /, %, \*\*, //
- **Comparison:** >, <, ==, !=, >=, <=
- **Logical:** and, or, not
- **Bitwise:** &, |, ~, ^, <<, >>
- **Assignment:** =, any combination of arithmetic or bitwise with =
- **Special:** is, is not, in, not in

*Go to Notebook* 

# Control Statements

1. **Conditional statements:** Better known as if-else blocks, these test a condition before executing a line
2. **Loops:** These repeatedly execute a line for a certain number of times or till a break condition is met

# A simple if statement

```
1 x = 6
2 if x<5:
3     print('x is less than 5')
4     print('Finished evaluating if block')
5 print('Finished this code block')
```

- Notice the indentation: **very important!**

*Go to Notebook* 

# Loops

- While loop: Repeat till a condition is met
- For loop: Repeat over a fixed number of iterations

```
1 x = 3
2 while(x<6):
3     print('The value of x is ',x)
4     x+=1
```

# For loops

- Loop over an **iterator**

- An iterator is an object that contains a fixed number of values (any values)
- Iterate through the values in the iterator, not just a fixed number of times (unlike C++)

```
1 x = range(3)
2 print(list(x))
3 for i in x:
4     print('The value of i is ',i)
```

*Go to Notebook* 

# Break, continue, pass


- **Break:** When executed, break out of loop entirely without any more iterations
- **Continue:** When executed, skip everything after that line and then proceed to next iteration
- **Pass:** When executed, continue to next line (as if nothing happened)

*Go to Notebook* 

# Functions

- Functions are bits of code that are separated from the main function ***(they have their own scope)***, but can be called at any point by the main function to perform a specific task.

Function **def** statement



```
1 def print_input(x):  
2     print(x)  
3  
4 k = 'Hello'  
5 print_input(k)  
6  
7 j = 'How are you'  
8 print_input(j)  
9  
10 x = 'I am good'  
11 print_input(x)
```

[Go to Notebook](#) 



# Example: System of Linear Equations

- Solve:  $\begin{cases} ax + by = e \\ cx + dy = f \end{cases} \Rightarrow \begin{cases} x = (de - bf)/(ad - bc) \\ y = (af - ce)/(ad - bc) \end{cases}$
- Unique solution exists only if  $ad - bc \neq 0$

Go to Notebook 

# Default Args

```
1 def addnumber(x,n=5):  
2     y = []  
3     for i in x:  
4         y.append(i+n)  
5     return y  
6  
7 a = range(3)  
8 print(addnumber(a))  
9 print(addnumber(a,n=2))
```

*Go to Notebook* 

# Classes and Objects

- Class: Set of related variables and functions that are grouped together
- Object: An instance of a class

## Defining a class

```
1 class Person:
2     def __init__(self, name, age, height_cm): ← Constructor
3     self.name = name
4     self.age = age
5     self.height_cm = height_cm #in centimeters
6
7     def print_name(self):
8         print(self.name)
9
10    def print_age(self):
11        print(self.age)
12
13    def print_height_cm(self):
14        print(self.height_cm)
15
16    def height_ftin(self):
17        inches = self.height_cm/2.54
18        feet = inches//12
19        inches = inches%12
20        return feet, inches
```

Dot operator

Always pass self as first arg

Go to Notebook →

# Inheritance

- A class can inherit all the functions and variables from a *base* class, and then add or modify them

Student class inherits from  
Person class

```
1 class Student(Person):
2     def __init__(self, name, age, height_cm, GPA, level):
3         Person.__init__(self, name, age, height_cm)
4         self.GPA = GPA
5         self.level = level
6
7     def print_name(self, status='regular'):
8         print(self.name, ' STUDENT: ', status)
9
10    def print_GPA(self):
11        print(self.GPA)
12
13    def print_level(self):
14        print(self.level)
15
```

Polymorphism  
ABCs

Go to Notebook 

# Challenge

- Use Python to write a `prime_check()` function, which takes in an integer and returns the following:
  - Negative integers or zero: “Invalid input”
  - If 1 or 2: “Prime number”
  - Any other integer: “This is a prime number” or “This is not a prime number”
- Algorithm for prime numbers:
  - **Sequentially divide the integer  $n$  by integers between 2 and  $n/2$**
  - If at any point, the remainder is zero, it is a prime number, otherwise it isn't

*Go to Notebook* 

# Further Resources

- Python Documentation: <https://docs.python.org/3/>
- Free Python course on EdX: <https://www.edx.org/professional-certificate/introduction-to-computing-in-python>
- Free e-Book:  
<https://www.cs.uky.edu/~keen/115/Haltermanpythonbook.pdf>
- Stick around the Advanced Python session in the afternoon