
ISA 414 – Managing Big Data

Lecture 3 – Preliminaries (Part II)

Built-in Functions, Variables, and Control Flow

Dr. Arthur Carvalho

arthur.carvalho@miamioh.edu



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

Lecture Objectives

- Learn basic commands in Python
 - How to declare variables
 - How to use built-in functions
 - How to control the flow of Python scripts

Lecture Instructions

1. Download the notebook “*Lecture 3.ipynb*” available on Canvas
2. Open the file “*Lecture 3.ipynb*” with VS Code

Introduction to Python

- Python was released in 1991
- We will focus on Python as used in data science
 - For example, using Jupyter notebooks
 - There is much more to it
- Currently, Python can be used for:
 - Data analytics
 - Scientific calculations
 - System/software development
 - Web development
 - ...

Introduction to Python

- Jupyter notebooks will abstract away many difficulties when working with Python
 - *E.g.*, the need to work with command lines
 - Consequently, we do not experience the full power of Python in this course
- As we progress, keep in mind that Python uses new lines to complete a command
 - Other programming languages often use semicolons or parentheses

VARIABLES

Introduction to Python

➤ Variables

- Can be used to store previously calculated values
- *E.g.*, saving the result of $1+1+1$ to a variable called `x`

`x = 1 + 1 + 1`

Name	▲ Type	Size	Value
x	int		3

- Note that the user-defined variable `x` can now be used in future computations

• *E.g.*, try the following command: `print(x + 2)`

Introduction to Python

- We saw the traditional way of assigning **values** to **variables**
 - One value to one variable
- Python allows one the declare variables in different ways
 - Many values to many variables
 - *E.g.*, `x, y, z, = 1, 2, 3`
 - One value to many variables
 - *E.g.*, `a = b = c = 10`
 - Many values to one variable
 - *E.g.*, **lists** (see future classes)

Introduction to Python

- Python has several built-in variable/data structure types

Text Type:	<code>str</code>
Numeric Types:	<code>int</code> , <code>float</code> , <code>complex</code>
Sequence Types:	<code>list</code> , <code>tuple</code> , <code>range</code>
Mapping Type:	<code>dict</code>
Set Types:	<code>set</code> , <code>frozenset</code>
Boolean Type:	<code>bool</code>
Binary Types:	<code>bytes</code> , <code>bytearray</code> , <code>memoryview</code>

Covered today

Covered in the future

Not covered in this course

- So, where is “data frame”?
 - Not a built-in data structure (see future lectures)
- One can use the `type` function to determine the type of a variable
 - Examples in the following slides

Introduction to Python

➤ Numeric variables

- Store numbers
- Try the following:
 1. Assign the value 10.5 to a variable called *x*
 2. Apply the function `type` to *x*
 3. What is the result?
- Next, try the following:
 1. Assign the value 10 to a variable called *x*
 2. Apply the function `type` to *x*
 3. What is the result?

Introduction to Python

➤ Boolean variables

- Store Boolean values (**True** or **False**) often resulting from logical comparisons
 - Internally, **True** = 1, **False** = 0
- Try the following:
 - Assign the value **10 > 5** to a variable called **x**
 - Assign the value **-20 > 0** to a variable called **y**
 - What is the value of **x**? What is the value of **y**?
 - What is the type of **x**? What is the type of **y**?

Introduction to Python

➤ Boolean variables

- Standard logical operators: `==` (equal), `!=` (not equal) `and`, `or`, `not` (negation)
- Try the following:
 - Set the variables `x = True` and `y = False`
 - What is the value of `x == y` ?
 - What is the value of `x != y` ?
 - What is the value of `x and y` ?
 - What is the value of `x or y` ?
 - What is the value of `not x` ?
 - What is the value of `x + 5`?
 - What is the value of `y - 4`?

Introduction to Python

➤ String variables

- Used to represent string values (texts) in Python
 - Delimited by quotes (either single or double)
- Crucial when working with textual data
 - More details in the future
- Example: `my_first_string = "ISA 414/515"`

Name	▲ Type	Size	Value
my_first_string	str	11	ISA 414/515

Introduction to Python

- Why do attribute types matter?
 - Certain operations require specific attribute types
 - Without running any code, answer the questions below:
 - How much is "Arthur" + "Carvalho" ?
 - How much is `pow("Arthur", 2)`, *i.e.*, Arthur²?
 - How much is `(True + False)*True` ?

Introduction to Python

- One can always try to coerce an attribute type into another one by using the functions `int`, `float`, `bool`, and `str`
 - Example:

```
w = "1"  
x = int(w)  
y = float(x)  
z = bool(x)
```

Name	▲ Type	Size	Value
w	str	1	1
x	int		1
y	float		1.0
z	bool		True

BUILT-IN FUNCTIONS

Introduction to Python

- Python comes loaded with built-in functions
 - A function is a block of code that only runs when it is **called**
 - One can **pass** data, known as **arguments** or **parameters**, into a function
 - A function can **return** data as a **result**
- We have already seen and used a few functions
 - `type()`, `str()`, `int()`, `float()`, and `bool()`
- Another crucial function is `print()`
 - Print the value of variables
 - Example: `print(x)`

Introduction to Python

➤ Documentation

- Typing the question mark ('?') character after a built-in function name returns the documentation associated with that function
- Example: try `pow?`

Signature: `pow(base, exp, mod=None)`

Docstring:

Equivalent to `base**exp` with 2 arguments or `base**exp % mod` with 3 arguments

Some types, such as ints, are able to use a more efficient algorithm when invoked using the three argument form.

Type: `builtin_function_or_method`

Introduction to Python

- Applying predefined functions
 - Most functions we will be using in this course have more than one argument
 - For example, `pow`
 - First argument = base
 - Second argument = exponent
 - The notation `mod=None` means that if the third argument is not provided, then the **default value** is *none*

Signature: `pow(base, exp, mod=None)`

Docstring:

Equivalent to `base**exp` with 2 arguments or `base**exp % mod` with 3 arguments

Introduction to Python

- When one calls a function, the order of the arguments matter, otherwise unexpected behavior might occur
 - For example: `pow(10, 2)` is not the same as `pow(2, 10)`
- Make sure the order of the arguments is the same as the order listed in the documentation of the function
- Another option is to explicitly use the name of the arguments when calling a function
 - The order of the arguments no longer matters
 - Example: `pow(base = 10, exp = 2)`
`pow(exp = 2, base = 10)`

Signature: `pow(base, exp, mod=None)`

CONTROL FLOW

Introduction to Python

- Our code has been very structured thus far
 - One command after the other
- We can specify conditions that must be satisfied before our code is executed
 - Control the flow of the code
- **CRUCIAL POINT TO REMEMBER**
 - Python relies on **indentation** to define “scope” (blocks of code)
 - *E.g.*, the scope of loops and functions
 - Other languages, such as R, use curly brackets
 - **Indentation** = spaces at the beginning of a code line

Introduction to Python

- The best way to define indentation is by using the **tab** (tabulator) key
 - By default, it creates 4 spaces in VS Code
 - This can be changed
 - By using tab, you do not have to memorize how many spaces you are using
- Technically speaking, any number of spaces is fine
 - As long as that number is the same inside each block of code

Introduction to Python

➤ Control flow

- There are three major ways of controlling the order in which individual commands are executed in Python

1. IF-ELSE statement:

```
if <logical_expression>:  
    ...  
Syntax: elif <logical_expression> :  
    ...  
else:  
    ...
```


Introduction to Python

1. IF-ELSE statement:

- The “**elif**” and “**else**” statements are optional
- Example:

```
x = 5
if x > 0 :
    print("Non-negative number")
elif x < 0:
    print("Negative number")
else:
    print("the number is 0")
```

Introduction to Python

1. IF-ELSE statement:

- Why do you get an error with the code below?

Code #1

```
x = 5
if x > 0 :
    print("Non-negative number")
```

Code #2

```
x = 5
if x > 0 :
    print("Non-negative number")
    print("Non-negative number - print 2")
```

Introduction to Python

2. FOR statement (loop):

- Repeats a group of commands for each possible value in a sequence/list of values

Syntax:

for value in sequence:

...

Example:

```
for x in range(10):  
    if x > 5:  
        print(x)
```

- Note: by default, the range function returns a sequence of numbers from 0 to the argument value minus 1

`range(10) = 0, 1, ..., 9`

Introduction to Python

2. FOR statement (loop):

- One can specify different values inside `range()`
 - Syntax: `range(start, end, by)`
 - Example:

```
for x in range(-1, 10, 2):  
    print(x)
```

Introduction to Python

2. FOR statement (loop):

- Python allows one to easily iterate over a string
 - Example:

```
for x in "Arthur":  
    print(x)
```

- Python allows one to iterate over values in a **list**
 - More on that in the future

```
for x in ["John", "Paul", "George", "Ringo"]:  
    print(x)
```

Introduction to Python

3. WHILE statement (loop)

- Repeats a group of commands until the Boolean condition ceases to apply

Syntax:

```
while <logical_expression>:  
    ...
```

Example:

```
x = 5  
while x > 0:  
    print(x)  
    x = x - 1
```

Homework #2

Homework #2

- You have been recently hired to work in the analytics team at Goldman Sachs
 - Your team is responsible for building models that predict stock prices
 - Instead of using a single model, each member of your team is responsible for creating one individual model
 - Individual models are later aggregated to form an *ensemble model*
 - You are responsible for developing a model based on Brownian motion

Homework #2

- The simplest version of Brownian motion has the form: $price_t = price_{t-1} + \epsilon$ where $\epsilon \sim N(0, \sigma)$
 - The standard deviation σ represents the uncertainty in the price movement. The subscript t represents an end-of-day stock price
- **Task:** write a script that simulates a single Brownian motion over 100 days
 - Start by importing the module `random` in the first line of your code
 - More on this in future classes
 - `import random`
 - In the second line, assume the initial stock price is equal to 50
 - `price = 50`
 - Hint: write a `for` loop that iterates 100 times. Assume $\sigma = 0.01$
 - In each iteration, draw a value from a normal distribution using the function `random.gauss(mu = 0, sigma=0.01)` and update the value stored in `price`
 - Upload your code on Canvas (Homework 2)

Summary

- We have learned the basics of Python
 - How to declare variables
 - How to use built-in functions
 - How to control the flow of Python scripts
- Next lecture
 - Advanced variable types and data structures

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.