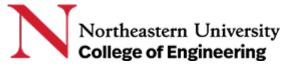
Python for Machine Learning and Data Analysis

Handan Liu, PhD
Northeastern University
Spring 2019



Python Language Essentials

Introduction to Python's core language features and packages Lecture 2



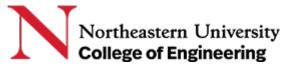
- Hello, World!
- Variables and Types
- Lists
- Basic Operators
- String Formatting
- Basic String Operations
- Conditions
- Loops
- Functions
- Classes and Objects
- Dictionaries
- Modules and Packages



Hello World!

- Python 2 and 3
- The simplest directive in Python is the "print" directive
- Indentation
 - Python uses indentation for blocks, instead of curly braces (like in C/C++). Both tabs and spaces are supported, but the standard indentation requires standard Python code to use 4 spaces. For example:

```
x = 1
if x == 1:
  # indented 4 spaces
  print("x is 1.")
```



• Use the "print" command to print the line "Hello, World!".

- Hello, World!
- Variables and Types
- Lists
- Basic Operators
- String Formatting
- Basic String Operations
- Conditions
- Loops
- Functions
- Classes and Objects
- Dictionaries
- Modules and Packages



Variables and Types

- Object oriented: not "statically typed"
- Numbers
 - Integer
 - Floating point number
- Strings
 - Strings are defined either with a single quote or a double quotes.
 - The difference between the two is that using double quotes makes it easy to include apostrophes (whereas these would terminate the string if using single quotes)

- The target of this exercise is to create a string, an integer, and a floating point number.
 - The string should be named mystring and should contain the word "hello".
 - The floating point number should be named *myfloat* and should contain the number 10.0, and
 - the integer should be named myint and should contain the number 20.

01/17/2019

- Hello, World!
- Variables and Types
- Lists
- Basic Operators
- String Formatting
- Basic String Operations
- Conditions
- Loops
- Functions
- Classes and Objects
- Dictionaries
- Modules and Packages

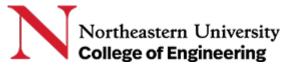


List

- Lists are very similar to arrays.
- They can contain any type of variable, and they can contain as many variables as you wish.
- Lists can also be iterated over in a very simple manner.
- Here is an example of how to build a list: "append" list
- Accessing an index which does not exist generates an exception (an error).

- In this exercise, you will need to add numbers and strings to the correct lists using the "append" list method. You must add the numbers 1, 2, and 3 to the "numbers" list, and the words 'hello' and 'world' to the strings variable.
- You will also have to fill in the variable second_name with the second name in the names list, using the brackets operator [].
- Note that the index is zero-based, so if you want to access the second item in the list, its index will be 1.

- Hello, World!
- Variables and Types
- Lists
- Basic Operators
- String Formatting
- Basic String Operations
- Conditions
- Loops
- Functions
- Classes and Objects
- Dictionaries
- Modules and Packages



Basic Operators

- Arithmetic Operators
 - Just as any other programming languages, the addition, subtraction, multiplication, and division operators can be used with numbers.
 - Modulo (%) Operator: return the integer remainder of the division; dividend % divisor = remainder
 - Power: using two multiplication symbols makes a power relationship.
- Using Operators with Strings
 - Python supports concatenating strings using the addition operator.
 - Python also supports multiplying strings to form a string with a repeating sequence.
- Using Operators with Lists
 - Lists can be joined with the addition operators.
 - Just as in strings, Python supports forming new lists with a repeating sequence using the multiplication operator.



- The target of this exercise is to create two lists called x_list and y_list, which contain 10 instances of the variables x and y, respectively.
- You are also required to create a list called big_list, which contains the variables x and y, 10 times each, by concatenating the two lists you have created.

- Hello, World!
- Variables and Types
- Lists
- Basic Operators
- String Formatting
- Basic String Operations
- Conditions
- Loops
- Functions
- Classes and Objects
- Dictionaries
- Modules and Packages



String Formatting

- Python uses C-style string formatting to create new, formatted strings.
 - "%" operator is used to format a set of variables enclosed in a "tuple" (a fixed size list), together with a format string, which contains normal text together with "argument specifiers", special symbols like "%s" and "%d".
- Let's say you have a variable called "name" with your user name in it, and you would then like to print(out a greeting to that user.) it's easy to understand and learn.

```
# This prints out "Hello, John!"
name = "John"
print("Hello, %s!" % name)
```



• To use two or more argument specifiers, use a tuple (parentheses):

```
# This prints out "John is 23 years old."
name = "John"
age = 23
print("%s is %d years old." % (name, age))
```

 Any object which is not a string can be formatted using the %s operator as well:

```
# This prints out: A list: [1, 2, 3]
mylist = [1,2,3]
print("A list: %s" % mylist)
```

- Here are some basic argument specifiers you should know:
 - %s String (or any object with a string representation, like numbers)
 - %d Integers
 - %f Floating point numbers
 - %.<number of digits>f Floating point numbers with a fixed amount of digits to the right of the dot.
 - %x/%X Integers in hex representation (lowercase/uppercase)

 You will need to write a format string which prints out the data using the following syntax:

Hello John Doe. Your current balance is \$53.44

- Hello, World!
- Variables and Types
- Lists
- Basic Operators
- String Formatting
- Basic String Operations
- Conditions
- Loops
- Functions
- Classes and Objects
- Dictionaries
- Modules and Packages



Basic String Operations

- Strings are bits of text. They can be defined as anything between quotes:
 - Double quotes and single quotes
 - How to print: this is 'my special'
- Length of a string
- Location of the first occurrence of a letter in a string
- Count the number of a letter in a string
- Print a slice of a string
- Print characters skipping: [start:stop:step]



- Reverse a string
- Convert lowercase and uppercase
- Split

Try to fix the code to print out the correct information by changing the string.

- Hello, World!
- Variables and Types
- Lists
- Basic Operators
- String Formatting
- Basic String Operations
- Conditions
- Loops
- Functions
- Classes and Objects
- Dictionaries
- Modules and Packages



Conditions

- Boolean
 - True
 - False
 - =
 - ==
 - !=

Boolean operators

- "and" operator
- "or" operator
- "in" operator
- "is" operator
- "not" operator

"if" statement

- Python uses indentation to define code blocks, instead of brackets.
 - The standard Python indentation is 4 spaces, although tabs and any other space size will work, as long as it is consistent.
- Notice that code blocks do not need any termination.

Here is an example for using Python's "if" statement using code blocks:

```
if <statement is="" true="">:
    <do something="">
elif <another statement="" is="" true="">: # else if
    <do something="" else="">
else:
    <do another="" thing="">
```

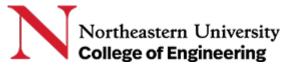
- A statement is evaluated as true if one of the following is correct:
 - 1. The "True" Boolean variable is given, or calculated using an expression, such as an arithmetic comparison.
 - 2. An object which is not considered "empty" is passed.

- Here are some examples for objects which are considered as empty:
 - An empty string: ""
 - 2. An empty list: []
 - 3. The number zero: 0
 - 4. The false Boolean variable: False



• Change the variables in the first section, so that each if statement resolves as True.

- Hello, World!
- Variables and Types
- Lists
- Basic Operators
- String Formatting
- Basic String Operations
- Conditions
- Loops
- Functions
- Classes and Objects
- Dictionaries
- Modules and Packages



Loops

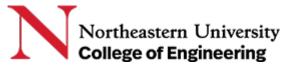
- There are two types of loops in Python: for and while.
- "for" loops
 - "range" function: returns a new list with numbers of that specified range
 - Note: the range function is zero based.
- "while" loop
 - Repeat as long as a certain Boolean condition is met.
- "break" and "continue" statements
 - "break" is used to exit a for-loop or a while-loop
 - "continue" is used to skip the current block, and return to the "for" or "while" statement.

can we use "else" clause for loops?

- YES!
- When the loop condition of "for" or "while" statement fails then code part in "else" is executed.
- If "break" statement is executed inside for loop then the "else" part is skipped.
- Note that "else" part is executed even if there is a "continue" statement.

- Loop through and print out all even numbers from the numbers list in the same order they are received.
- Don't print any numbers that come after 237 in the sequence.

- Hello, World!
- Variables and Types
- Lists
- Basic Operators
- String Formatting
- Basic String Operations
- Conditions
- Loops
- Functions
- Classes and Objects
- Dictionaries
- Modules and Packages



Functions

- What are Functions?
 - A convenient way to divide your code into useful blocks, allowing you to order your code, make it more readable, reuse it and save some time.
 - Also functions are a key way to define interfaces so programmers can share their code.
- How do you write functions in Python?
 - Python makes use of blocks.
 - A block is a area of code of written in the format of:
 - Block head: its format is block_keyword block_name(argument1, argument2, ...)
 - Block keyword: "def", followed with the function's name as the block's name.



block_head:

1st block line

2nd block line

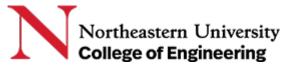
- How do you call functions in Python?
 - Simply write the function's name followed by (), placing any required arguments within the brackets.

In this exercise you'll use an existing function, and while adding your own to create a fully functional program.

- Add a function named list_benefits() that returns the following list of strings: "More organized code", "More readable code", "Easier code reuse", "Allowing programmers to share and connect code together"
- Add a function named build_sentence(info) which receives a single argument containing a string and returns a sentence starting with the given string and ending with the string " is a benefit of functions!"
- Run and see all the functions work together!

Learn the Basics

- Hello, World!
- Variables and Types
- Lists
- Basic Operators
- String Formatting
- Basic String Operations
- Conditions
- Loops
- Functions
- Classes and Objects
- Dictionaries
- Modules and Packages



Classes and Objects

- Objects are an encapsulation of variables and functions into a single entity.
- Objects get their variables and functions from classes.
- Classes are essentially a template to create your objects.
- A very basic class would look something like this:

```
class MyClass:
    variable = "MSIS"

    def function(self):
        print("This is a message inside the class.")

myobjectx = MyClass()
```

Accessing Object Variables

 To access the variable inside of the newly created object "myobjectx" you would do the following:

```
class MyClass:
                                                  class MyClass:
  variable = "MSIS"
                                                    variable = "MSIS"
 def function(self):
                                                    def function(self):
    print("This is a message inside the class.")
                                                      print("This is a message inside the class.")
myobjectx = MyClass()
                                                  myobjectx = MyClass()
myobjectx.variable
                                                  myobjecty = MyClass()
print(myobjectx.variable)
                                                  myobjecty.variable = "MSCSE"
                                                  # Then print out both values
                                                  print(myobjectx.variable)
                                                  print(myobjecty.variable)
```



Accessing Object Functions

 To access a function inside of an object you use notation similar to accessing a variable:

```
class MyClass:
    variable = "MSIS"

def function(self):
    print("This is a message inside the class.")
myobjectx = MyClass()

myobjectx.function()
```

Exercise

- We have a class defined for vehicles.
- Create two new vehicles called car1 and car2.
- Set car1 to be a red convertible worth \$60,000.00 with a name of Fer, and car2 to be a blue van named Jump worth \$25,000.00.

Learn the Basics

- Hello, World!
- Variables and Types
- Lists
- Basic Operators
- String Formatting
- Basic String Operations
- Conditions
- Loops
- Functions
- Classes and Objects
- Dictionaries
- Modules and Packages



Dictionaries

- A dictionary is a data type similar to arrays, but works with keys and values instead of indexes.
- Each value stored in a dictionary can be accessed using a key, which is any type of object (a string, a number, a list, etc.) instead of using its index to address it.
- For example, a database of phone numbers could be stored using a dictionary like this:

Dictionary

- Iterating over dictionaries
 - Dictionaries can be iterated over, just like a list.
 - However, a dictionary, unlike a list, does not keep the order of the values stored in it.
 - To iterate over key value pairs, use the following syntax: items()
- Removing a value
 - To remove a specified index, use either one of the following notations:

Exercise

• Add "Jake" to the phonebook with the phone number 857273443, and remove Jill from the phonebook.

Learn the Basics

- Hello, World!
- Variables and Types
- Lists
- Basic Operators
- String Formatting
- Basic String Operations
- Conditions
- Loops
- Functions
- Classes and Objects
- Dictionaries
- Modules and Packages



Modules and Packages

- What is a module?
 - In programming, a module is a piece of software that has a specific functionality.
- Writing modules
 - Modules in Python are simply Python files with a .py extension.
 - The name of the module will be the name of the file.
 - A Python module can have a set of functions, classes or variables defined and implemented. In the example above, we will have two files, we will have:

```
mygame/game.py
mygame/draw.py
```



 Modules are imported from other modules using the import command. In this example, the game.py script may look something like this:

```
# game.py
# import the draw module
```

• The draw module may look something like this:

- Importing module objects to the current namespace
 - We may also import the function draw_game directly into the main script's namespace, by using the from command.

```
# game.py
# import the draw module
from draw import draw_game
def main():
    result = play_game()
    draw_game(result)
```

- Importing all objects from a module: import *
 from draw import *
- Custom import name: import xxx as yyy

Module initialization

- The first time a module is loaded into a running Python script, it is initialized by executing the code in the module once.
- If another module in your code imports the same module again, it will not be loaded twice but once only so local variables inside the module act as a "singleton" they are initialized only once.
- Extending module load path
 PYTHONPATH=/foo python game.py sys.path.append("/foo")

- Exploring built-in modules
 - Check out the full list of built-in modules in the Python standard library: https://docs.python.org/3/library/
 - Two important functions to explore modules: dir, help
- Writing packages
 - Packages are namespaces which contain multiple packages and modules themselves.
 - Each package in Python is a directory which MUST contain a special file called <u>__init__.py</u>. This file can be empty, and it indicates that the directory it contains is a Python package, so it can be imported the same way a module can be imported.
 - If we create a directory called foo, which marks the package name, we can then create a module inside that package called bar. We also must not forget to add the init .py file inside the foo directory.

To use the module bar, we can import it in two ways:

import foo.bar

 we must use the foo prefix whenever we access the module bar. from foo import bar

 we don't, because we import the module to our module's namespace.

Exercise

• In this exercise, you will need to print an alphabetically sorted list of all functions in the re module, which contain the word find.

import re

Your code goes here

The End!

Assignment: Practice the example code I ran in this class.