Introduction to Python: Data types

HORT 59000
Lecture 8
Instructor: Kranthi Varala
Why Python?

• Readability and ease-of-maintenance
  • Python focuses on well-structured easy to read code
  • Easier to understand source code…
  • ..hence easier to maintain code base

• Portability
  • Scripting language hence easily portabbale
  • Python interpreter is supported on most modern OS’s

• Extensibility with libraries
  • Large base of third-party libraries that greatly extend functionality. Eg., NumPy, SciPy etc.
Python Interpreter

• The system component of Python is the interpreter.

• The interpreter is independent of your code and is required to execute your code.

• Two major versions of interpreter are currently available:
  • Python 2.7.X (broader support, legacy libraries)
  • Python 3.6.X (newer features, better future support)
Python execution model

- Interpreter has two phases:
  - Source code is compiled into byte code
  - Byte code is executed on the Python Virtual Machine
  - Byte code is regenerated every time source code OR the python version on the machine changes.
  - Byte code generation saves repeated compilation time.
Script vs. command line

- Code can be written in a python script that is interpreted as a block.
- Code can also be entered into the Python command line interface.
  - You can exit the command line with Ctrl-z on windows and Ctrl-d on unix
- For complex projects use an IDE (For example, PyCharm, Jupyter notebook).
  - PyCharm is great for single-developer projects
  - Jupyter is great sharing code and output with markup
First script

- This is the command line interface
- Simply type in the command and the output, if any, is returned to the screen.
- May also be written as a script:
Variables and Objects

- Variables are the basic unit of storage for a program.
- Variables can be created and destroyed.
- At a hardware level, a variable is a reference to a location in memory.
- Programs perform operations on variables and alter or fill in their values.
- Objects are higher level constructs that include one or more variables and the set of operations that work on these variables.
- An object can therefore be considered a more complex variable.
Classes vs. Objects

• Every Object belongs to a certain class.
• Classes are abstract descriptions of the structure and functions of an object.
• Objects are created when an instance of the class is created by the program.
• For example, “Fruit” is a class while an “Apple” is an object.
What is an Object?

• Almost everything is an object in Python, and it belongs to a certain class.

• Python is dynamically and strongly typed:
  • Dynamic: Objects are created dynamically when they are initiated and assigned to a class.
  • Strong: Operations on objects are limited by the type of the object.

• Every variable you create is either a built-in data type object OR a new class you created.
Core data types

- Numbers
- Strings
- Lists
- Dictionaries
- Tuples
- Files
- Sets
Numbers

• Can be integers, decimals (fixed precision), floating points (variable precision), complex numbers etc.
• Simple assignment creates an object of number type such as:
  • a = 3
  • b = 4.56
• Supports simple to complex arithmetic operators.
• Assignment via numeric operator also creates a number object:
  • c = a / b
• a, b and c are numeric objects.
• Try dir(a) and dir(b) . This command lists the functions available for these objects.
Strings

• A string object is a ‘sequence’, i.e., it’s a list of items where each item has a defined position.
• Each character in the string can be referred, retrieved and modified by using its position.
• This order is called the ‘index’ and always starts with 0.

```python
>>> S = 'Hello'
>>> len(S)
5
>>> S[0]
'H'
>>> S[4]
'o'
>>> S[5]
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
IndexError: string index out of range
```
Strings … continued

• String objects support concatenation and repetition operations.

```python
>>> S + 'World!
'HelloWorld!

>>> S + ' World!
'Hello World!

>>> S * 4
'HelloHelloHelloHello'

>>> S + ' World! ' * 4
'Hello World! World! World! World! World! '

>>> (S + ' World! ') * 4
'Hello World! Hello World! Hello World! Hello World! '
```
Lists

- List is a more general sequence object that allows the individual items to be of different types.
- Equivalent to arrays in other languages.
- Lists have no fixed size and can be expanded or contracted as needed.
- Items in list can be retrieved using the index.
- Lists can be nested just like arrays, i.e., you can have a list of lists.
Lists

• Simple list:

```python
>>> L = [123, 3.14, 'Hello']
>>> L
[123, 3.1400000000000001, 'Hello']
```

• Nested list:

```python
>>> DDL = [[[1,2,3]],
         [[4,5,6]],
         [[7,8,9]]]
>>> DDL[2][1]
8
```
Dictionaries

• Dictionaries are unordered mappings of 'Name : Value' associations.
• Comparable to hashes and associative arrays in other languages.
• Intended to approximate how humans remember associations.
Files

- File objects are built for interacting with files on the system.
- Same object used for any file type.
- User has to interpret file content and maintain integrity.

```python
>>> f = open('test.txt', 'w')
>>> f.write('Hello\t')
>>> f.write('world!\n')
>>> f.close()
>>> f = open('test.txt')
>>> text = f.read()
>>> text
'Hello\tworld!\n'
>>> print(text)
Hello   world!
```
Mutable vs. Immutable

• Numbers, strings and tuples are immutable i.e. cannot be directly changed.
• Lists, dictionaries and sets can be changed in place.

```python
>>> S[0]
'H'
>>> S[0] = 'h'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> L[1]
3.1400000000000001
>>> L[1] = 3.145
```
Tuples

- Tuples are immutable lists.
- Maintain integrity of data during program execution.
Sets

• Special data type introduced since Python 2.4 onwards to support mathematical set theory operations.
• Unordered collection of *unique* items.
• Set itself is mutable, BUT every item in the set has to be an immutable type.
• So, sets can have numbers, strings and tuples as items but cannot have lists or dictionaries as items.