Working With Data

Vivek K. S., Deepak G.

Information Systems Decision Sciences (ISDS)
MUMA College of Business
University of South Florida
Tampa, Florida

2017

Introduction

- In Programming, we deal with data day in, day out.
- Most of them concern the built-in Python Data Types, strings and bytes & bytearrays.
- Text strings is the most familiar and commonly used data in programming.
- In Python 3, Strings are Unicode strings and not byte arrays.
- In Python 2, the language had the in-built ability to differentiate between normal byte strings and Unicode character strings.

Unicode and ASCII

- ASCII was defined in the 1960s and only used 7 bits.
- Remember that the basic unit of storage in computers is bytes which is 8 bits each.
- With 8 bits in a byte, we could represent 256 unique values.
- But since ASCII uses only 7 bits, the range came down to 128 unique values.
- Naturally, ASCII could represent only a limited set of characters that includes the 26 upper case and lower case letters, the 10 digits and special characters and non-printing control codes.

Unicode and ASCII

- Due to the mass-adoption of computing technology and the growing demand for language support and need for special symbols, ASCII has since become very obsolete.
- The Unicode is an ongoing international standard to define the characters of all the world's languages, plus symbols from mathematics and other fields.

According to the Unicode consortium -
"Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language."

The Unicode Standard

- The latest version of Unicode Standard (6.2) defines over 110,000 characters, each with a unique name and identification number.
- The characters are divided into eight-bit sets called planes. The first 256 planes are the basic multilingual planes.
- In Python 3, a “u followed by four hex numbers specifies a character in one of Unicode’s 256 multilingual planes.
- The first 2 numbers represents the plane’s number (in the range 00 to FF) and the next 2 numbers represent the index of the character within the plane.
- Note that plane 00 is ASCII.
- For characters in the higher planes, we need more bits. The Python escape sequence for these is “U followed by eight hex characters; the leftmost ones need to be 0.

The Unicode Standard

- The Python unicodedata module has functions that translate in both directions:
- lookup() takes a case-insensitive name and returns a Unicode character.
- name() returns an uppercase name for a unicode character.
- Rest of the coding exercise is available at –

String Formatting

- Python has two ways of formatting strings, loosely called old style and new style.
- Both styles are supported in Python 2 and 3 (new style in Python 2.6 and up).
- The old style of string formatting has the form string
- The following are the conversion types in the old style.
  - %s string
  - %d decimal integer
  - %x hex integer
  - %o octal integer
  - %f decimal float
  - %e exponential float
  - %g decimal or exponential float

Code Examples

Using the Conversion types, we could represent the built-in data types in different ways.

```
To represent an integer:
'%s' % 42 # as string
'42'
'%d' % 42 # as decimal
'42'
'%x' % 42 # as hexadecimal
'2a'
'%o' % 42 # as octal
'52'
```

Code Examples

Floating point numbers can be represented using the conversion types as follows.

```
'%s' % 7.03 # as string
'7.03'
'%f' % 7.03 # as float
'7.030000'
'%e' % 7.03 # as exponential float
'7.030000e+00'
'%g' % 7.03 # as decimal or exponential float
'7.03'
```

String Representations

We use plenty of string formatting in our print statements. Here is a simple example of how it is done.

```
days = 4
language = 'Python'
print('We are here to learn %s for %d days!'
%(language,days))
We are here to learn Python for 4 days!
```

New Style Formatting

Old style formatting is still supported by Python 2. But since we are going to be using Python 3, we will adopt the new style. An example of the new style formatting on the same print statement looks like this.

```
days = 4
language = 'Python'
print('We are here to learn {} for {} days!'
.format(language,days))
We are here to learn Python for 4 days!
```

Regular Expressions

Regular expressions are used for complex pattern matching.

- Regular expressions are patterns used to match character combinations in strings.
- It is a particular kind of formal grammar used to parse strings and other textual information that are known as "Regular Languages" in formal language theory.
- It is common to all programming languages and is available in Python through the re (short for regular expression) module.

```
import re
result = re.match('on', 'Pythonic code')
```

Complete Coding Exercise

Complete coding exercise is available at
`https://github.com/vivek14632/Python-Workshop/tree/master/Introducing%20Python/Chapter%207`

Summary

- We learned the use of Unicode Standard in Python and how it provides versatility to the language's string operations.
- We learned Regular expressions and how pattern matching is done.
- We learned the old and new style of String formatting in Python and its various features.
- We learned encoding and decoding using the Unicode standard.