DATA 133 - Introduction to Data Science I

Instructor: Renzhi Cao
Computer Science Department
Pacific Lutheran University

Announcements

- Quiz #5 on next Tuesday, review everything we covered this week
- Project 1 due date on Sakai.

Reference book

- Data Science from Scratch - First Principles with Python. O'Reilly Media, 2015.

Pictures from: https://www.google.com/search?

Introduction

- Data Science

People are still crazy about Python after twenty-five years, which I find hard to believe.
—Michael Palin

Variables

- In Python, like in other languages, we store values in variables. Unlike other languages, in Python the variables don’t have a “type”
- Use of single quotes ‘’ represents text. No quotes represents numbers

- >>>message = 'Hello'
- >>>print(message)
- >>>message = "Hello"
- >>>print(message)
- >>>message = '"Hello"'
- >>>print(message)

Variables

Rules of naming a variable:

- Don’t start with numbers
- Don’t use @ or -
- Don’t use reserved words

```
and       del       from      not       while
as        elif      global    or        with
assert    else      if        pass      yield
break     except    import    print
class     exec      in        raise
continue  finally   is        return
def       for       lambda    try
```

Practice

Can I use the following variable names?

- 1ab
- ab@a
- aAAA3
- ABDA2
- AND
- for
- For
- a_12A
- b-32D

Try them in the interactive environment and also in Jupyter!

Numbers

- Integer. 10
- Long Integer – an unbounded integer value. 10L
- int(x) converts x to an integer
- float(x) converts x to a floating point
- The interpreter shows a lot of digits

```
>>> 132224
132224
>>> 132323 ** 2
17509376329L
>>> 1.23232
1.2323200000000001
>>> print 1.23232
1.23232
>>> 1.3E7
13000000.0
>>> int(2.0)
2
>>> float(2)
2.0
```

Numbers

- int(10.39)
- int(100.9999)
- int(1001.00001)
- float(87)
- float(eight)

complex

- Built into Python
- Same operations are supported as integer and float

```
>>> x = 3 + 2j
>>> y = -1j
>>> x + y
(3+1j)
>>> x * y
(2-3j)
```

Operators

- Operations in Python are based on sign precedence

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Ccomplement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'td>
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Python2: Integer vs float operations

- Integer operation will result in only the “integer” part of the operation
  - 5/3 equals 1

- Float operation will result in the “float” value of the operation
  - 5/3.0 equals 1.66666667
  - 5.0/3 equals 1.66666667
  - 5.0/3.0 equals 1.666666667

- You can fix that by adding the words:

from __future__ import division

  - At the beginning of your code

- Let’s try it together

Modules

- Certain features of Python are not loaded by default:

from __future__ import division

```
import re
my_regex = re.compile("[0-9]+", re.I)

import re as regex
my_regex = regex.compile("[0-9]+", regex.I)

import matplotlib.pyplot as plt

match = 10
from re import * # uh                    nction
print match # "<function re.match>
```

You may not want to do it

inputs

- The **raw_input**(string) method returns a line of user input as a string
- The parameter is used as a prompt
- The string can be converted by using the conversion methods **int**(string), **float**(string), etc.

Practice

1. Test from Jupyter:
Get a score from user and assign it to variable ‘score’
Convert variable ‘score’ to float
Assign 2 to variable N
print expression: score/N
print expression: int(score) / N
print expression: int(score) / float(N)
print expression: score//N

2. Use any text editor (e.g., Rstudio) to create a python script test.py, and copy the code you tested into the script, run it from command.

Break

String

- Record both textual information (your name as example) and arbitrary collections of bytes (such as image file's contents)

- Strings are sequences of characters.

String

- Strings are *immutable*
- + is overloaded to do concatenation

```
>>> x = 'hello'
>>> x = x + ' there'
>>> x
'hello there'
```

String

- Can use single or double quotes, and three double quotes for a multi-line string

```
>>> 'I am a string'
'I am a string'
>>> "So am I!"
'So am I!'
>>> s = """And me too!
though I am much longer
than the others :)"""
'And me too!\nthough I am much longer\nthan the others :)‘
>>> print s
And me too!
though I am much longer
than the others :)
```

String

```
>fruit = ‘banana’
>letter = fruit[1]
>len(fruit)
>fruit[-1]
>fruit[-2]
```

Traverse a string

```
>for char in fruit:
        print char
>r= fruit[0:2]
```

String

```
>fruit = ‘banana’
>fruit[:]              # all of fruit as a top-level copy (0:len(fruit))


> fruit + ‘xyz’       # Concatenation

> fruit * 8           # Repetition


> fruit[0] = ‘a’      # immutable objects cannot be changed

> new = ‘a’ + fruit[1:]     # this is fine
```

Substring and methods

- **len**(String) – returns the number of characters in the String

- **str**(Object) – returns a String representation of the Object

```
>>> len(x)
6
>>> str(10.3)
'10.3'
```

String methods

```
smiles = "C(=N)(N)N.C(=O)(O)O"
>>> smiles.find("(O)")
15
>>> smiles.find(".")
9
>>> smiles.find(".", 10)
-1
>>> smiles.split(".")
['C(=N)(N)N', 'C(=O)(O)O']
>>>
```

Use “find” to find the start of a substring.

Start looking at position 10.

Find returns -1 if it couldn’t find a match.

Split the string into parts with “.” as the delimiter

String methods

Strings have methods:

```
>word= “banana”
>word.find(‘a’) or word.upper() or word.replace(‘a’,’b’) or word.split(‘,’)

> S = ‘aaa,bbb,ccc, dd\n’
> S.rstrip()        # remove whitespace characters on the right side

>dir(S)             # help
```

String methods

```
if "Br" in “Brother”:
        print "contains brother“

email_address = “clin”
if "@" not in email_address:
        email_address += "@brandeis.edu“
```

String formatting

- Similar to JAVA's printf (%s for string, %d for integer).
- <formatted string> % <elements to insert>
- Can usually just use %s for everything, it will convert the object to its String representation.

```
>>> "One, %d, three" % 2
'One, 2, three'
>>> "%d, two, %s" % (1,3)
'1, two, 3'
>>> "%s two %s" % (1, 'three')
'1 two three'
>>>
```

Practice

Create a script that:

1. Create a string with any characters in total length of 10. (you can manually assign it or asks the user - Raw_input method)
2. Prints the string letter by letter. Each letter in a different line
3. Prints the string in lower case
4. Prints the string in upper case
5. Prints the string backwards
6. Create string with “,” inside, and use split method to process it
7. Prints first three characters
8. Prints last four characters

Cheers! Successful second day!

Practice of R

Finish all in-class exercises and turn it in Sakai
Quiz on next Tuesday.
Read book Page 15 - 19.