Python for ML
Lecture 1: Basics of Python (part1)

Lê Anh Cường - 2020

Content

- Basic statements
- List
- Tuples
- Dictionary
- Functions
- Class

Install Anaconda

- https://www.datacamp.com/community/tutorials/installing-anaconda-windows

- Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

Other option

- Install Python
- Install Jupyter

As an existing Python user, you may wish to install Jupyter using Python's package manager, pip, instead of Anaconda.

First, ensure that you have the latest pip; older versions may have trouble with some dependencies:

```
pip3 install --upgrade pip
```

Then install the Jupyter Notebook using:

```
pip3 install jupyter
```

Install Python

- **Ubuntu 17.10, Ubuntu 18.04** (and above) come with Python 3.6 by default. You should be able to invoke it with the command `python3`.
- **Ubuntu 16.10 and 17.04** do not come with Python 3.6 by default, but it is in the Universe repository. You should be able to install it with the following commands:

Shell

```
$ sudo apt-get update
$ sudo apt-get install python3.6
```

Install Python

- https://phoenixnap.com/kb/how-to-install-python-3-windows
- https://www.python.org/downloads/

Basic statements

- Variables and Types
- Expressions
- Conditional statement
- For
- While
- Input and Output

Variables and Types

- Types:
  - Number: integer and real
  - String
  - List
  - Dictionary
  - A Class

Python does not have a **character** data type, a single **character is** simply a **string with a** length **of** 1

Number

Integers in Python 3 are of unlimited size. Python 2 has two integer types - int and long. There is no '**long integer**' in Python 3 anymore.

```
#number: very big number
x = 12345678910111213123456789101112131234567891011121312345678910111213
y = x**4
print('x =',x)
print('y =',y)
print(type(x))
print(type(y))
```

```
x = 12345678910111213123456789101112131234567891011121312345678910111213
y = 232305723559300478066364749794484442504555159705371666490944890336101807077521003135310776747499017752184427332182293177583173482328879436904119235339561231017378290042730987040018928377340840807508333120339385971111464015796494298597740999527696015027987690469414161
<class 'int'>
<class 'int'>
```

https://www.tutorialspoint.com/python3/python_numbers.htm

Float

```
x = 12345678910111213123.4567891011121312345678910111213123456789101112 13
print(type(x))
y = x**4
print(y)
```

```
<class 'float'>
2.3230572355930005e+76
```

String

https://www.tutorialspoint.com/python3/python_strings.htm

Strings are amongst the most popular types in Python. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double quotes. Creating strings is as simple as assigning a value to a variable.

Python does not support a character type; these are treated as strings of length one, thus also considered a substring.
To access substrings, use the square brackets for slicing along with the index or indices to obtain your substring. For example

```
var1 = 'Hello World!'
var2 = "Python Programming"

print ("var1[0]: ", var1[0])
print ("var2[1:5]: ", var2[1:5])
```

```
var1[0]:  H
var2[1:5]:  ytho
```

String: update value?

```
#String
name = "Le Van A"
name[0]='n'
print(name)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-25-17151c9ff506> in <module>
      1 #String
      2 name = "Le Van A"
----> 3 name[0]='n'
      4 print(name)

TypeError: 'str' object does not support item assignment
```

```
#String
name = 'Le Van A'
name = 'n' + name[1:]
print(name)
```

```
ne Van A
```

If statement

```
# if
if True:
    print('hi')

a,b,c = 3,2,4
if a>b and a>c:
    print('max=',a)
elif b>c and b>a:
    print('max=',a)
else:
    print('max=',c)
```

```
hi
max= 4
```

List

The list is the most versatile datatype available in Python, which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that the items in a list need not be of the same type.

```
list1 = ['physics', 'chemistry', 1997, 2000]
list2 = [1, 2, 3, 4, 5, 6, 7 ]

print ("list1[0]: ", list1[0])
print ("list2[1:5]: ", list2[1:5])
```

```
list1[0]:  physics
list2[1:5]:  [2, 3, 4, 5]
```

```
a = [2,4,6]
a[1]=3
print(a)
del a[2]
print(a)
```

```
[2, 3, 6]
[2, 3]
```

List: Basic List Operations

Lists respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.
In fact, lists respond to all of the general sequence operations we used on strings.

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1,2,3] : print (x,end = ' ')	1 2 3	Iteration

List: Indexing, Slicing and Matrixes

Since lists are sequences, indexing and slicing work the same way for lists as they do for strings.

```
L = ['C++'', 'Java', 'Python']
```

Python Expression	Results	Description
L[2]	'Python'	Offsets start at zero
L[-2]	'Java'	Negative: count from the right
L[1:]	['Java', 'Python']	Slicing fetches sections

List: Built-in List Functions and Methods

Python includes the following list functions

Sr.No.	Function & Description
1	len(list) Gives the total length of the list.
2	max(list) Returns item from the list with max value.
3	min(list) Returns item from the list with min value.
4	list(seq) Converts a tuple into list.

List: Built-in List Functions and Methods

Python includes the following list methods

Sr.No.	Methods
1	list.append(obj) Appends object obj to list
2	list.count(obj) Returns count of how many times obj occurs in list
3	list.extend(seq) Appends the contents of seq to list
4	list.index(obj) Returns the lowest index in list that obj appears
5	list.insert(index, obj) Inserts object obj into list at offset index
6	list.pop(obj = list[-1]) Removes and returns last object or obj from list
7	list.remove(obj) Removes object obj from list
8	list.reverse() Reverses objects of list in place
9	list.sort([func]) Sorts objects of list, use compare func if given

for

```
a=[1,3,5,6,8]
for x in a:
    print(x, end=' ')
print()
for i in range(len(a)):
    print(a[i], end=' ')
print()
for i in range(len(a)-1,-1,-1):
    print(a[i], end=' ')
```

```
1 3 5 6 8
1 3 5 6 8
8 6 5 3 1
```

```
a = [1,2,3,4,5,6]
for i in range(0,len(a),2):
    print(a[i])
```

```
1
3
5
```

while

```
a = [1,2,3,4,5,6]
i=0
while not a[i]==4:
    print(a[i])
    i+=1
```

```
1
2
3
```

```
a = [1,2,3,4,5,6]
i=0
while True:
    if a[i]==4:
        break
    print(a[i])
    i+=1
```

```
1
2
3
```

```
a = [1,2,3,4,5,6]
i=0
while i<len(a):
    if a[i]==4:
        i+=1
        continue
    else:
        print(a[i])
        i+=1
```

```
1
2
3
5
6
```

Tuples

A tuple is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. The main difference between the tuples and the lists is that the tuples cannot be changed unlike lists. Tuples use parentheses, whereas lists use square brackets.

```
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5 )
tup3 = "a", "b", "c", "d"
print(tup1)
print(tup2)
print(tup3)
```

```
('physics', 'chemistry', 1997, 2000)
(1, 2, 3, 4, 5)
('a', 'b', 'c', 'd')
```

```
print(tup1[0])
print(tup2[2:4])
```

```
physics
(3, 4)
```

Tuples: basic operations

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1,2,3) : print (x, end = ' ')	1 2 3	Iteration

Tuples: Indexing, Slicing, and Matrixes

```
T=('C++', 'Java', 'Python')
```

Python Expression	Results	Description
T[2]	'Python'	Offsets start at zero
T[-2]	'Java'	Negative: count from the right
T[1:]	('Java', 'Python')	Slicing fetches sections

Tuples: Built-in Tuple Functions

Sr.No.	Function & Description
1	cmp(tuple1, tuple2) Compares elements of both tuples.
2	len(tuple) Gives the total length of the tuple.
3	max(tuple) Returns item from the tuple with max value.
4	min(tuple) Returns item from the tuple with min value.
5	tuple(seq) Converts a list into tuple.

Exercises

1. Given a list of numbers, the task is to pick even numbers to put into a list and odd numbers into another list.
2. Given a list containing a mix of numbers and strings. Please build two lists containing numbers and strings separately.
3. Give an example of matrix multiplication and display the matrices in rows and columns.