Neural Network and Learning Machines
Lab 1: Introduction to python

Objectives

1. Why python?
2. Introduction to python.

Why python?

- Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language
- Python is friendly language where syntax is simple to read and follow
- It is a fully-functional programming language that can do anything almost any other language can do
- It is a general language that does a little of everything at a good enough complexity-performance tradeoff
- It has a full suite of tools for productionizing machine learning
- There is a python package for most conceivable math functions:
    - Numerical linear algebra: Numpy
    - General scientific computing (e.g., integration, DSP): Scipy
    - Convex optimization: CVXOPT
    - Statistical modeling: Statsmodel, PyMC3
    - Symbolic algebra: SymPy
- Thus, Python can be used to make games, do data analysis, control robot and hardware, create GUIs, or even to create websites.

Environment setup

- You can download any of python version from here
    - https://www.python.org/downloads/windows/
- If you are using different OS than windows then you can download your version from here
    - https://www.python.org/downloads/ according to your OS
- After setting up python environment you can check your installation by running this command on your cmd/terminal

  python –version

  You should see now which python version you have installed in your machine.

Using Python IDE

Usually we use an IDE to write a code with any programming language. Python environment comes with “Python shell” you can use it to run any python command. Also you can use “Python IDLE” to write a complete python program and compile & run it. Examples

- PyCharm
- Spider

Lab 1: Introduction to python

- Jupyter Notebook
- Anaconda

Introduction to python

IO operations

- Hello world
  In : print('hello world')
  Out : hello world
  In : 'hello world'
  Out : 'hello world
- Taking input
  In : x = input("Enter a number")
  print (x) # 5
  Out : 5

  Note: the received data is always treated as string so you should cast it if you need other type.

Data types and Variables

Variables are nothing but reserved memory locations to store values. It means that when you create a variable, you reserve some space in the memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to the variables, you can store integers, decimals or characters in these variables

Example:

```
counter = 100          # An integer assignment
miles   = 1000.0       # A floating point
name    = "John"       # A string
a = b = c = 1
a, b, c = 1, 10.5, "hello"
```

Operators

In : 4/3
Out : 1
In : 4.0/3.0
Out : 1.3333333333333333
In : 4/3.0
Out : 1.3333333333333333

Lab 1: Introduction to python

```
In : int(4.0)/int(3.0)
Out : 1
In : 1+2
Out : 3
In : 2**3
Out : 8
In : 9//2
Out : 4
In : -9//2
Out : -5
In : [1,2,3]+[2,3,4]
Out : [1, 2, 3, 2, 3, 4]
In :  type(1),type(2.0),type([3,4]), type(True)
Out : (<class 'int'>, <class 'float'>, <class 'list'>,<class 'bool'>)
```

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, 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'
^ |	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

Decision making

Example:

```
x = input("Enter the value of X \n")

y = input("Enter the value of Y \n")

if ( x > y):

   print ("X is bigger than y")

else:

   print ("X is not bigger than y")
```

Lines and indentation

Python does not use braces({}) to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount.

Example

```
if True:
   print ("True")
else:
   print ("False")
```

Looping

In python you can use while loops or for loops. Here's the syntax for while loop

While EXPRESSION:

BODY OF THE LOOP

And here's the syntax for "for loops"

for iterating_var in sequence:

BODY OF THE LOOP

Example for printing a numbers from 1 to 5:

```
# with while loop

x = 1

while(x <= 5):

   print (x)

   x = x + 1

# with for loops

for x in range(5):

   print (x+1)
```

range(n) generates an iterator to progress integers starting with 0 up to n-1.

You can use **break** statement to exit any loop or **continue** statement to skip one iteration in the loop

Functions

```
In : def power(x,y):
         z = x**y
         return z
In : power(2,5)
Out : 32
In : g = lambda x,y:  x**y
In : g(2,5)
Out : 32
In [15]: (lambda x,y: (x**y))(2,5)
Out : 32
```

Dictionaries

```
In : x = {
         'stud1':'20140001',
         'stud2':'20140002',
       }
In : x['stud1']
Out : '20140001'
```

Lab 1: Introduction to python

Let's try a decision tree for classifying fruits

```
In : def classify(x):
        decision = ''
        if x['color'] == 'green':
          if x['size'] == 'big':
            decision = 'watermelon'
          elif x['size'] =='medium':
            decision = 'apple'
          else:
            decision = 'other'
        else:
          decision = 'other'
        return decision
In : classify({'color':'green','size':'big'})
Out : 'watermelon'
In : classify({'color':'green','size':'medium'})
Out : 'apple'
In : classify({'color':'red','size':'small'})
Out : 'other'
```

Iterators

```
In : data = [
            {'color':'green','size':'big'},
            {'color':'yellow','shape':'round','size':'big'},
            {'color':'red','size':'medium'},
            {'color':'green','size':'big'},
            {'color':'red','size':'small','taste':'sour'},
            {'color':'green','size':'small'}
            ]
In : results = []
   for x in data:
            results = results + [classify(x)]
   results
Out : ['watermelon', 'other', 'other', 'watermelon', 'other', 'other']

In : [classify(x) for x in data]
Out : ['watermelon', 'other', 'other', 'watermelon', 'other', 'other']
In : list(map(classify,data))list
Out : ['watermelon', 'other', 'other', 'watermelon', 'other', 'other']
```

Let's try to count watermelon in data

```
In : result = map(classify,data)
        count = 0
        for r in result:
            if r == 'watermelon':
                count = count + 1
        print(count)
Out: 2
In [27]: sum([classify(x)=='watermelon' for x in data])
Out[27]: 2
```

Or similarly
In [28]: len(list(filter(lambda x: classify(x)==’watermelon’, data)))
Out[28]: 2

Reading data from file

Example

```
filename = "hello.txt"

file = open(filename, "r")

for line in file:

    print (line)
```

- File methods
  - open(fname , mode), close()
  - read(),readline() and readlines()
  - write() ,writelines()
  - Append ()

References

- https://machinelearningmastery.com/machine-learning-in-python-step-by-step/
- https://github.com/StefanCobeli/Python-for-Machine-Learning
- https://www.tutorialspoint.com/python/python_basic_operators.htm