Python Programming for Machine Learning

Parcours Progis

Etudes, Medias, communication, Marketing

Bahareh Afshinpour.

07.11.2024

Université Grenoble Alpes

Sciences Po Grenoble UGA

DATA TYPES

- A type is how Python represents different types of data.
  - Integers, real numbers, string and boolean
- In Python, a string is a sequence of characters.
  - A string can be spaces or digits. A string can also be special characters.
- You can change the type of the expression in Python, this is called typecasting.

```
# Integer

11
```

```
# Float

2.14
```

```
# String

"Hello, Python 101!"
```

VARIABLES

- We can use variables to store values.
- We assign a value to a variable using the assignment operator, i.e, the equal sign.
- We can then use the value somewhere else in the code by typing the exact name of the variable.
- We can use the type command in variables as well.

PYTHON CONDITIONS AND IF STATEMENTS

Python supports the usual logical conditions from mathematics:

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

```
b is greater than a
```

An "if statement" is written by using the if keyword.

IF EXAMPLE- CRÉER UN MENU POUR LE PETIT-DÉJEUNER

```
print("1-Crêpes")
print("2-Céréales")
print("3-Gaufres")
M=int(input("quel voulez-vous au petit déjeuner?"))
if(M==1):
    meal="Crêpe"
elif(M==2):
    meal="Céréale"
elif(M==3):
    meal="Gaufre"

C_type=0
if (meal=="Crêpe"):
    print("1- Crêpe nature")
    print("2- Crepe au nutella")
    print("3- Crepe au sucre")
    C_type=int(input(" Choisissez un type de Crêpes"))

if (C_type==1):
    t=" nature"
elif(C_type==2):
    t=" au nutella"
elif(C_type==3):
    t=" au sucre"
else:
    t=""

print("Vous avez choisi "+meal+t+" pour petit déjeuner" )
```

```
1-Crêpes
2-Céréales
3-Gaufres
quel voulez-vous au petit déjeuner?
```

```
1-Crêpes
2-Céréales
3-Gaufres
quel voulez-vous au petit déjeuner? 2
Vous avez choisi Céréale pour petit déjeuner
```

```
1-Crêpes
2-Céréales
3-Gaufres
quel voulez-vous au petit déjeuner? 1
1- Crêpe nature
2- Crepe au nutella
3- Crepe au sucre
 Choisissez un type de Crêpes 3
Vous avez choisi Crêpe au sucre pour petit déjeuner
```

NESTED IF STATEMENT

The syntax of the nested **`if construct`** with else condition will be like this

```
if boolean_expression1:
   statement(s)
   if boolean_expression2:
      statement(s)
```

```
num=int(input("Please, enter one number"))
print ("num = ",num)
if (num%2==0):
    if num%3==0:
        print ("Divisible by 3 and 2")
    else:
        print ("divisible by 2 not divisible by 3")
else:
    if num%3==0:
        print ("divisible by 3 not divisible by 2")
    else:
        print ("not Divisible by 2 not divisible by 3")
```

```
Please, enter one number 8
num =  8
divisible by 2 not divisible by 3
```

```
Please, enter one number 9
num =  9
divisible by 3 not divisible by 2
```

```
Please, enter one number 11
num =  11
not Divisible by 2 not divisible by 3
```

LISTS

Index	0	1	2	3	4
List Data	David	4.12	6	**[3,9]**	657

[David,4.12,6,[3,9],657]

- Lists are a popular data structure in Python.
- Each box has a numerical reference called an index that is used to refer to the individual data item.
- A list is represented with square brackets.
- Lists can contain strings, floats, integers. Also, we can nest other lists.
- Note that in Python the first element of the list shown here has an index of zero.

LIST OPERATIONS

- Lists are mutable; can be changed in-place
- Lists are dynamic; size may be changed

```
>>> r = [1, 2.0, 3, 5]
>>> r[3] = 'word'              # replace an item by index
>>> r
[1, 2.0, 3, 'word']
```

```
>>> len(r)                     # length of list; number of items
4
```

```
>>> 3 in r                     # membership test
True
```

LIST METHODS, PART 1

- Lists have a set of built-in methods
- Some methods change the list in-place

```
>>> r = [1, 2.0, 3, 5]
>>> r.append('thing')              # add a single item to the end
>>> r
[1, 2.0, 3, 5, 'thing']
>>> r.append(['another', 'list'])  # list treated as a single item
>>> r
[1, 2.0, 3, 5, 'thing', ['another', 'list']]
```

```
>>> r = [2, 5, -1, 0, 20]
>>> r.sort()
>>> r
[-1, 0, 2, 5, 20]
```

```
>>> s = 'a few words'
>>> w = s.split()                # splits at white-space (blank, newline)
>>> w
['a', 'few', 'words']
```

EXAMPLE

- Create a list, add a new element to it, and then sort it.

```
r=[]
r=[10,5,2,7]
print(r)
```

```
[10, 5, 2, 7]
```

```
r.append(34)
print(r)
```

```
[10, 5, 2, 7, 34]
```

```
r.sort()
print(r)
```

```
[2, 5, 7, 10, 34]
```

EFFECTUER DES TÂCHES RÉPÉTITIVES (UNE BOUCLE)

- A for loop is used for iterating over a sequence.
- The for loop does not require an indexing variable to set beforehand.

[1]:
```
LetterNum=1
for letter in "Bonjour":
    print("la lettre", LetterNum ,"est", letter)
    LetterNum+=1
```

```
la lettre 1 est B
la lettre 2 est o
la lettre 3 est n
la lettre 4 est j
la lettre 5 est o
la lettre 6 est u
la lettre 7 est r
```

LOADING PYTHON LIBRARIES

In [ ]:

```
#Import Python Libraries
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib as mpl
import seaborn as sns
```

Press `Shift+Enter` to execute the *jupyter* cell

Reading data using pandas

```
#Read csv file
df = pd.read_csv("http://rcs.bu.edu/examples/python/data_analysis/Salaries.csv")
```

Note: The above command has many optional arguments to fine-tune the data import process.

There is a number of pandas commands to read other data formats:

```
pd.read_excel('myfile.xlsx',sheet_name='Sheet1', index_col=None, na_values=['NA'])
pd.read_stata('myfile.dta')
pd.read_sas('myfile.sas7bdat')
pd.read_hdf('myfile.h5','df')
```

END