Python Programming Basics

Testing Data Types in Python

Contents

Data Types in Python

Numeric - Float	Contains decimal numbers
Numeric - Integer	Contains only whole numbers
Character	Holds character strings
Categorical	A vector that can contain only predefined values, and is used to store categorical data.
Logical	Can only take on two values, TRUE or FALSE.

Data Types in Python

Vector	1 dimension	Sequence of data elements of the same basic type.
Matrix	2 dimensions	Like a Vector but additionally contains the dimension attribute.
Array	2 or more dimensions	Hold multidimensional data. Matrices are a special case of two-dimensional arrays.
Data frame	2 dimensions	Table-like data object allowing different data types for different columns.
List	Collection of data objects, each element of a list is a data object.	

Numeric - Float

Numeric type object can store decimal as well as whole numbers.

Create two numeric objects x and y and assign values 4.5 and 3567 respectively.

```
x = 4.5
x
```

```
4.5
```

```
y = 3567
y
```

```
3567
```

```
type(x)
```

type() function is used to check the data type of an object.

```
float
```

```
type(y)
```

```
int
```

Numeric – Integer

Python shows type of object x as float. To convert it to an integer format, use **`int()`** function.

```
x =int(x)
x
```

```
 4
```

So, new value of **x** is **4** after converting it into integer

```
type(x)
```

```
int
```

Numeric – Integer

#To create an integer variable in Python use **int()** function.

```
f = int(22.5)
f
```

```
22
```

```
type(f)
```

```
int
```

```
x=8
type(x)
```

```
int
```

Note : The default class of an integer is an integer

Character

A character object is used to represent strings. A string is specified by using quotes (both single and double quotes will work). To convert any object into a character type, use **`str()`** function.

```
z = "Welcome to Python  Ready Reckon-er"
z
```

```
'Welcome to Python  Ready Reckon-er'
```

```
x = "4.5"
x
```

```
'4.5'
```

Any value enclosed in quotes is stored as a character object

```
type(z)
```

```
str
```

```
type(x)
```

```
str
```

Categorical

Factor objects are used to categorize data and can store both strings and integers.

```
# Create an object x
```

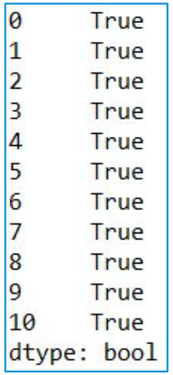
```
import pandas as pd
x = pd.Series(["high", "medium", "low", "low", "medium", "high",
"high", "high", "medium", "low","low"])
```

pd.Series() creates an one-dimensional ndarray.

```
# Check whether object x i
```

```
x.str.isalpha()
```

isalpha() function returns True or False after checking whether the object is of type character or not



```
0      True
1      True
2      True
3      True
4      True
5      True
6      True
7      True
8      True
9      True
10     True
dtype: bool
```

Categorical

```
#Create a categorical object using Categorical() function
```

```
x = pd.Categorical(x)
x.dtype
```

```
CategoricalDtype(categories=['high', 'low', 'medium'], ordered=False)
```

- A Categorical is a categorical variable that can take only one of a fixed, finite set of possibilities. Those possible categories are the levels.
- `unique` are the unique data values. Output below tells how many levels are there in x.
- Alternatively, one can use `unique()` function to check levels.

```
x.unique()
```

```
[high, medium, low]
Categories (3, object):
```

categorical object **x** has 3 levels. By default the levels are **sorted chronologically**

Categorical

```
#To specify the order of the factor
```

```
x = pd.Categorical(x,categories=['low', 'medium','high'])
x
```

```
[high, medium, low, low, medium, ..., high, high, medium, low, low]
Length: 11
Categories (3, object): [low, medium, high]
```

categories= takes the levels in the order in which you want them to be ordered

*

One of the most important uses of factors is in statistical modeling, since categorical variables enter into statistical models differently than continuous variables, storing data as factors insures that the modeling functions will treat such data correctly.

Logical

Logical type objects can only take 2 values i.e. TRUE or FALSE.

```
# Create an object x and assign a value 4.5 and check whether it is an integer
x = 4.5
isinstance(x,int)
```

```
False
```

isinstance() returns true if the object argument is an instance of the classinfo argument

```
# Create two numeric objects y and z
# Check whether y is greater than z or not
y = 4
z = 7
Result = y > z
Result
```

```
False
```

- ❑ With this kind of statement, you are asking Python to **evaluate the logical question** “Is it true that y is greater than z?”
- ❑ The object(Result) storing the answer of above question is of type logical (bool standing for Boolean)
- ❑ You can check the class of the object using **type()**

Vector

A Vector is a Sequence of data elements of the same basic type.
Create three vectors: numeric, character and logical. All the elements of a vector must be of the same type.

```
# Numeric vector
```

```
a = [1,2,5.3,6,-2,4]
a
```

```
[1, 2, 5.3, 6, -2, 4]
```

```
# Character vector
```

```
b = ["one","two","three"]
b
```

```
['one', 'two', 'three']
```

```
# Logical vector
```

```
import numpy as np
d = np.array([4,24,6,4, 2,7])
d > 5
```

```
array([False,  True,  True, False, False,  True])
```

np.array() creates an array with one or multiple dimensions

Matrices

Matrix is a two-dimensional data structure in Python. It similar to vectors but additionally contains the dimension attribute. To convert any object into a matrix type, use `np.array()` function.

```
# Create a matrix with 3 rows and 2 columns.
```

```
X = np.array([2, 3, 4, 5, 6, 7]).reshape(3,2)
X
```

```
array([[2, 3],
       [4, 5],
       [6, 7]])
```

- ❑ **np.array()** function is used to create a matrix (multi-dimensional array).
- ❑ **reshape(rows,cols)** is used to specify the dimensions of the matrix

Note that the matrix is filled in by row-wise in Python.

Matrices

It is possible to name the rows and columns of matrix during creation

```
import pandas as pd
x = pd.DataFrame(np.array([2, 3, 4, 5, 6, 7]).reshape(3,2),
index=["X","Y","Z"], columns=["A","B"])
x
```

```
   A  B
X  2  3
Y  4  5
Z  6  7
```

```
#Dimension names can be accessed or changed with three helpful
functions columns, index, rename()
```

```
x.columns
Index(['A', 'B'], dtype='object’)
x.index
Index(['X', 'Y', 'Z'], dtype='object’)
x.rename(columns = {'A':'a', 'B':'b'}, inplace = True)
x.columns
Index(['a', 'b'], dtype='object')
```

rownames can be changed in similar manner

Arrays

- An array holds multidimensional rectangular data i.e. each row is the same length, and likewise for each column and other dimensions

```
# Create an array with four rows, two columns and three “tables”.
```

```
a = np.arange(1,25).reshape(3,4,2)
a
```

```
array([[[ 1,  2],
        [ 3,  4],
        [ 5,  6],
        [ 7,  8]],

       [[ 9, 10],
        [11, 12],
        [13, 14],
        [15, 16]],

       [[17, 18],
        [19, 20],
        [21, 22],
        [23, 24]]])
```

np.arrange() is one of the array creation routines based on numerical ranges. It creates an instance of ndarray with evenly spaced values and returns the reference to it.
reshape (t,r,c)
t = no. of tables
r = no. of rows
c = no. of columns

Data Frames

Data frame is a two-dimensional array like structure. It is a list of vectors of equal length. Data frames are the primary data structure in Python. To convert any object into a data frame type, use **`pd.DataFrame()`** function.

Create a data frame from different vectors

```
data  = {'x':[12,23,45],'y':[13,21,6],'z':["a","b","c"]}
df = pd.DataFrame(data)
df
```

creating vectors x, y, z

```
    x   y  z
0  12  13  a
1  23  21  b
2  45   6  c
```

- ❑ **DataFrame()** function combines them in a table.
- ❑ object **data** is a dataframe containing three vectors **x, y, z**

* DataFrames are distinct from matrices because they can include heterogeneous data types among columns/variables.

Data Frames

Let's have a look at the structure of our data frame.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
x     3 non-null int64
y     3 non-null int64
z     3 non-null object
dtypes: int64(2), object(1)
memory usage: 152.0+ bytes
```

- **info()** shows the structure of an object
- **<u>Note</u>** : **z** is a character vector but by default Python stores it in the data frame as object

Lists

List is a mutable, ordered sequence of elements. Each element inside a list is called items. Lists are defined by having values between square brackets. To convert any object into a list type, use **`list()`** function.

Create a list of 2 vectors - a numeric & other character

```
n=[2, 3, 5]
s=["aa", "bb", "cc", "dd", "ee"]
x=list(n+s)
x
```

list() is used to create lists,

```
[2, 3, 5, 'aa', 'bb', 'cc', 'dd', 'ee']
```

```
m = [2, 4, 6, 1, 3, 5]
p = ['even', 'odd']
y=list(m+p)
y
```

```
[2, 4, 6, 1, 3, 5, 'even', 'odd']
```

Quick Recap

In this session, we discussed about various data types of Python and how to create them. Here is a quick recap:

Data types	1. Numeric 2. Integer 3. Character 4. Categorical 5. Logical 6. Vector 7. Matrices 8. Array 9. Data Frame 10. List
Type Conversion Functions	`int(), str(), pd.DataFrame(), list(), np.array()`