Python: Introduction and Data Types

Kunal Raghav

2018-08-31

Introduction

Python is an interpreter based language, i.e. a pytho program is evaluated line by line. Python's interpreter can be accessed by typing `python` , `python3` or the variable name which is specified in your environment variables on the `terminal` / `command prompt`.

Typing Python

- In python statements don't require a `;` to end a statement, a statement in python is written in a single line.
  - A line break `enter keypress` ends the current statemnt.
- Comments are typed using # before the sentence.
- Statements in python don't require to be enclosed in braces to show if they're a part of a code block.
  - Python uses indentation to show they're in the same code block.
  - Python language hence looks cleaner to first time coders.

```
a=10          # statement 1
while a>0:    # code block begins
  print(a)
  print("These lines are in")
  print("the same code block.")
  print("We use a tab space to indent code.")
  a-=1
print("Now we're outside the code block.")
```

Variables

Variables are used to store data and do operations. Python being a friendly language doesn't require much effort to declare variables.

Rules

- All variables name should begin with a letter.
- Variable names are case sensitive.
- Variable are given data types automatically according to their data unless explicitly declared.

- Variable names shouldn't have names similar to inbuilt functions such as `for` , `in` , `range` , `input` and other **reserved words**.

Example Declaration

```
a=4          # int
b="hello"
c='hello'    # strings can be declared using both '/" symbols
d= 4.2       # float
e= int(3.6) # gives 3 because we're explicitly converting a
             # float to int.
l=[]         # list
l1=list()    # also list
dic={}       # dictionary
dic1=dict() # also dictionary
t=()         # tuple
t1=tuple()   # also tuple
t2=1,2       # also tuple
a1,b2 = 3,4 # gives a1=3 and b2=4
```

Data Types

Integer

This data type is used to store integers, such as `1`, `2`, `3`, `-4`, etc.

- If the given integer value crosses higher than the limit of the integer it is automatically converted to `long`.

Float

This data type is used to store decimals such as `3.14159265`, etc.

- If the given decimal is higher than the float limits the variable automatically changes it's data type to `double`.

String

This data type is used to store strings such as `'despacito'`, `'arre bhai bhai bhai'`, `'Sacred Games'`, etc.

List

A list data type is an `array` or list of various other data, which may or may not be of the same data type.

- List is mutable i.e. it's value can be modified without changing it's location in the memory.
- Lists can also be nested inside one another.
- Lists do not have a pre determined size, hence can be expanded limitlessly.

Manipulation

Declaration

```
lis=[]              # empty list
lis=list()          # also empty list
lis=[1,2,3,4]       # list of integers
lis=[1,2,4.3]       # list of integers and floats
lis=[1,"string",[2.3,"nestedList"]]     # nested lists
```

Adding an Element

- An element is added in a list using the `append()` function.

```
emptyList = []
for i in range(5):
    emptyList.append(i)
print(emptyList)        # prints [0,1,2,3,4]
```

Accessing/Modifying an Element

An element inside a list is accessed using index numbers which goes from `0` to `n-1` where `n` is the total number of elements.

- Nested list items are accessed by adding another index number after the index number of the `nestedList` is specified.

```
a=["string",3.4,["Kunal",11]]

# Printing 3.4

print(a[1])

# 3.4 is the 2nd element in the list hence has the index number of 1

# Printing 11

print(a[2][1])

# 11 is the 2nd element of nested list at 3rd position

# Modifying "string" to "CHANGE"

a[0] = "CHANGE"   # a ["CHANGE",3.4,["Kunal",11]]

# Modifying "Kunal" to "Raghav"

a[2][0] = "Raghav"
```

Deleting an Element

An element once inside a list can be removed using `pop` and `del` functions.

- `del` function deletes the value at the specified index number.

```
a=["this","is","list","of","strings"]

#delete "is"

del a[1]    # a=["this","list","of","strings"]
```

- `pop` pops the "value" from the list from the last positon or from the index number specified and returns it.

```
a=["this","is","list","of","strings"]

#pop "strings"

a.pop()    # a=["this","is","list","of"]
```

```
6
7 #pop "this"
8
9 b=a.pop(0)   # a=["is","list","of"] AND b="this"
```

Tuple

A tuple is similar to a list but is immutable i.e. no elements can be added, removed or modified without redeclaring the tuple.

Declaration

A tuple can be declared using `()` , , and `tuple()`.

```
1 a=()      # empty tuple
2 b=tuple() # also empty tuple
3 c=1,2,3  # is same as c=(1,2,3)
```

Usage

- As tuples can't be modified they're converted to lists explicitly to perform operations and coverted back to tuples to prevent accidental modification.

```
1 a = (1,2,3,4)
2
3 # adding an element 5 at the end of a
4
5 a = list(a)      # gives [1,2,3,4]
6 a.append(5)     # gives [1,2,3,4,5]
7 a = tuple(a)     # gives (1,2,3,4,5)
```

Dictionary

A dictionary is special type of list where there are no index numbers, instead we use `keys` to refer to the values stored in a dictionary.

- A typical dictionary looks like:

```
{"key1":"value1","key2":"value2"}
```

  - `"key1"` and `"key2"` are `keys` used to refer to `"value1"` and `"value2"`.

- A dictionary like lists supports nesting.

Manipulation

Declaration

A dictionary can be declared using `{}` and `dict()`.

```
dic={}              # empty dictionary
dic=dict()          # also empty dictionary

# dictionary containing various data types
dic={"keyOne":1, "key2":"value2"}

dic={1:"string",2:{2.1:"nestedDict"}}    # nested dictionary
```

Adding & Modifying Elements

An element can be added or modified in a dictionary by using the following syntax:

```
dict={}

#Adding an Element

dict["ThisIsKey"] = "SomeValue"
print(dict)          # prints {"ThisIsKey":"SomeValue"}

#Modifying "SomeValue" to "SomeOtherValue"
dict["ThisIsKey"] = "SomeOtherValue"
print(dict)          # prints {"ThisIsKey":"SomeOtherValue"}
```

Deleting Elements

An element in a dictionary can be deleted using `del` and `pop` functions.

```
dic={"ThisKey":"Value","ThatKey":"ThatValue","OtherKey":"OtherValue"}

```

```
# Deleting an element using del.
del dic["ThisKey"]
print(dic)
#  prints {"ThatKey":"ThatValue","OtherKey":"OtherValue"}

# Deleting an element using pop.

b = dic.pop("ThatKey") # b= "ThatValue"

print(dic)
#prints {"OtherKey":"OtherValue"}
```

- For a dictionary the pop function requires a key, because there is no concept of last element in a dictionary.