Python for JNTUK Students

Sreekanth Kolamala

M.tech Computer Science, IIT, Kanpur

Types

Python has 5 data types-Numbers, Strings, Lists, Tuple, Dictionary. In this chapter, we will discuss Numbers and Strings. We will see the other 3 in the next chapter.

Numbers

Numbers are further divided into 4-int, long, float, complex.

```
a = 12
print type(a)
b = 12.3
print type(b)
c = 2323231211142342424
print type(c)
d = complex(2,3)
print d
print type(d)
```

```
## <type 'int'>
## <type 'float'>
## <type 'long'>
## (2+3j)
## <type 'complex'>
```

Strings

Python doesn't differete between single quotes and double quotes. Unlike in C, the is not difference between char ans string-both are same.

```
name = "sreekanth"
occupation = 'software engineer'
print type(name)
qoute = "I'am good" #observe that a single can be used within doubles
print "qoute:" + qoute
qoute = "qoute:" + 'He said "I am not going" to her'
print qoute
```

```
## <type 'str'>
## qoute:I'am good
## qoute:He said "I am not going" to her
```

```
qoute = "He said "I am not going" to her"
#this will not work.
```

Lists

lists are similar to C arrays but more powerful. Lists store multiple values. unlike C arrays, elements in a python list don't have be of the same type. lists are indexed from 0.

```
fruits = ['apple', 'mango', 'pineapple']
numbers = [10, 20, 30, 40, 50]
#elements in a list can be accessed using index.
print numbers[1]
#even negative index is allowed, in which case length of list is added
#to the index.
print numbers[-1]
fruitsandnumbers =  ['apple', 'mango', 'pineapple', 10, 20, 30, True]
#mixing elements of different types is fine
print fruitsandnumbers
print len(numbers)
#len is a built-in function used to find the length of lists.
```

```
## 20
## 50
## ['apple', 'mango', 'pineapple', 10, 20, 30, True]
## 5
```

Slicing

Slicing is simple way of creating a new list from (part of) an old list.

```
newList = oldList[start:end]
```

```
numbers = [10, 20, 30, 40, 50, 60, 70]
sublist = numbers[1:4]
print sublist
print numbers[0:3]
print numbers[:3] #missing starting number, with replaced with 0
print numbers[4:7]
print numbers[4:] #missing ending number is replaced with length of the list.
#negative indices can also be given.
#In which case, the length of the list is added
print numbers[-2:7]
print numbers[:-1]  #this will print all elements except the last one.
```

```
## [20, 30, 40]
## [10, 20, 30]
## [10, 20, 30]
## [50, 60, 70]
## [50, 60, 70]
## [60, 70]
## [10, 20, 30, 40, 50, 60]
```

As shown in the example, sublist is a new list consisting of elements 1 to 3(remember, 'end' is not inclusive) in the numbers list. you can also specify negative numbers for indexes in which case, the length of list is automatically added. For example `numbers[-2:7]` is equivalent to `numbers[5:7]`.

List methods

Lists are not fixed in length. They increase and decrease when elements are added and removed.

```
numbers = [10,20,30,40,50]
numbers.append(60)  #adds 60 at the end of the list
print numbers
numbers.insert(2, 25) #you can even specify an index
print numbers
#there isn't a method to remove element from an index but you
#use slicing
numbers = numbers[:1] + numbers[2:]
print numbers #this will remove element at index 1.
print numbers.pop() #removes and returns the last element in the list
print numbers
# More methods
print numbers.index(40) #returns the index of the element 20 if present
numbers.sort()  #sorts list
print numbers
```

```
## [10, 20, 30, 40, 50, 60]
## [10, 20, 25, 30, 40, 50, 60]
## [10, 25, 30, 40, 50, 60]
## 60
## [10, 25, 30, 40, 50]
## 3
## [10, 25, 30, 40, 50]
```

Sets

Sets are like lists but duplicates are not allowed.

```
presentees = {1,5,6, 7, 10, 1, 6, 6, 5}
print presentees
presentees.add(10)
print presentees
```

```
## set([1, 10, 5, 6, 7])
## set([1, 10, 5, 6, 7])
```

Strings as lists

In python, strings are lists. So, all the list operations can be done on strings.

```
name = "sreekanth"
print name[0]
print name[1:5] #slicing
print name + "kolamala" # concatination
#
```

```
## s
## reek
## sreekanthkolamala
```

```
a = 12
def add(a,b):
  return a + b
print add(2,3)
```

```
## 5
```

```
System.out.println("hi");
int a = 10;
int hex = 0x7FFFFFFF; //the max int
```

```
var a = 10;
console.log(a)
```

R Markdown

```
> print("hello world")
```

[1] "hello world"

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

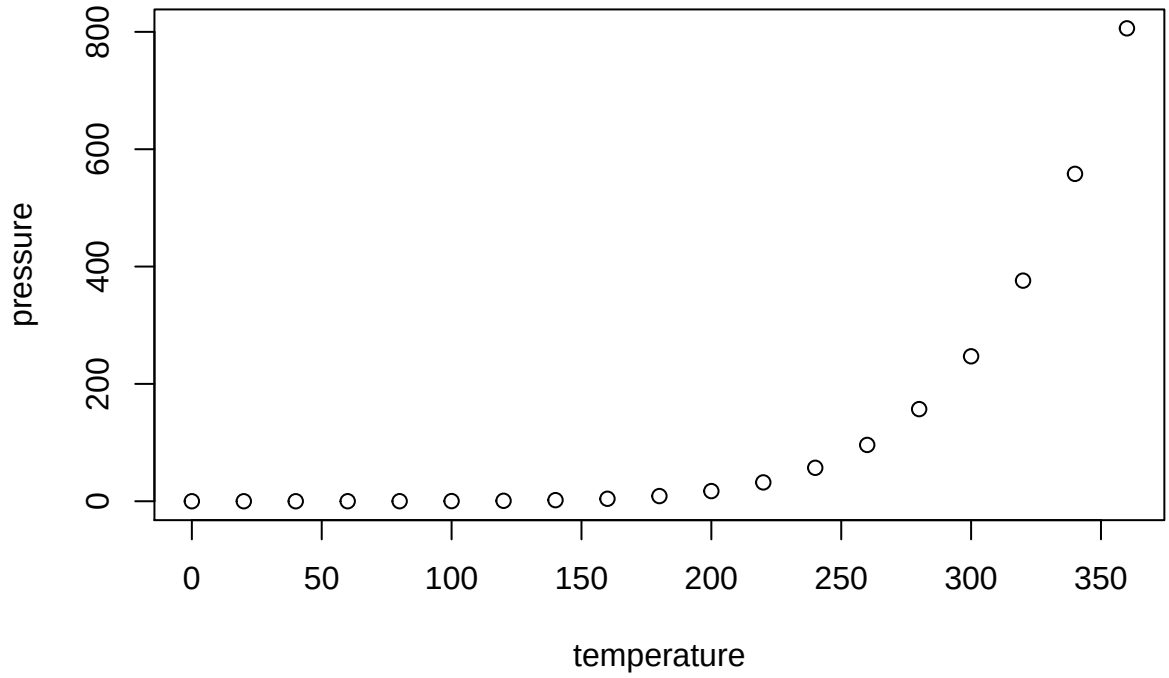
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

https://stackoverflow.com/questions/36808263/separate-columns-for-text-and-code-output-in-markdown?noredirect=1&lq=1