04_04_file_strings_dict

Unknown Author

April 1, 2014

Part I

Data I/O, Strings, Dictionary

1 Data I/O

In this lecture we shall concentrate on a few more programming tools which might be useful to you later. Let us try to write a simple interpreter today. In the process we shall learn how to write to a file and recall reading from a file.

1.1 Reading data from the screen

This you already do using `input`. Some of you already use `raw_input`, though I did not do it in class. Today I'll formally introduce it. Raw input directly stores the input in a string. You can later process the string in any way you want.

In [1]:
```
s = raw_input("Input anything : ")
print type(s)
print s
slst = s.split()
varname = slst[0]
varvalue = float(slst[1])
print "%s has value %f." % (varname, varvalue)
```

```
Input anything : a 7
<type 'str'>
a 7
a has value 7.000000.
```

1.2 Command line input

Sometimes one runs a program where the input to the program is passed via the command line. It is easy to do that in python. For that one has to import a package called `sys`.

In [2]:
```
program = \
"""
import sys

x = float(sys.argv[1])
print x*x
"""
```

```
progfile = open("files/myprogfile.py", 'w')
progfile.write(program)
progfile.close()
```

To see that it works, we can print the file

In [3]:
```
progfile = open("files/myprogfile.py", 'r')
print progfile.read()
progfile.close()
```

```
import sys

x = float(sys.argv[1])
print x*x
```

In [4]:
```
#%run "files/myprogfile.py"
```

In [5]:
```
%run "files/myprogfile.py" 1.5
```

```
2.25
```

1.3 Raising errors

Till now we had been trying to deal with errors using conditionals. However to make the program more readable, one is encouraged to use the `try..except` mechanism provided in python.

In [6]:
```
xstr = raw_input("Test :")
x = float(xstr)
```

```
Test :34.3
```

In [7]:
```
def read_number() :
    xstr = raw_input("Input a number : ")
    try :
        x = float(xstr)
    except ValueError :
        raise ValueError('Cannot understand the number %s.' % xstr)
    return x
```

In [8]:
```
try :
    x = read_number()
except ValueError, e :
    print e
    sys.exit(1)

print x*x
```

```
Input a number : 23.4
547.56
```

2 Strings

Some examples of what you can do with strings.

In [9]:
```
eg_str = "This is a sentence. This is another sentence.\nThis is the second line. This
which started in the second line but\nended in the fourth line."
print eg_str
```

```
This is a sentence. This is another sentence.
This is the second line. This is the second
sentence which started in the second line but
ended in the fourth line.
```

In [10]:
```
# Substrings
eg_str[10:20]
```
Out [10]:
```
'sentence. '
```
In [11]:
```
print "The word 'sentence' begins at %d, whereas the word 'sentience' \
starts at %d." % (eg_str.find('sentence'), eg_str.find('sentience'))
```
```
The word 'sentence' begins at 10, whereas the word 'sentience' starts
at -1.
```
In [12]:
```
# Also in works
print ('sentence' in eg_str)
print ('starts' in eg_str)
```
```
True
False
```
In [13]:
```
print 'health'.startswith('heal')
print 'Python'.endswith('tail')
```
```
True
False
```
In [14]:
```
# Replace
print eg_str.replace('sentence', 'verdict')
```
```
This is a verdict. This is another verdict.
This is the second line. This is the second
verdict which started in the second line but
ended in the fourth line.
```
In [15]:
```
print eg_str.split()
print eg_str.splitlines()
print eg_str.split('a')
print eg_str.split('.')
for str in eg_str.split('.') :
    print str
```
```
['This', 'is', 'a', 'sentence.', 'This', 'is', 'another', 'sentence.',
'This', 'is', 'the', 'second', 'line.', 'This', 'is', 'the', 'second',
'sentence', 'which', 'started', 'in', 'the', 'second', 'line', 'but',
'ended', 'in', 'the', 'fourth', 'line.']
['This is a sentence. This is another sentence.', 'This is the second
line. This is the second', 'sentence which started in the second line
but', 'ended in the fourth line.']
['This is ', ' sentence. This is ', 'nother sentence.\nThis is the
second line. This is the second\nsentence which st', 'rted in the
second line but\nended in the fourth line.']
['This is a sentence', ' This is another sentence', '\nThis is the
second line', ' This is the second\nsentence which started in the
second line but\nended in the fourth line', '']
This is a sentence
 This is another sentence

This is the second line
 This is the second
sentence which started in the second line but
ended in the fourth line
```
In [16]:
```
# Checking type of characters in a string
print "'2334' contains only digits : %s" % '2334'.isdigit()
print "'a123' contains only digits : %s" % 'a123'.isdigit()
print "Space '    \n  \t  ' : %s" % '  \n  \t   '.isspace()
print "'' is a space : %s" % ''.isspace()
```

```
'2334' contains only digits : True
'a123' contains only digits : False
Space '
          ' : True
'' is a space : False
```

In [17]:
```
# Removing initial and trailing characters
print "+" + '    This is a   stupid sentence.          \n'.strip() + "+"
```
```
+This is a   stupid sentence.+
```

In [18]:
```
# delimiter.join(list of strings)

list_of_sentences = ["Sky is blue", "Classes are boring", "Examples are stupid"]
print '. '.join(list_of_sentences) + '.'
print "-"*50
print ".\n".join(list_of_sentences) + '.'
```
```
Sky is blue. Classes are boring. Examples are stupid.
--------------------------------------------------
Sky is blue.
Classes are boring.
Examples are stupid.
```

3 Dictionary

In [19]:
```
names = ['Eric', 'Ila', 'Emma', 'John', 'Umesh', 'Asha', 'Akash', 'Kate', 'Uma', 'Sam'
scores = [7,8,6,9,10,6,8,7,7,9]

score_dict={'Eric' : 7, 'Ila' : 8}
print "Ila scored %d." % score_dict['Ila']

# One can add.
score_dict={}
print score_dict

for i in range(len(names)) :
    score_dict[names[i]] = scores[i]

print score_dict
```
```
Ila scored 8.
{}
{'Emma': 6, 'Akash': 8, 'Sam': 9, 'Ila': 8, 'Asha': 6, 'Kate': 7,
'Umesh': 10, 'Uma': 7, 'John': 9, 'Eric': 7}
```

In [20]:
```
for name in score_dict :
    print "%s scored %d" % (name, score_dict[name])
```
```
Emma scored 6
Akash scored 8
Sam scored 9
Ila scored 8
Asha scored 6
Kate scored 7
Umesh scored 10
Uma scored 7
John scored 9
Eric scored 7
```

In [21]:
```
def print_score(n) :
    if n in score_dict :
        print "%s scored %d." %(n, score_dict[n])
    else :
        print "%s is not on list." % n
```

In [22]:
```
print_score('Peter')
print_score('Asha')
```

```
Peter is not on list.
Asha scored 6.
```

In [23]:
```
def tabulate_scores(sd) :
    print "Name     :    Score"
    for name in sorted(sd) :
        print "%5s    :%9d" % (name, sd[name])
    return None
```

In [24]:
```
tabulate_scores(score_dict)
```

```
Name     :    Score
Akash    :        8
 Asha    :        6
 Emma    :        6
 Eric    :        7
  Ila    :        8
 John    :        9
 Kate    :        7
  Sam    :        9
  Uma    :        7
Umesh    :       10
```