



Data Processing Using Python

Data Retrieval and Represent

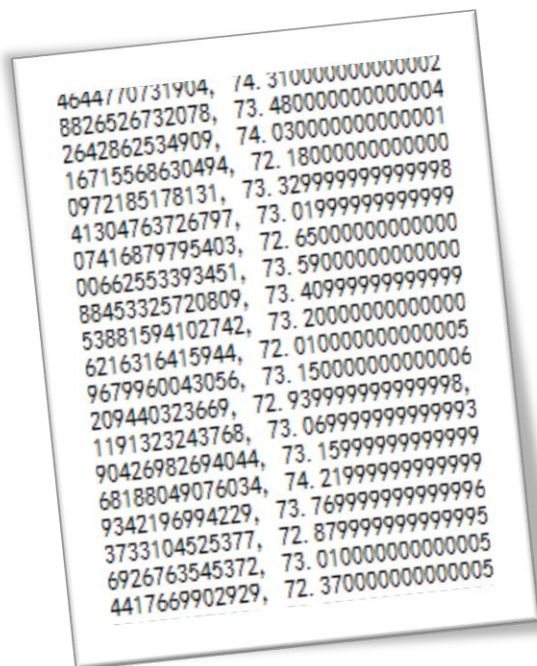
ZHANG Dazhuang

Department of Computer Science and Technology

Department of University Basic Computer Teaching

Data Processing Using Python

1 LOCAL DATA RETRIEVAL



How to get local data?

Open, read/write and close of file.

- Read/write after open.

- Read files

Write files

- Why files need to be closed

Open File

4



```
>>> f1 = open('d:\\infile.txt')  
>>> f2 = open(r'd:\outfile.txt', 'w')  
>>> f3 = open('record.dat', 'wb', 0)
```

file_obj = open(filename, mode='r', buffering=-1, ...)

- **mode** is an optional parameter with default value 'r'
- **buffering** is an optional integer used to set the buffering policy. Pass 0 to switch buffering off (only allowed in binary mode), 1 to select line buffering (only usable in text mode), and an integer > 1 to indicate the size in bytes of a fixed-size chunk buffer.

open() -mode

5

Mode	Function
r	Open for reading (default)
w	Open for writing, truncating the file first
a	Open for writing, appending to the end of the file if it exists.
x	Open for exclusive creation, failing if the file already exists
b	binary mode
+	open a disk file for updating (reading and writing)
t	text mode (default)

Return Value

- `open()` returns a **file object**
- File object is **iterative**
- There exists **functions/methods** to read/write/close files.
 - `f.read()`, `f.write()`, `f.readline()`, `f.readlines()`, `f.writelines()`
 - `f.close()`
 - `f.seek()`

Write a File-f.write()

- **file_obj.write(str)**
 - Write a string into file

 Source

```
>>> f = open('firstpro.txt', 'w')
>>> f.write('Hello, World!')
>>> f.close()
```

```
firstpro.txt :
Hello, World!
```

 Source

```
>>> with open('firstpro.txt', 'w') as f:
        f.write('Hello, World!')
```

Read a File-f.read()

- **file_obj.read(size)**
 - Read at most size **byte** of data from file, return a string.
- **file_obj.read()**
 - Read file till the end, return a string



```
>>> with open('firstpro.txt') as f:  
    p1 = f.read(5)  
    p2 = f.read()  
    print(p1,p2)
```

Output:
Hello, World!

Other Read/Write Functions



```
# Filename: companies_a.py
with open('companies.txt') as f:
    cNames = f.readlines()
print(cNames)
```

- file_obj.readlines()
- file_obj.readline()
- file_obj.writelines()

Output:

['GOOGLE Inc.\n', 'Microsoft Corporation\n', 'Apple Inc.\n', 'Facebook, Inc.']

Example



Add sequence number 1, 2, 3, ... to the strings in file companies.txt, and write into another file scompanies.txt.



```
# Filename: revcopy.py
with open('companies.txt') as f1:
    cNames = f1.readlines()
for i in range(0, len(cNames)):
    cNames[i] = str(i+1) + ' ' + cNames[i]
with open('scompanies.txt', 'w') as f2:
    f2.writelines(cNames)
```

Output:

```
1 GOOGLE Inc.
2 Microsoft Corporation
3 Apple Inc.
4 Facebook, Inc.
```

Other File Related Functions

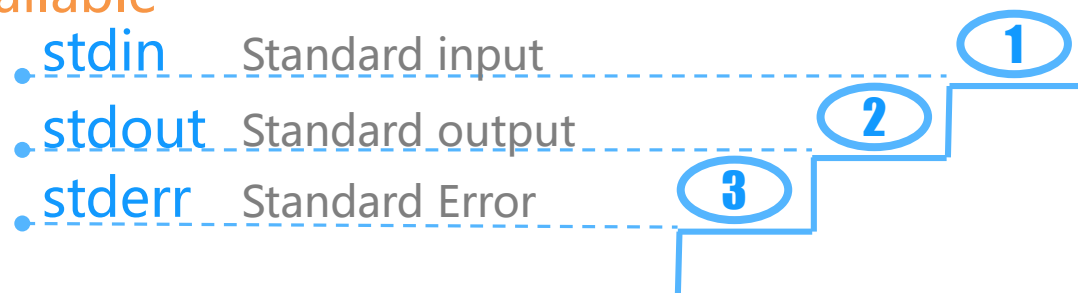


```
# Filename: companies_b.py
s = 'Tencent Technology Company Limited'
with open('companies.txt', 'a+') as f:
    f.writelines('\n')
    f.writelines(s)
    cNames = f.readlines()
print(cNames)
```

- **file_obj.seek(offset , whence=0)**
 - Set the file pointer in file, with **offset** bytes of alignment from *whence* (an optimal parameter with default value 0. 0 stands for the beginning of file, 1 means current position, 2 means the end).

Standard File

- When a program begins, the following three files are available



```
>>> newcName = input('Enter the name of new company: ')
```

```
Enter the name of new company: Alibiabia
```

```
>>> print(newcName)
```

```
Alibiabia
```

```
>>> import sys
>>> sys.stdout.write('hello')
```

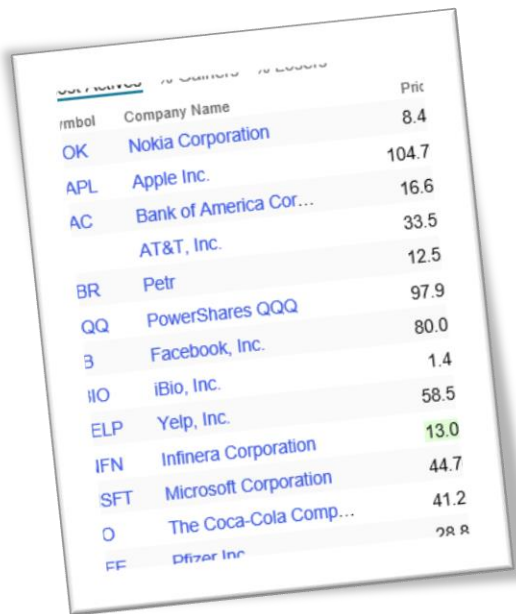
Data Processing with Python

2

INTERNET DATA RETRIVAL

Data Retrieval with Python

14



Symbol	Company Name	Price
OK	Nokia Corporation	8.4
APL	Apple Inc.	104.7
AC	Bank of America Cor...	16.6
	AT&T, Inc.	33.5
BR	Petr	12.5
QQ	PowerShares QQQ	97.9
B	Facebook, Inc.	80.0
IIO	iBio, Inc.	1.4
ELP	Yelp, Inc.	58.5
IFN	Infinera Corporation	13.0
SFT	Microsoft Corporation	44.7
O	The Coca-Cola Comp...	41.2
FF	Pfizer Inc	28.8

How to get data on the Internet?

Crawl webpage, and interpret the content.

- Crawling
 - **Urllib** built-in module
 - urllib.request
 - **Requests**
(third party library)
 - **Scrapy** framework
- Interpreting
 - **BeautifulSoup** library
 - **re** module

Third party
crawling and
interpreting

Requests Library

- Requests library is a simple, easy and user-friendly Python HTTP third party library.
- Requests Official Site: <http://www.python-requests.org/>
- Basic method

```
requests.get()
```

request resource at given URL ,
corresponding to GET in HTTP.

```
Respect the crawling protocol robots.txt
```

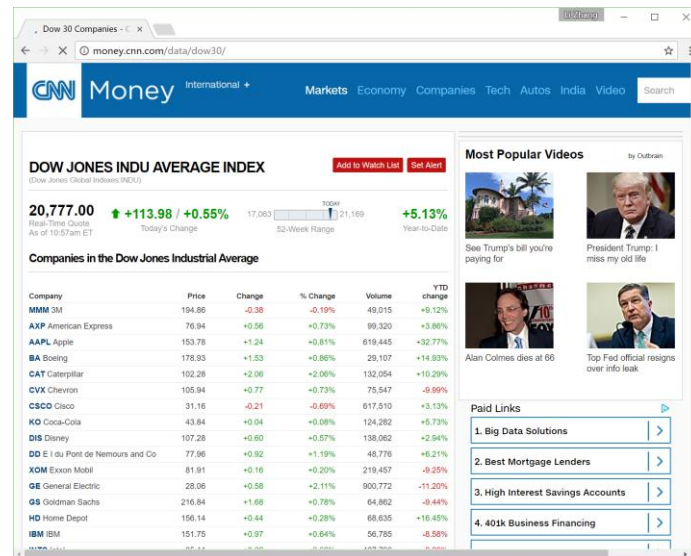
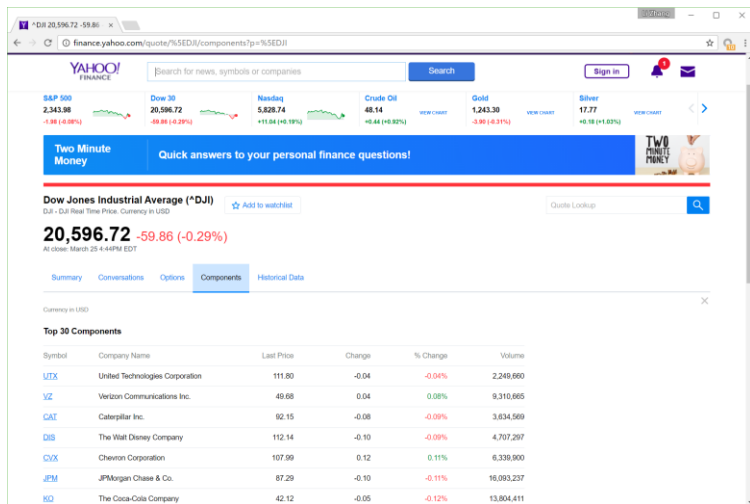


```
>>> import requests
>>> r = requests.get('https://book.douban.com/subject/1084336/comments/')
>>> r.status_code
200
>>> print(r.text)
```

Dow Jones Constituent

16

<http://finance.yahoo.com/q/cp?s=%5EDJI+Component>



<http://money.cnn.com/data/dow30/>

Get Doe Jones Constituent with Requests¹⁷

```
<tr>
  <td class="wsod_firstCol"><a href="/quote/quote.html?symb=INTC" class="ws
  <td class="wsod_aRight"><span stream="last_167459" class="wsod_stream">35
  <td class="wsod_aRight"><span stream="change_167459" class="wsod_stream">
  <td class="wsod_aRight"><span stream="changePct_167459" class="wsod_strea
  <td class="wsod_aRight">17,171,872</td>
  <td class="wsod_aRight"><span class="negData">-3.39%</span></td>
</tr>

<tr>
  <td class="wsod_firstCol"><a href="/quote/quote.html?symb=JNJ" class="wso
  <td class="wsod_aRight"><span stream="last_174239" class="wsod_stream">12
  <td class="wsod_aRight"><span stream="change_174239" class="wsod_stream">
  <td class="wsod_aRight"><span stream="changePct_174239" class="wsod_strea
  <td class="wsod_aRight">6,571,254</td>
  <td class="wsod_aRight"><span class="posData">+10.21%</span></td>
</tr>
```

- Including multiple strings
 - 'AXP', 'American Express Company', '77.77'
 - 'BA', 'The Boeing Company', '177.83'
 - 'CAT', 'Caterpillar Inc.', '96.39'
 - ...

A blue speech bubble containing the word "File" in a stylized font.

Filename: dji.py

import requests

re = requests.get('http://money.cnn.com/data/dow30/') # the url may change

print(re.text)

Interpreting Webpages

- **BeautifulSoup** is a Python library which helps extract data from HTML or XML files.
- Official Website:
<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>



```
soup.find_all('p', 'comment-content')
```

`<p class="comment-content">`不知道第几次重读。每过一段时间再读，都有新的收获。心变得很柔软，脑里的迷雾被驱散。更多的关注他人，关心这个世界，自私是多么无趣的事情啊。我想，写一本能温暖人心，帮助困难的人们书，比世界上很多事情都有意义。`</p>`

- **re** regular expression module
- Reference:

<https://docs.python.org/3.5/library/re.html>

```
'<span class="user-stars allstar(.*) rating'
```

```
<span class="user-stars  
allstar50 rating" title="力荐  
></span>
```

Data Processing Using Python

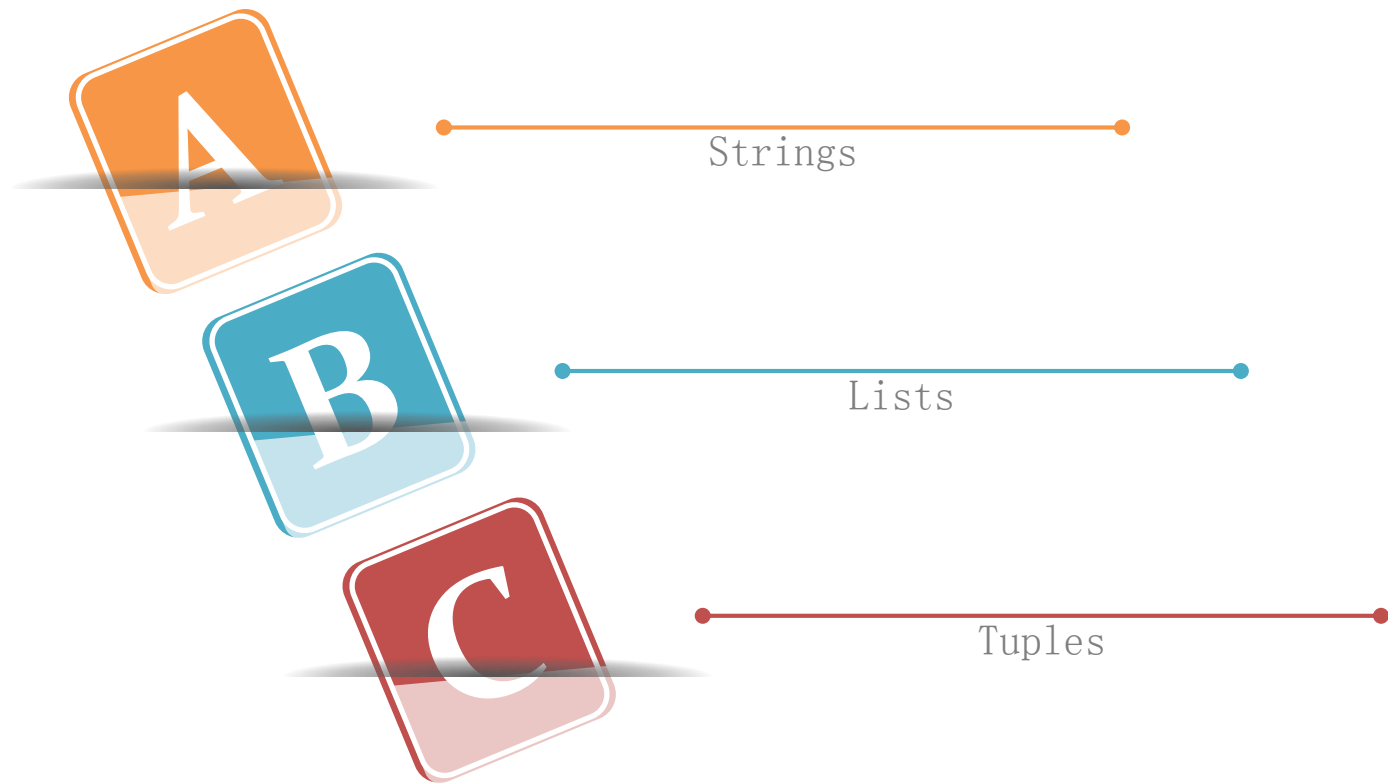
3

SEQUENCE

Sequence

20

- aStr = 'Hello, World!'
- aList = [2, 3, 5, 7, 11]
- aTuple = ('Sunday', 'happy')
- pList = [('AXP', 'American Express Company', '78.51'),
('BA', 'The Boeing Company', '184.76'),
('CAT', 'Caterpillar Inc.', '96.39'),
('CSCO', 'Cisco Systems, Inc.', '33.71'),
('CVX', 'Chevron Corporation', '106.09')]

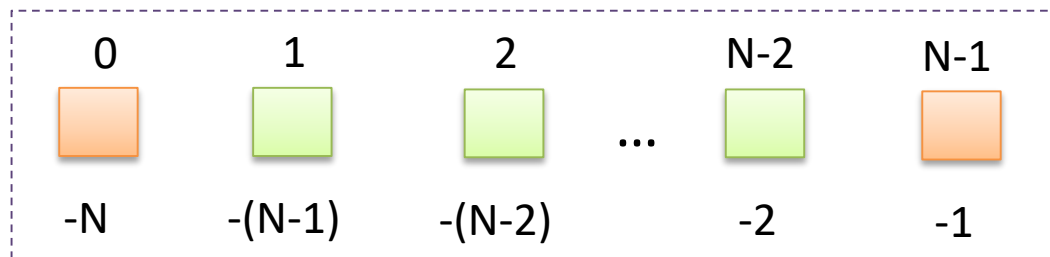


Sequence in Python

22

week	0	1	2	3	4	5	6
	'Monday'	'Tuesday'	'Wednesday'	'Thursday'	'Friday'	'Saturday'	'Sunday'
	-7	-6	-5	-4	-3	-2	-1

Sequence



Visit mode

- Elements are visited by index offset from 0.
- One or multiple elements can be visited at one time

Sequence-Related Function

standard operator

Value comparison
Object identity
Comparison
Boolean operation

Sequence operator

Get(seq[index])
Repeat(seq*expr)
Connect(seq1+seq2)
Judge(obj **in** seq)

Built-in Function

Sequence type conversion
Available function for
sequence type(enumerate,
reversed, sorted, zip, ...)

Standard Operator



```
>>> 'apple' < 'banana'
```

```
True
```

```
>>> [1,3,5] != [2,4,6]
```

```
True
```

```
>>> aTuple = ('BA', 'The Boeing Company', '184.76')
```

```
>>> bTuple = aTuple
```

```
>>> bTuple is not aTuple
```

```
False
```

```
>>> ('86.40' < '122.64') and ('apple' > 'banana')
```

```
False
```


Standard Operator

Value Comparison

<	>
<=	>=
==	!=

Object identity Comparison

is
is not

Boolean operation

not
and
or

Sequence Operator

S
ource

```
>>> week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
>>> print(week[1], week[-2], '\n', week[1:4], '\n', week[:6], '\n', week[::-1])
Tuesday Saturday
['Tuesday', 'Wednesday', 'Thursday']
['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
['Sunday', 'Saturday', 'Friday', 'Thursday', 'Wednesday', 'Tuesday', 'Monday']
>>> 'apple' * 3
'appleappleapple'
>>> 'pine' + 'apple'
'pineapple'
>>> 'BA' in ('BA', 'The Boeing Company', '184.76')
True
```

Sequence Operator

x in s

x not in s

s + t

s * n, n * s

s[i]

s[i:j]

s[i:j:k]

Sequence Type Conversion

28

list()

str()

tuple()



```
>>> list('Hello, World!')
['H', 'e', 'l', 'l', 'o', ',', ' ', 'W', 'o', 'r', 'l', 'd', '!']
>>> tuple("Hello, World!")
('H', 'e', 'l', 'l', 'o', ',', ' ', 'W', 'o', 'r', 'l', 'd', '!')
```

Available Functions for Sequence

29

enumerate()	reversed()
len()	sorted()
max()	sum()
min()	zip()



```
>>> aStr = 'Hello, World!'
```

```
>>> len(aStr)
```

```
13
```

```
>>> sorted(aStr)
```

```
[' ', '!', ',', 'H', 'W', 'd', 'e', 'l', 'l', 'l', 'o', 'o', 'r']
```

Data Processing Using Python

4

STRING

Different Formats of String

```
lf = [('AXP', 'American Express Company', '78.51'),  
      ('BA', 'The Boeing Company', '184.76'),  
      ('CAT', 'Caterpillar Inc.', '96.39'),  
      ('CSCO', 'Cisco Systems, Inc.', '33.71'),  
      ('CVX', 'Chevron Corporation', '106.09')]
```

A speech bubble icon containing the word "Source" in orange text.

```
>>> aStr = 'The Boeing Company'  
>>> bStr = "The Boeing Company "  
>>> cStr = "I'm a student."  
>>> dStr = '''The Boeing  
company'''
```

Example



Replace "World" in "Hello, World!" with "Python" , and compute the number of punctuation marks.

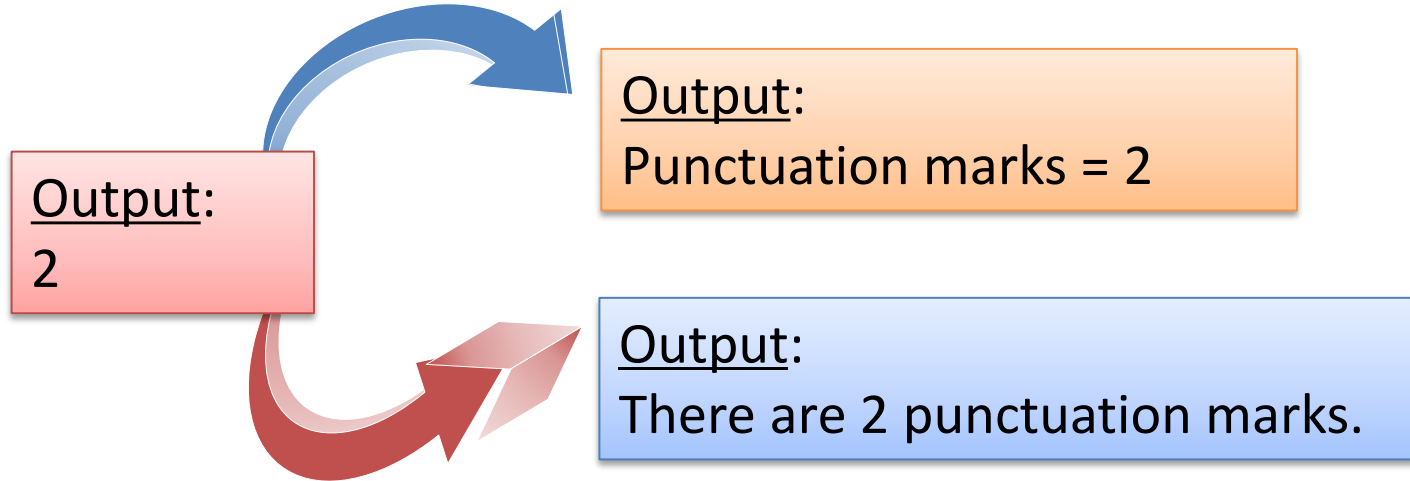


```
# Filename: puncount.py
aStr = "Hello, World!"
bStr = aStr[:7] + "Python!"
count = 0
for ch in bStr[:]:
    if ch in ',.!?':
        count += 1
print(count)
```

Output:

2

String and Output Format



```
print('There are %d punctuation marks. ' % (count))  
format_string % (arguments_to_convert)  
print('There are {0:d} punctuation marks. '.format(count))  
format_string.format(arguments_to_convert)
```

Type Specifier

34

Type	Meaning
b	Binary format. Outputs the number in base 2
o	Octal format. Outputs the number in base 8
x	Hex format. Outputs the number in base 16, using lower- case letters for the digits above 9 (upper-case if use 'X')
c	Character. Converts the integer to the corresponding unicode character before printing.
d	Decimal Integer. Outputs the number in base 10.
f	Fixed point. Displays the number as a fixed-point number. The default precision is 6.
e	Exponent notation. Prints the number in scientific notation using the letter 'e' to indicate the exponent. The default precision is 6.

Other Available Format

符号	描述
+m.nf	Output number with sign, keep n digits, and total length is m (if the number is longer than m, then neglect the constraint)
<	Forces the field to be left-aligned, default filling the right with spaces
0>5d	Forces the field to be right-aligned, use 0 to fill left part, total length is 5
^	Forces the field to be centered within the available space.
{{}}	Output {}

[Alignment][Sign][Minimum width][.Precision][Type]

```
>>> age, height = 21, 1.758
>>> print("Age:{0:<5d}, Height:{1:5.2f}".format(age, height))
Age:21    , Height: 1.76
```

Use format() to Output Formatted String

Source

```
>>> cCode = ['AXP' , 'BA' , 'CAT' , 'CSCO' , 'CVX' ]
>>> cPrice = ['78.51' , '184.76' , '96.39' , '33.71' , '106.09' ]
>>> for i in range(5):
    print('{:<8d}{{:8s}}{{:8s}'.format(i, cCode[i], cPrice[i]))
0    AXP    78.51
1    BA     184.76
2    CAT     96.39
3    CSCO    33.71
4    CVX    106.09
>>> print('I get {:d}{{{}}}'.format(32))
I get 32 {}!
```

String Application



Determine whether string "acdhdca" is a palindrome, and whether 354435 is a palindrome.

File

```
# Filename: compare.py
sStr = "acdhdca"
if (sStr == ''.join(reversed(sStr))):
    print('Yes')
else:
    print('No')
```

File

```
# Filename: compare.py
import operator
sStr = "acdhdca"
if operator.eq(sStr, ''.join(reversed(sStr)))==0:
    print('Yes')
else:
    print('No')
```

sStr == sStr[::-1]

Useful Methods for String

38

capitalize()	center()	count()	encode()	endswith()	find()
format()	index()	isalnum()	isalpha()	isdigit()	islower()
isspace()	istitle()	isupper()	join()	ljust()	lower()
lstrip()	maketrans()	partition()	replace()	rfind()	rindex()
rjust()	rpartition()	rstrip()	split()	splitlines()	startswith()
strip()	swapcase()	title()	translate()	upper()	zfill()

Application of String



There are some downloaded contents with following format:
What do you think of this saying "No pain, No gain"?
For content between double quotes, first determine whether it corresponds with title format, and convert the string into title format then output.



F_{ile}

Filename: totitle.py

aStr = 'What do you think of this saying "No pain, No gain"'

index = aStr.index("\\", 0, len(aStr))

rindex = aStr.rindex("\\", 0, len(aStr))

tempStr = aStr[index+1:rindex]

if tempStr.istitle():

 print('It is title format.')

else:

 print('It is not title format.')

print(tempStr.title())

tempstr= aStr.split("\\")[1]

Escape Character

Character	Meaning
\0	Empty Character
\a	ASCII Bell (BEL)
\b	ASCII Backspace (BS)
\t	ASCII Horizontal Tab (TAB)
\n	ASCII Linefeed (LF)
\v	ASCII Vertical Tab (VT)
\f	ASCII Formfeed (FF)
\r	ASCII Carriage Return (CR)
\"	Double quote (")
\'	Single quote (')
\\	Backslash (\)
\\(在行尾时)	Backslash and newline ignored

\ooo Character with octal value ooo

\xXX Character with hex value XX



```
>>> aStr = '\101\t\x41\n'
```

```
>>> bStr = '\141\t\x61\n'
```

```
>>> print(aStr, bStr)
```

A

A

a

a

Data Processing Using Python

5

LIST

scalable
container
object



```
>>> aList = list('Hello.')
>>> aList
['H', 'e', 'l', 'l', 'o', '.']
>>> aList = list('hello.')
>>> aList
['h', 'e', 'l', 'l', 'o', '.']
>>> aList[0] = 'H'
>>> aList
['H', 'e', 'l', 'l', 'o', '.']
```

Contain
different
types of
objects



```
>>> bList = [1, 2, 'a', 3.5]
```

Format of List

- `aList = [1, 2, 3, 4, 5]`
- `names = ['Zhao', 'Qian', 'Sun', 'Li']`
- `bList = [3, 2, 1, 'Action']`
- `pList = [('AXP', 'American Express Company', '78.51'),
('BA', 'The Boeing Company', '184.76'),
('CAT', 'Caterpillar Inc.', '96.39'),
('CSCO', 'Cisco Systems, Inc.', '33.71'),
('CVX', 'Chevron Corporation', '106.09')]`



One school holds a competition, the rate of each singer is decided by 10 judges and audience. The rule of rating is to remove the highest and lowest rating of 10 judges, and average with the rate of audience. Judges: 9、 9、 8.5、 10、 7、 8、 8、 9、 8 and 10, Audience: 9
Compute the final result.



```
# Filename: scoring.py
jScores = [9, 9, 8.5, 10, 7, 8, 8, 9, 8, 10]
aScore = 9
jScores.sort()
jScores.pop()
jScores.pop(0)
jScores.append(aScore)
aveScore = sum(jScores)/len(jScores)
print(aveScore)
```

```
[7, 8, 8, 8, 8.5, 9, 9, 9, 10, 10]
[8, 8, 8, 8.5, 9, 9, 9, 10]
[8, 8, 8, 8.5, 9, 9, 9, 10, 9]
8.722222222222
```



Merge weekday list (['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']) with weekend (['Saturday', 'Sunday']) add sequence numbers and display the result.



Filename: week.py

```
week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
weekend = ['Saturday', 'Sunday']
week.extend(weekend)
for i,j in enumerate(week):
    print(i+1, j)
```

Output:

```
1 Monday
2 Tuesday
3 Wednesday
4 Thursday
5 Friday
6 Saturday
7 Sunday
```

List Methods

append()
copy()
count()
extend()
index()
insert()
pop()
remove()
reverse()
sort()

Parameters

list.sort(key=None, reverse=False)



```
>>> numList = [3, 11, 5, 8, 16, 1]
```

```
>>> fruitList = ['apple', 'banana', 'pear', 'lemon', 'avocado']
```

```
>>> numList.sort(reverse = True)
```

```
>>> numList
```

```
[16, 11, 8, 5, 3, 1]
```

```
>>> fruitList.sort(key = len)
```

```
>>> fruitList
```

```
['pear', 'apple', 'lemon', 'banana', 'avocado']
```

List Comprehension

List comprehensions,
list comps

Dynamically create list
Easy, flexible and useful

Generator expression

```
>>> sum(x for x in range(10))  
45  
lazy evaluation
```

S_{ource}

```
>>> [x for x in range(10)]  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
>>> [x ** 2 for x in range(10)]  
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]  
>>> [x ** 2 for x in range(10) if x ** 2 < 50]  
[0, 1, 4, 9, 16, 25, 36, 49]  
>>> [(x+1, y+1) for x in range(2) for y in range(2)]  
[(1, 1), (1, 2), (2, 1), (2, 2)]
```

```
[ expression for expr in sequence1  
    for expr2 in sequence2 ...  
    for exprN in sequenceN  
    if condition ]
```

Data Processing Using Python

6

TUPLE

Tuple

- Basic operations of tuple are similar to list.



```
>>> 2014
2014
>>> 2014,
(2014,)
```



```
>>> bTuple = (['Monday', 1], 2, 3)
>>> bTuple
(['Monday', 1], 2, 3)
>>> bTuple[0][1]
1
>>> len(bTuple)
3
>>> bTuple[1:]
(2, 3)
```

- List element is variable
- Tuple element is not variable



```
>>> aList = ['AXP', 'BA', 'CAT']
>>> aTuple = ('AXP', 'BA', 'CAT')
>>> aList[1] = 'Alibiabia'
>>> print(aList)
['AXP', 'Alibiabia', 'CAT']
>>> aTuple[1]= 'Alibiabia'
>>> print(aTuple)
aTuple[1]='Alibiabia'
TypeError: 'tuple' object does not support item assignment
```

- Type of function



```
>>> aList = [3, 5, 2, 4]
>>> aList
[3, 5, 2, 4]
>>> sorted(aList)
[2, 3, 4, 5]
>>> aList
[3, 5, 2, 4]
>>> aList.sort()
>>> aList
[2, 3, 4, 5]
```



```
>>> aTuple = (3, 5, 2, 4)
>>> sorted(aTuple)
[2, 3, 4, 5]
>>> aTuple
(3, 5, 2, 4)
>>> aTuple.sort()
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'tuple' object has no attribute 'sort'

Application of Tuple

Where to use?



Variable Length Position Parameter (Tuple)

53

Parameter type in Python function:

- Position or keyword parameter
- Only position parameter
- Variable Length Position Parameter
- Variable length keyword parameter with default value



```
>>> def foo(args1, args2 = 'World!'):
    print(args1, args2)
>>> foo('Hello,')
Hello, World!
>>> foo('Hello,', args2 = 'Python!')
Hello, Python!
>>> foo(args2 = 'Apple!', args1 = 'Hello,')
Hello, Apple!
```

```
>>> def foo(args1, *argst):
    print(args1)
    print(argst)
```

Variable Length Position Parameter (Tuple)

54



```
>>> def foo(args1, *argst):  
    print(args1)  
    print(argst)  
>>> foo('Hello,', 'Wangdachui', 'Niuyun', 'Linling')  
Hello,  
( 'Wangdachui', 'Niuyun', 'Linling')
```

Tuple as a Return Type

Number of return value(s)	Return Type
0	None
1	object
>1	tuple



```
>>> def foo():  
        return 1, 2, 3  
>>> foo()  
(1, 2, 3)
```