SI 100B Tutorial:
Intro. to HW1 - Flight Analytics

Longtian Qiu
from the SI 100B staff
https://si100b.org/staff

Where to Get the PA?

- The course homepage: https://si100b.org/. Click link in the syllabus section;
- Direct access: http://gitlab.q71998.cn/homework-fall2020/homework1;
- GitLab: http://gitlab.q71998.cn/homework-fall2020/homework1

Policy on Academic Integrity

“We enforce academic integrity strictly. If you participate in any form of cheating, you will receive punishment. Do not try to hack the GitLab grading system in any way. You can view the full version of the code of conduct on the course homepage: https://si100b.org/resource-policy/#policies.”

When Should I Start to Do this HW?

- Now!
- Starting early does not guarantee a high score but starting few days before the due usually means you will fail to obtain most of the score.

What tools do you need?

If-else

```
if expression:
    #suite
    pass
elif expression:
    #suite
    pass
else:
    #suite
    pass
```

It selects exactly one of the suites by evaluating the expressions one by one until one is found to be true then that suite is executed (and no other part of the if statement is executed or evaluated) ;

If all expressions are false, the suite of the else clause, if present, is executed.

For-Loop

```
for target_list in expression_list:
    #suite
    pass
else:
    #suite
    pass

while expression:
    #suite
    pass
```

The expression iterable object ;

Each item in turn is assigned to the target list using the standard rules for assignments and then the suite is executed;

The suite is then executed once for each item provided by the iterator, in the order returned by the iterator;

When the sequence is empty or an iterator raises a StopIteration exception, the suite in the else clause, if present, is executed, and the loop terminates.

While-Loop

```
while expression:
    #suite
    pass
```

This repeatedly tests the expression and

if it is true, executes the first suite

if the expression is false (which may be the first time it is tested) the suite of the else clause, if present, is executed and the loop terminates

Break and continue have same effects as in for loop

Break and Continue

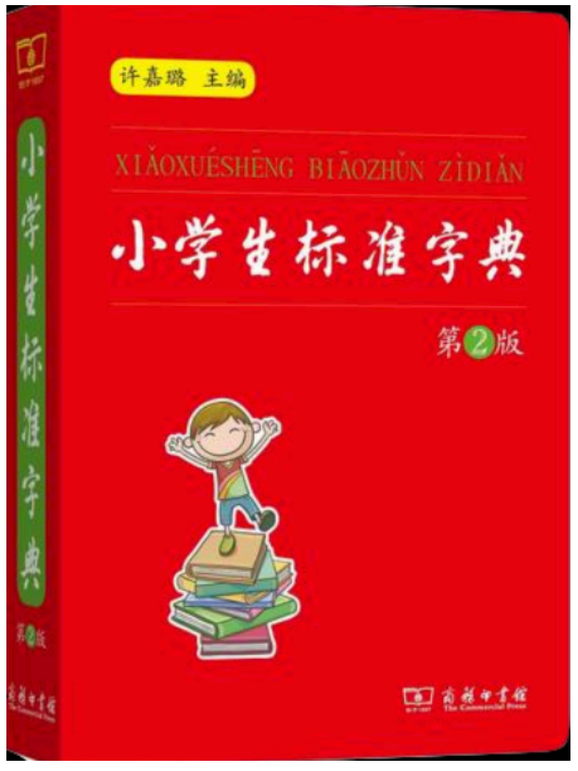
To alter flow of control inside loop:

Execution of break causes immediate termination of the inner most enclosing loop

Execution of continue causes all subsequent statements after the continue statement are not executed for this particular iteration

How we store data

- Dictionary is a data type
- Dictionaries are indexed by keys, which can be any immutable type;
- Dictionary is unordered in contrary to List.
- Examples:




```
>>> dict = {'apple': 1222, 23: 'banana'}
>>> dict['apple']
1222
>>> dict[23]
'banana'
>>>
```

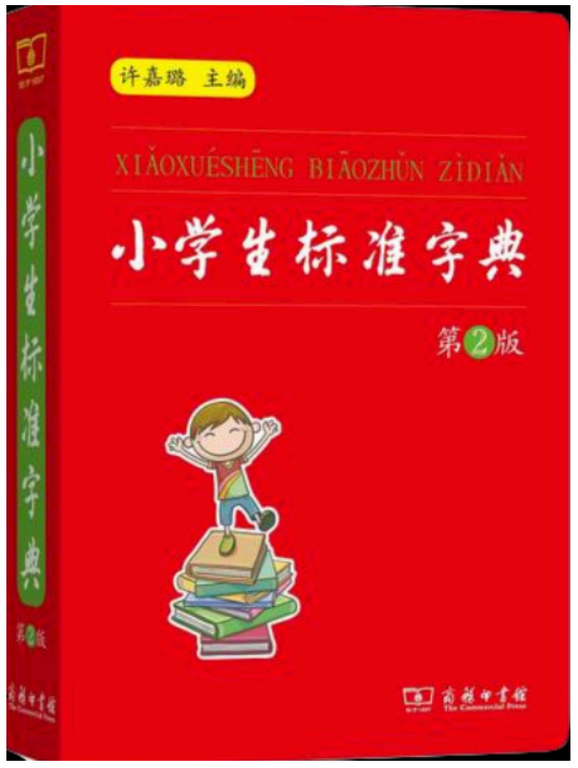
How we store data

- Operations on dictionary
- Additional examples:

```
>>> dict = {'apple': 1222, 23: 'banana'}
>>> dict['peach'] = 7777
>>> dict
{'apple': 1222, 23: 'banana', 'peach': 7777}
```

- Deletion example:

```
>>> dict
{'apple': 1222, 23: 'banana', 'peach': 7777}
>>> del dict[23]
>>> dict
{'apple': 1222, 'peach': 7777}
```



more detail on: https://docs.python.org/3.8/tutorial/datastructures.html#dictionaries

How we store data

- List is a compound data type used to group together other values.
- Lists might contain items of different types, but usually the items all have same type
- List is ordered and values inside are accessed by index

- Example:

```
>>> squares = [1, 4, 9, 16, 25]
>>> squares
[1, 4, 9, 16, 25]
>>> squares[0]
1
>>> squares[-1]
25
>>> squares[-3:]
[9, 16, 25]
```

more detail on : https://docs.python.org/3.8/tutorial/introduction.html#lists

How we store data

- Operations on list
- Addition examples:

```
>>> squares
[1, 4, 9, 16, 25]
>>> squares.append(2333)
>>> squares
[1, 4, 9, 16, 25, 2333]
```

- Deletion examples:

```
>>> squares
[1, 4, 9, 16, 25, 2333]
>>> del squares[0]
>>> squares
[4, 9, 16, 25, 2333]
```

- Another useful function on list len(): return the number of elements in list

```
>>> squares
[4, 9, 16, 25, 2333]
>>> len(squares)
5
```

more detail on : https://docs.python.org/3.8/tutorial/introduction.html#lists

String Processing Methods

String-specific methods (search)

- **s.find(sub[, start[, end]]) return the lowest index in S where substring sub is found such that sub is contained within s[start:end]; Return -1 on failure**

- **s.rfind(sub[, start[, end]]) return the highest index in S where substring sub is found such that sub is contained within s[start:end]; Return -1 on failure**

- **s.index(sub[, start[, end]]) and s.rindex(sub[, start[, end]]) are similar to s.find and s.rfind, except that Raises ValueError on failure**

- **s.count(sub[, start[, end]]) return the number of non- overlapping occurrences of substring sub in string s[start:end]**

more detail on: https://docs.python.org/3.7/library/stdtypes.html

String Processing Methods

String-specific methods (split and partition)

- **s.split(self, sep=None, maxsplit=-1) return a list of the of words (max. No. maxsplit, -1:no limit) in the string, using sep as the delimiter string from left to right, seps are excluding in the list; sep=None removes all whitespaces and splits by space**

- **s.rsplit(self, sep=None, maxsplit=-1) similar to s.split, but right to left**

- **s.partition(self, sep) return a 3-tuple t of words in the string, using sep as the delimiter string from left to right;**

- **s.rpartition (self, sep) similar to s.partition but right to left**

more detail on: https://docs.python.org/3.7/library/stdtypes.html

String Processing Methods

Example for s.split:

```
>>> '1,2,3'.split(',')
['1', '2', '3']
>>> '1,2,3'.split(',', maxsplit=1)
['1', '2,3']
>>> '1,2,,3,'.split(',')
['1', '2', '', '3', '']
```

Lexicographic order

- Given two different words of the same length, say $a = a_1a_2...a_k$ and $b = b_1b_2...b_k$, the order of the two words depends on the alphabetic order of the symbols in the first place i where the two words differ (counting from the beginning of the words): $a < b$ if and only if $a_i < b_i$ in the underlying order of the alphabet A.
- If two words have different lengths, the usual lexicographical order pads the shorter one with "blanks" (a special symbol that is treated as smaller than every element of A) until the words are the same length, and then the words are compared as in the previous case.

more detail: https://en.wikipedia.org/wiki/Lexicographic_order

Lexicographic order

- To be more specific, alphabetic order is the order in ASCII TABLE.
- In python, strings are compared by lexicographic order
- Example:

```
In[8]: a = "ABC"
In[9]: b = "BCD"
In[10]: a < b
Out[10]: True
```

```
In[2]: a = "ABC"
In[3]: b = "ADE"
In[4]: a < b
Out[4]: True
```

ASCII (1977/1986)

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_ 0	NUL 0000	SOH 0001	STX 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
1_ 16	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
2_ 32	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3_ 48	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4_ 64	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5_ 80	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6_ 96	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7_ 112	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F

Letter Number Punctuation Symbol Other Undefined Character changed from 1963 version and/or 1965 draft

Lexicographic order

- To be more specific, alphabetic order is the order in ASCII TABLE.
- In python, strings are compared by lexicographic order
- Example:

```
In[2]: a = '15'
In[3]: b = '123'
In[4]: a > b
Out[4]: True
```

ASCII (1977/1986)

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_ 0	NUL 0000	SOH 0001	STX 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
1_ 16	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
2_ 32	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3_ 48	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4_ 64	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5_ 80	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6_ 96	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7_ 112	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F

Letter Number Punctuation Symbol Other Undefined Character changed from 1963 version and/or 1965 draft

Sort

sort(*, *key=None*, *reverse=False*)

This method sorts the list in place, using only < comparisons between items. Exceptions are not suppressed - if any comparison operations fail, the entire sort operation will fail (and the list will likely be left in a partially modified state).

```
#!/usr/bin/python

# 获取列表的第二个元素
def takeSecond(elem):
    return elem[1]

# 列表
random = [(2, 2), (3, 4), (4, 1), (1, 3)]

# 指定第二个元素排序
random.sort(key=takeSecond)

# 输出类别
print ('排序列表: ', random)
```

```
排序列表: [(4, 1), (2, 2), (1, 3), (3, 4)]
```

What are You going to do in this HW?

Task 0: Set up and Read in Data

Task 0

- **read data from CSV file**

```python
def read_csv_for_data(filename: str):
    """
    Task 0

    @filename: the file name of the file you need to read in;

    @return: your custom data structure.
    """
    with open(filename) as f:
        data = f.read()
    # you may check input by uncommenting next line
    # print(data)
    raise NotImplementedError
```

Task 0

- **parse the data and store data in a structured way, such as** *a list of dictionaries*

```
Python 3.7.6 (v3.7.6:43364a7ae0, Dec 18 2019, 14:12:53)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
In[2]: string = '2011,GS,98,N407S,ANC,SEA,5,2354'
In[3]: string.split(',')
Out[3]: ['2011', 'GS', '98', 'N407S', 'ANC', 'SEA', '5', '2354']
```

Task 0

- **parse the data and store data in a structured way, such as** *a list of dictionaries*

```
example = [
    {
        'flightNumber': 111,
        'airline': 'CA',
        'distance': 333
    },
    {
        'flightNumber': 222,
        'airline': 'SS',
        'distance': 333
    },
    {
        'flightNumber': 333,
        'airline': 'KK',
        'distance': 444
    },
    {
        'flightNumber': 444,
        'airline': 'EA',
        'distance': 555
    }
]
```

Task 1: Who Flies More than 1500 Miles?

Task1

- Read in the data from the CSV file.
- Find the flights that fly more than 1500 miles.
- Return the concatenation of their AIRLINE and FLIGHT_NUMBER.
- The strings in the returned list is ordered in ascending order of their distance

Task1

DISTANCE > 1500

sample.csv

YEAR	AIRLINE	FLIGHT_NUMBER	...	DISTANCE	...
2011	GS	98	...	1448	...
2012	AA	2336	...	2330	...
2013	SS	840	...	1896	...
2014	CA	258	...	2142	...
2015	QS	135	...	1348	...
2015	NK	451	...	1334	...
2015	NK	972	...	868	...
2015	AA	1323	...	192	...

So, you should return ['SS840', 'CA258', 'AA2336']

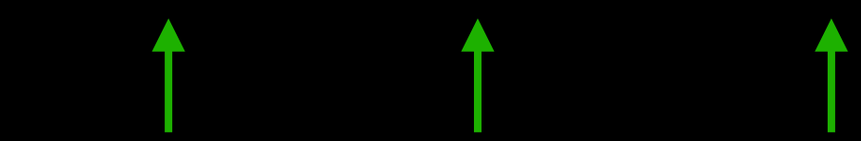
Task 1

- **Order matters!!!**

Example:

return is ordered in **ascending** order of their distance

1896 **2142** **2330**



[‘SS840’, 'CA258', 'AA2336']

Task 1

- **Order matters!!!**

Example:

return is ordered in ascending order of their distance

when two flights have same distance, compare their by dictionary order.


```
Python 3.7.6 (v3.7.6:43364a7ae0, Dec 18 2019, 14:12:53)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
In[2]: "AA2336" < "QS128"
Out[2]: True
```

Task 2: Information Filtering

Task2

- Read in the data from the CSV file.
- Implement task2 function.

```python
def task2(filename: str, airline: str, key: str, value) -> int:
    """
    Task 2

    @filename: the file name of the data file.
    @airline: the airline as a string of IATA code.
    @key: the key that we want to filter against.
    @value: the value that we want to filter against.

    @return: the number of flights that satisfy as an integer.
    """
    # Read in the data from the CSV file `filename` with tools you build in task 0.
    # keys, rows = read_csv_for_data(filename) or data = read_csv_for_data(filename)

    # Now write your filtering code.
    ## YOUR CODE HERE ##
    raise NotImplementedError
```

- return the number of the flights from airline (in the format of IATA carrier cod e), the value of whose key is **less than** value.

Task2

- Input parameter value should be treated differently according to the type of input parameter key
- value is treated as integer when key is one of `YEAR`, `FLIGHT_NUMBER`, `SCHEDULED_DEPARTURE`, `DEPARTURE_TIME`, `DEPARTURE_DELAY`, `TAXI_OUT`, `WHEELS_OFF`, `ELAPSED_TIME`, `DISTANCE`, `WHEELS_ON`, `TAXI_IN`, `SCHEDULED_ARRIVAL`, `ARRIVAL_TIME`, `ARRIVAL_DELAY`
- value is treated as string when key is one of `AIRLINE`, `TAIL_NUMBER`, `ORIGIN_AIRPORT`, `DESTINATION_AIRPORT`.

Task2

- An example: call task2("sample.csv", "AA", "YEAR","2015");
- The number of flights from American Airlines (whose IATA code is AA) whose YEAR is less than 2015

sample.csv

YEAR	AIRLINE	FLIGHT_NUMBER	…
2011	GS	98	…
2012	AA	2336	…
2013	SS	840	…
2014	CA	258	…
2015	QS	135	…
2015	NK	451	…
2015	NK	972	…
2015	AA	1323	…

Valid

Invalid, since the YEAR should be less than 2015

So, you should return 1

Task 3: On-time Rate of Airlines

Task3

- Read in the data from the CSV file.
- On-time rate:
  - Let N be the number of flights whose ARRIVAL_DELAY < 0.
  - On-time rate is N / the total number of flights of the airline.
- Return a list of tuples with format of (airline, rate)
- Sort the list in decreasing order of rate which represents the on-time rate of the given airline.
- If the on-time rate of two airlines happen to be the same, they should be lexicographically sorted in the increasing order of airline.

Task3

- An example

sample.csv

	YEAR	AIRLINE	FLIGHT_NUMBER	...	ARRIVAL_DELAY
1 / 1 = 1.0	2011	GS	98	...	-22
2 / 2 = 1.0	2012	AA	2336	...	-9
0 / 1 = 0.0	2013	SS	840	...	5
1 / 1 = 1.0	2014	CA	258	...	-9
1 / 1 = 1.0	2015	QS	135	...	-21
	2015	NK	451	...	-33
1 / 2 = 0.5	2015	NK	972	...	1
	2015	AA	1323	...	-11

So, you should return

[('AA', 1.0), ('CA', 1.0), ('GS', 1.0), ('QS', 1.0), ('NK', 0.5), ('SS', 0.0)]

Happy Coding!*

*Do remember to start early!