

Introduction to Computation

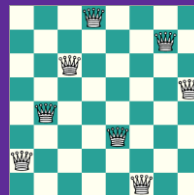
Autumn, 2023

Prof. Fan Cheng

Shanghai Jiao Tong University

chengfan85@gmail.com

<https://github.com/ichengfan/itc>





Outline

- Tuple
- Dict
- String
- Set

Tuple



Tuple (元组)

- In Python tuples are written with **round brackets ()** (the round brackets can be omitted)

tuple_a = (element1, element2, ...,)

- To create a tuple with a single element, we need to add a “,” at the end

tuple_a = (element1,)

- tuple() method will make a tuple (similar to list()) (tuple类型的构造函数)

- Tuple is ordered and immutable (不可变).

- len(), type(), [], in and not in, for

```
tup = ("hello world", 1896, 2018, 3.14, "Frog", [1,2,3], (3,4,5))
print(tup, type(tup), len(tup))
print(tup[4])
print(tup[-1])
print(tup[1:3])
print(tup[:3])
print(tup[5:2:-1])
```

```
('hello world', 1896, 2018, 3.14, 'Frog', [1, 2, 3], (3, 4, 5)) <class 'tuple'> 7
Frog
(3, 4, 5)
(1896, 2018)
('hello world', 1896, 2018)
([1, 2, 3], 'Frog', 3.14)
```

```
tup = (1,2,3)
tup_1 = 1,2,3
print(tup, tup_1)
```

```
(1, 2, 3) (1, 2, 3)
```

```
tup_1 = (1)
tup_2 = 1,
tup_3 = (1,)
print(type(tup_1), type(tup_2), type(tup_3))
```

```
<class 'int'> <class 'tuple'> <class 'tuple'>
```

```
a = tuple([1, 2, 3, 2, 1, 4, 1, 3, 4])
print(a)
```

```
(1, 2, 3, 2, 1, 4, 1, 3, 4)
```

元组：不可修改的list

Tuple: unchangeable

- When you modify tuple in program, there will be a grammar error. Include del

```
a = (2.0, 9, 3)
a[0] = 9
```

```
Traceback (most recent call last):
  File "C:\Users\fcheng\OneDrive\CS124计算导论\2018\lecture notes\1.py", line 2, in <module>
    a[0] = 9
TypeError: 'tuple' object does not support item assignment
```

- +, * still works for tuple since we don't modify a tuple but create one!

```
a = (2.0, 9, 3)
print(a)
b = a * 3
a = a + (4,)
print(a)
print(b)
```

```
(2.0, 9, 3)
(2.0, 9, 3, 4)
(2.0, 9, 3, 2.0, 9, 3, 2.0, 9, 3)
```

- Simultaneous assignment is based on Tuple: a, b = 2, 3
 - The left and the right should have the same number of elements

```
a, b = 1, 2, 3
```

```
Traceback (most recent call last):
  File "C:\Users\fcheng\OneDrive\CS124计算导论\2018\lecture notes\1.py", line 1, in <module>
    a, b = 1, 2, 3
ValueError: too many values to unpack (expected 2)
```

```
a, b, c, d = 1, 2, 3
```

```
ValueError: not enough values to unpack (expected 4, got 3)
```

Tuple: packing and unpacking



```
x, y, z, w = 1, "2", 3.0, [4j]
a = x, y, z, w
print(type(a))
a1, a2, a3, a4 = a
print(a1, a2, a3, a4)
```

```
<class 'tuple'>
1 2 3.0 [4j]
```

- Packing: several variables can be grouped into a tuple
 - `x = a, b, c, d` # `x` will be a tuple with four elements
- Unpacking: a tuple can be broken up into several elements
 - `x1, x2, x3, x4 = x` # Then `x1 = a, x2=b, x3=c, x4=d`
 - **The left and right should have the same number of elements**

```
a = 1, 2, 3, 4
x, y, z = a
```

```
Traceback (most recent call last):
  File "C:\Users\fccheng\OneDrive\CS124计算导论\2018\lecture notes\1.py", line 8, in <module>
    1 2 3.0 [4j]
      x, y, z = a
ValueError: too many values to unpack (expected 3)
```

List VS. Tuple

- List可修改, Tuple不可修改
- 某些操作, list是直接修改, 而tuple是创建一个新的

```
16 lst = [1, 2, 3]
17 st = (1, 2)
18
19 print(lst, id(lst))
20 print(st, id(st))
21
22
23 lst += [4]
24 st += (3, 4)
25
26 print(lst, id(lst))
27 print(st, id(st))
```

```
[1, 2, 3] 2742421490048
(1, 2) 2742421183680
[1, 2, 3, 4] 2742421490048
(1, 2, 3, 4) 2742421108544
```

```
5 x = [1, 2, 3]
6 a = (x)
7 b = (x,)
8 print(a, b)
```

```
[1, 2, 3] ([1, 2, 3],)
```

```
a = (1, [1,2,3], 2, 3)
print(a)
a[1][0] = -1
print(a)
```

```
(1, [1, 2, 3], 2, 3)
(1, [-1, 2, 3], 2, 3)
```

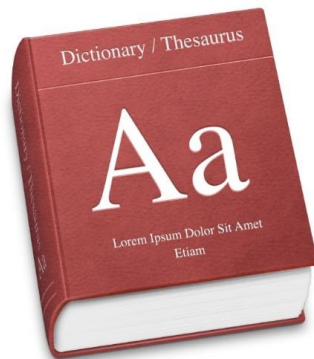
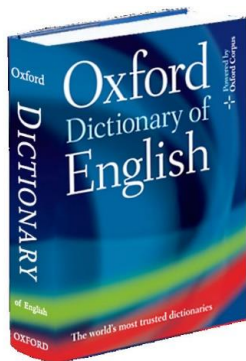
Quick test

- Given a tuple `a = (1,2,3)`
 - What is the result of `a[0] = -1`?
 - What is `print(a[2:1:-1])`
 - `x, y, z = a`, what are `x`, `y` and `z`?
- `a=1`与`a=(1,)`的差异

Dict



Dictionary



- In Dict, we use **keys** to index values; it is unordered
- Dict: Words and their interpretations
- Word -- Key(关键字)
- Interpretation – Value(值)

Dict is more efficient than list

Dict: Creation

- The elements of dict should be a pair like: **key:value**. “林则徐”:“民族英雄”
- All the pairs are enclosed by {}

```
a = {} # empty dict
print(type(a))
a = {"hello" : 1} # dict with 1 element
print(type(a))
b = {"test": "SJTU", "print": 1} # dict with 2 elements
print(type(b))
```

```
<class 'dict'>
<class 'dict'>
<class 'dict'>
```

- The key can be any immutable type. 关键字是唯一的.

```
a = {"test":1, 1.2:3, 3:"hello", 5:[1,2,3]}
print(a, type(a))
```

```
{'test': 1, 1.2: 3, 3: 'hello', 5: [1, 2, 3]} <class 'dict'>
```

- List type cannot be the key

```
a = {[1,2,3]:4} #TypeError: unhashable type: 'list'
```

- len(dict) returns the length of dict

```
a = {"test":1, 1.2:3, 3:"hello", 5:[1,2,3]}
print(len(a))
```

4

List: []
Tuple: ()
Dict: {}

Dict: {}, get()

- Values of a dictionary can be accessed via `dict[key]`. If the key is not existed in the dict, an error will arise
- Suppose we have a dict as follows:
 - `chn = {"Hello": "你吼啊", "World": "世界", "SJTU": "上海交大", "PI": 3.14, "Second": "秒"}`
 - All its keys are: “Hello”, “World”, “SJTU”, “PI”, “Second”
 - An error will arise if you use other values to access the chn

```
chn = {"Hello": "你吼啊", "World": "世界", "SJTU": "上海交大", "PI": 3.14, "Second": "秒"}
print(chn["Hello"])
print(chn["SJTU"])
print(chn["PI"])
```

```
你吼啊
上海交大
3.14
```

```
print(chn[0]) # KeyError: 0
print(chn[1]) # KeyError: 1
```

- `get()` that gives you the same result as `[]` does

```
chn = {"Hello": "你吼啊", "World": "世界", "SJTU": "上海交大", "PI": 3.14, "Second": "秒"}
print(chn["Hello"], chn.get("Hello"))
print(chn["SJTU"], chn.get("SJTU"))
print(chn["PI"], chn.get("PI"))
```

```
你吼啊 你吼啊
上海交大 上海交大
3.14 3.14
```

In principle, there is **no order** on the keys, unless you specify it. For example: `a = {"test": 1, 1.2: 3, 3: "hello", 5: [1, 2, 3]}`

Dict: Removing Items

- `delete` dict[key]: The del keyword removes the item with the specified key name
- `dict.pop()`: The pop() method removes the item with the specified key name:
- `dict.popitem()`: The popitem() method removes the last inserted item (in versions before 3.7, a random item is removed instead 😊)
- The `del` keyword can also delete the dictionary completely:

```
chn = {"Hello": "你吼啊", "World": "世界", "SJTU": "上海交大", "PI": 3.14, "Second": "秒"}
```

```
chn.pop("Hello")  
print(chn)
```

```
chn.popitem()  
print(chn)
```

```
del chn["PI"]  
print(chn)
```

```
chn.clear()  
print(chn)
```

```
chn = {"Hello": "你吼啊", "World": "世界", "SJTU": "上海交大", "PI": 3.14, "Second": "秒"}  
print(chn)
```

```
del chn  
print(chn)
```

```
{'Hello': '你吼啊', 'World': '世界', 'SJTU': '上海交大', 'PI': 3.14, 'Second': '秒'}
```

Traceback (most recent call last):

```
File "c:/Users/popeC/OneDrive/CS124计算导论/2020 秋季/lecture notes/course_code.py", line 348, in <module>  
    print(chn)
```

NameError: name 'chn' is not defined

```
{'World': '世界', 'SJTU': '上海交大', 'PI': 3.14, 'Second': '秒'}  
{'World': '世界', 'SJTU': '上海交大', 'PI': 3.14}  
{'World': '世界', 'SJTU': '上海交大'}  
{}
```

Dict: modification, copy and merge

- The values can be **reassigned/updated/added** via its key

```
a = {"1":1, "2":2}
a["2"] = -1 # update
a["3"] = 3.14 #add a new one
print(a)
```

```
{'1': 1, '2': -1}
```

```
a = {} # a is empty now
```

- The dictionary in python is **mutable**. You need to make a copy in case of modification: **dict.copy()**
 - It is also possible to use the **dict()** constructor to make a new dictionary

```
a = {"test":1, 1.2:3, 3:"hello", 5:[1,2,3]}
b = a
c = a.copy()
print(id(a), id(b), id(c))
print(a==b, b==c, c==a)
```

```
49958320 49958320 51525456
True True True
```

b is the alias of a; c is a copy of a

```
chn = {"Hello":"你吼啊", "World":"世界", "SJTU":"上海交大", "PI":3.14, "Second":"秒"}
chn1 = dict(chn)
print(chn1 == chn)
print(id(chn1) == id(chn))
```

```
True
False
```

- When you need to merge two dictionaries x and y, **z = {**x, **y}**

```
a = {"test":1, 1.2:3}
b = {3:"hello", 5:[1,2,3]}
c = {**a, **b}
print(c)
```

```
{'test': 1, 1.2: 3, 3: 'hello', 5: [1, 2, 3]}
```

Dict: modification, copy and merge

- For a dictionary dict, there are several methods to get its useful information:
 - dict.keys(), returns all its keys
 - dict.values(), returns all its values
 - dict.items(), returns all its items (key:value pairs)

```
a = {"test":1, 1.2:3, 3:"hello", 5:[1,2,3]}  
print(a.keys(), type(a.keys()), list(a.keys()))  
print(a.values(), type(a.values()), list(a.values()))  
print(a.items(), type(a.items()), list(a.items()))
```

```
dict_keys(['test', 1.2, 3, 5]) <class 'dict_keys'> ['test', 1.2, 3, 5]  
dict_values([1, 3, 'hello', [1, 2, 3]]) <class 'dict_values'> [1, 3, 'hello', [1, 2, 3]]  
dict_items([('test', 1), (1.2, 3), (3, 'hello'), (5, [1, 2, 3])]) <class 'dict_items'>  
[('test', 1), (1.2, 3), (3, 'hello'), (5, [1, 2, 3])]
```

dict.keys(), dict.values(), and dict.items() will not directly return a list. You should transform them yourself.

- in and not in can be used to test the membership

```
print("test" in a)  
print("🐼" not in a)  
print("哈哈" in a)
```

```
True  
True  
False
```

Loop Through a Dictionary

- Print all key names in the dictionary, one by one:
 - for x in dict: print(x)
- Print all values in the dictionary, one by one:
 - for x in dict: print(dict[x])
- You can also use the values() method to return values of a dictionary:
 - for x in thisdict.values(): print(x)
- Loop through both keys and values, by using the items() method:
 - for x in thisdict.items(): print(x,y)

```
chn = {"Hello": "你吼啊", "World": "世界", "SJTU": "上海交大", "PI": 3.14, "Second": "秒"}  
for x in chn:  
    print(x)  
  
for x in chn:  
    print(chn[x])  
  
for x in chn.values():  
    print(x)  
  
for x, y in chn.items():  
    print(x, y)
```

```
Hello  
World  
SJTU  
PI  
Second  
你吼啊  
世界  
上海交大  
3.14  
秒  
你吼啊  
世界  
上海交大  
3.14  
秒  
Hello 你吼啊  
World 世界  
SJTU 上海交大  
PI 3.14  
Second 秒
```


Dictionary Methods

- Python has a set of built-in methods that you can use on dictionaries.

Method	Description
<code>clear()</code>	Removes all the elements from the dictionary
<code>copy()</code>	Returns a copy of the dictionary
<code>fromkeys()</code>	Returns a dictionary with the specified keys and value
<code>get()</code>	Returns the value of the specified key
<code>items()</code>	Returns a list containing a tuple for each key value pair
<code>keys()</code>	Returns a list containing the dictionary's keys
<code>pop()</code>	Removes the element with the specified key
<code>popitem()</code>	Removes the last inserted key-value pair
<code>setdefault()</code>	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
<code>update()</code>	Updates the dictionary with the specified key-value pairs
<code>values()</code>	Returns a list of all the values in the dictionary

Dict应用：统计数量

- Dict的结构中key:value, 其中value可用来保存有用的信息，譬如数量
- 给定一篇文章，统计每个单词出现的数量
 - <http://en.sjtu.edu.cn/about-sjtu/presidents-welcome>

```
speech = '''
```

```
Welcome to Shanghai Jiao Tong University!
```

```
Established in 1896 as Nan Yang College, Shanghai Jiao Tong University is one of the first
```

```
.....
```

```
President of Shanghai Jiao Tong University
```

```
'''
```

```
#print(speech)
lst = speech.split()

dt = {}

for x in lst:
    if x in dt:
        dt[x] += 1
    else:
        dt[x] = 1

ndt = sorted(dt.items(), key=lambda kv: kv[1], reverse = True)
print(ndt)
```

```
[('the', 19), ('and', 19), ('to', 12), ('of', 11), ('in', 8), ('is', 8), ('SJTU', 7), ('its', 7), ('a', 6), ('it', 5), ('has', 5), ('for', 5), ('Shanghai', 3), ('Jiao', 3), ('Tong', 3), ('as', 3), ('unite', 3), ('their', 3), ('university', 3), ('or', 3), ('University', 2), ('one', 2), ('first', 2), ('name', 2), ('book', 2), ('means', 2), ('when', 2), ('with', 2), ('development', 2), ('students', 2), ('education', 2), ('been', 2), ('knowledge', 2), ('are', 2), ('become', 2), ('Welcome', 1), ('University!', 1), ('Established', 1), ('1896', 1), ('Nan', 1), ('Yang', 1), ('College', 1), ('national', 1), ('institutions', 1), ('higher', 1), ('learning', 1), ('China.', 1), ('The', 1), ('"Jiao', 1), ('Tong"', 1), ('comes', 1), ('from', 1), ('Yi', 1), ('Jing.', 1), ('Literally', 1), ('"Jiao"', 1), ('"Tong"', 1), ('harmony.', 1), ('In', 1), ('says', 1), ('heaven', 1), ('earth', 1), ('deep', 1), ('harmony', 1), ('peace', 1), ('blessing', 1), ('descend', 1), ('upon', 1), ('all', 1), ('living', 1), ('things;', 1), ('leaders', 1), ('people', 1), ('combine', 1), ('influences', 1), ('nation', 1), ('enjoys', 1), ('universal', 1), ('flowering', 1), ('prosperity.', 1), ('With', 1), ('shining', 1), ('history', 1), ('more', 1), ('than', 1), ('122', 1), ('years.', 1), ('grown', 1), ('shared', 1), ('weal', 1), ('woe', 1), ('country.', 1), ('At', 1), ('present.', 1), ('implementing', 1), ('grand', 1), ('blueprint', 1), ('future', 1), ('determined', 1), ('make', 1), ('continued', 1), ('efforts', 1), ('build', 1), ('itself', 1), ('into', 1), ('class', 1), ('world.', 1), ('SJTU', 1), ('fondly', 1), ('called', 1), ('dynamic', 1), ('comprehensive', 1), ('excellent', 1), ('education', 1), ('cutting-edge', 1), ('scientific', 1), ('research', 1), ('social', 1), ('service.', 1), ('From', 1), ('early', 1), ('stage', 1), ('development', 1), ('took', 1), ('first-class', 1), ('talents', 1), ('principal', 1), ('mission.', 1), ('And', 1), ('today', 1), ('developed', 1), ('talent', 1), ('idea', 1), ('exploration', 1), ('capacity', 1), ('personality', 1), ('nurturing.', 1), ('Whether', 1), ('Academician', 1), ('newly', 1), ('enrolled', 1), ('freshman', 1), ('whether', 1), ('one's', 1), ('field', 1), ('mechanical', 1), ('engineering', 1), ('contemporary', 1), ('literature.', 1), ('truth-seeking', 1), ('spirit', 1), ('encourages', 1), ('faculty', 1), ('join', 1), ('common', 1), ('search', 1), ('answers', 1), ('fundamental', 1), ('pressing', 1), ('questions', 1), ('that', 1), ('society', 1), ('faced', 1), ('with.', 1), ('Members', 1), ('community', 1), ('united', 1), ('by', 1), ('this', 1), ('spirit', 1), ('pursue', 1), ('intellectual', 1), ('excellence', 1), ('dedicated', 1), ('discovery', 1), ('classroom', 1), ('laboratory', 1), ('community.', 1), ('For', 1), ('members', 1), ('place', 1), ('where', 1), ('dreams', 1), ('start', 1), ('fly.', 1), ('Besides', 1), ('remarkable', 1), ('history', 1), ('also', 1), ('famous', 1), ('honored', 1), ('traditions.', 1), ('Its', 1), ('tradition', 1), ('gratitude', 1), ('responsibilities', 1), ('already', 1), ('deeply', 1), ('rooted', 1), ('hearts', 1), ('alumni', 1), ('some', 1), ('behavior', 1), ('philosophy.', 1), ('Thanks', 1), ('your', 1), ('interest', 1), ('you', 1), ('always', 1), ('welcome', 1), ('visit', 1), ('us', 1), ('on', 1), ('campus', 1), ('through', 1), ('website', 1), ('en.sjtu.edu.cn', 1), ('at', 1), ('any', 1), ('time.', 1), ('Lin', 1), ('Zhongqin', 1), ('President', 1)]
```

Dict is more efficient than list. When you need to store and search elements in a data structure, consider Dict first

String

<https://docs.python.org/3.8/library/string.html>



String: basics

String can be defined by "...", '...', and """"...""""
Escape character: "\\n", "\\t"
input(): always returns a string

- String is **immutable**. You cannot change a string. Yet you can **reassign or create** it
 - `[]`: String can be accessed by `str[i]`
 - `str[start:stop:step]`: The rule for a slice of the string are the same with list and tuple
 - `len(str)`, returns the length of a string
 - `+`: `str1 + str2`, returns the concatenation of two strings
 - `*`: `str*n`, repeats the string `n` times (What if $n \leq 0$?)
 - Loop: for `x` in `str1`:

```
a = "苟利国家生死以，岂因祸福避趋之。"  
b = "Exciting!"  
  
print(a[0], a[2:4], a[ : : -1])  
print(len(a), len(b))  
print(a+b)  
print(b)  
print(b*2)  
print(a+b*3)
```

```
苟 国家 。之趋避福祸因岂，以死生家国利苟  
16 9  
苟利国家生死以，岂因祸福避趋之。Exciting!  
Exciting!  
Exciting!Exciting!  
苟利国家生死以，岂因祸福避趋之。Exciting!Exciting!Exciting!
```

```
a = "江山依旧"  
print(a)
```

江山依旧

```
a[0] = "1"
```

```
TypeError: 'str' object does not support item assignment
```

string is an **immutable** list!

String: >, =, <

Copy from Lect. 3

- In practice, letters are ordered: 'a' < 'b' < 'c' < ... < 'z'
- In ASCII, characters are ordered by their ASCII:
'\n' < '0' < '9' < 'A' < 'Z' < 'a' < 'z'
- In practice, letters are ordered: 'a' < 'b' < 'c' < ... < 'z'
- **Alphabetical order** (https://en.wikipedia.org/wiki/Alphabetical_order) is a system whereby strings of characters are placed in order based on the position of the characters in the conventional ordering of an alphabet. For two strings,
 - Their first letters are compared. If they differ, then the string whose first letter comes earlier in the alphabet comes before the other string
 - If the first letters are the same, then the second letters are compared, and so on
 - If a position is reached where one string has no more letters to compare while the other does, then the first (shorter) string is deemed to come first in alphabetical order
- In python, strings are compared by alphabetical order
 - >=, <=, ==, >, <, !=

How to get ASCII of 'x'

String: comparison

Copy from Lect. 3

```
print("" < "1aA")
print("abc">"Abc")
print("123">"abc")
print("abc"<"xyz")
print("1896"=="1 8 9 6")
print("3.14" == "3.14 ")
print("abc123" >='abc')
print("xyz" <="XYZ")
```

```
True
True
False
True
False
False
True
False
```

String module

- Python provides string module, where many useful constants and methods are included
 - <https://docs.python.org/3/library/string.html>
 - To use this module, `import string`
 - `import` is a keyword, `string` is the name of module

```
import string
print(string.ascii_letters)
print(string.punctuation)
print(string.ascii_lowercase)
```

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
abcdefghijklmnopqrstuvwxyz
```

- `string.ascii_letters` includes all the letters
- `string.punctuation` includes all the punctuation
- `string.ascii_lowercase` includes all the lowercase letters

```
string.ascii_letters
string.ascii_lowercase
string.ascii_uppercase
string.digits
string.hexdigits
string.octdigits
string.punctuation
string.printable
string.whitespace
```


String: methods

- Python string module provides several built-in functions to process strings
 - <https://docs.python.org/3/library/stdtypes.html>
- `str.find(sub, start, end)`: Return the lowest index in the string where substring `sub` is found within the slice `s[start:end]`. Optional arguments `start` and `end` are interpreted as in slice notation. Return -1 if `sub` is not found.

```
fruit = 'banana'
print(fruit.find('a'))
print(fruit.find('na'))
print(fruit.find('na', 3))
print(fruit.find('na', 1, 3))
```

```
1
2
4
-1
```

- `str.count(sub[, start[, end]])`: Return the number of non-overlapping occurrences of substring `sub` in the range `[start, end]`. Optional arguments `start` and `end` are interpreted as in slice notation.

```
str = "exciting!"
print(str.count("i"))
print(str.count("z"))
```

```
2
0
```

split(): split a string

- `str.split()`: Split a string into a **list** where **each word** is a list item:
"I am from SJTU." → ["I", "am", "from", "SJTU"]
- Parameter:
separator: Optional. Specifies the separator to use when splitting the string. By default any whitespace is a separator

```
str1 = "123 13.4 Helloworld"  
a = str1.split()  
print(a)
```

```
str1 = "I am from SJTU."  
a = str1.split()  
print(a)
```

```
['123', '13.4', 'Helloworld']  
['I', 'am', 'from', 'SJTU.']
```

默认分隔符：空格

```
str1 = "123*456*3.14*SJTU-IEEE"  
a = str1.split('*')  
print(a)
```

```
str1 = "hello, my name is Peter, I am 26 years old"  
a = str1.split(", ")  
print(a)
```

```
['123', '456', '3.14', 'SJTU-IEEE']  
['hello', 'my name is Peter', 'I am 26 years old']
```

自定义分隔符

```
str1 = input("Please enter the lengths of edges of a triangle: \n")  
a = str1.split()  
x, y, z = float(a[0]), float(a[1]), float(a[2])  
if x>0 and y>0 and z>0 and x+y>z and y+z>x and z+x>y:  
    print("It is a triangle.")  
else:  
    print("It is not a triangle")
```

实际应用

```
Please enter the lengths of edges of a triangle:  
123 1 1  
It is not a triangle
```

String: join()

- **str.join():** The join function is the inverse of **split()**. It takes a list of strings and concatenates the elements with a space between each pair

```
str1 = "a string, with stuff"
print(str1.split())
str2 = "苟利国家生死以， 岂因祸福避趋之。"
print(str2.split())
```

```
['a', 'string,', 'with', 'stuff']
['苟利国家生死以，', ' ', '岂因祸福避趋之。']
```

List!

```
li1 = ['a', "string", "with", "stuff"]
print("".join(li1))
li2 = ["苟利国家生死以，", " ", "岂因祸福避趋之。"]
print("".join(li2))
```

```
astringwithstuff
苟利国家生死以， 岂因祸福避趋之。
```

- **str.join():** could also specify the separator letter

```
myTuple = ("John", "Peter", "Vicky")
x = "#".join(myTuple)
print(x)
```

```
myTuple = ("John", "Peter", "Vicky")
x = " :) ".join(myTuple)
print(x)
```

```
John#Peter#Vicky
John :) Peter :) Vicky
```

```
myDict = {"name": "John", "country": "Norway"}
mySeparator = "TEST"
```

```
x = mySeparator.join(myDict)
print(x)
```

```
nameTESTcountry
```

iterable

String: more

The strip() method removes any whitespace from the beginning or the end:

- str.isdigit(): Return true if all characters in the string are digits and there is at least one character, false otherwise
- str.upper(): Return a copy of the string with all the cased characters converted to uppercase
- str.rstrip([chars]): Return a copy of the string with trailing characters removed
- str.replace(old, new[, count]): Return a copy of the string with all occurrences of substring old replaced by new

```
three = '3'
print(three.isdigit())

str1 = str.upper("Exciting!")
print(str1)

test = "newlines  \t \n\n\n"
print(test.rstrip())

s = "a string, with stuff"
print(s.replace("stuff", "characters"))
print(s.replace("s", "X", 1))
```

```
True
EXCITING!
newlines
a string, with characters
a Xtring, with stuff
```

Always check the string documents before implementing a function yourself

- String can be processed directly on the content without defining a variable

```
print("Hello world".lower())
print("I am from Shanghai".split())
```

```
hello world
['I', 'am', 'from', 'Shanghai']
```

https://www.w3schools.com/python/python_strings.asp

String.format()

“I am {} from {}".format(name, city)

- {}: placeholder 占位符
- format(): formats the specified value(s) and insert them inside the string's placeholder. The format() method returns the formatted string.
- You may use index or variable to denote the placeholders

```
txt1 = "My name is {fname}, I'am {age}".format(fname = "John", age = 36)
txt2 = "My name is {0}, I'am {1}".format("John",36)
txt3 = "My name is {}, I'am {}".format("John",36)
print(txt1)
print(txt2)
print(txt3)
```

```
My name is John, I'am 36
My name is John, I'am 36
My name is John, I'am 36
```

```
txt1 = "My name is {name1}. He is {name2} and she is {name3}".format( name1= "John", name2 = "Bill", name3 = "Lily")
txt2 = "My name is {name1}. He is {name1} and she is {name1}".format( name1= "John", name2 = "Bill", name3 = "Lily")
txt3 = "My name is {0}. He is {1} and she is {2}".format( "John", "Bill", "Lily")
txt4 = "My name is {2}. He is {1} and she is {2}".format( "John", "Bill", "Lily")
print(txt1)
print(txt2)
print(txt3)
print(txt4)
```

#Accessing arguments by position or name

```
My name is John. He is Bill and she is Lily
My name is John. He is John and she is John
My name is John. He is Bill and she is Lily
My name is Lily. He is Bill and she is Lily
```

String.format(): b, o, x

d: 10进制(默认)

b, o, x: 二、八、十六进制

```
str1 = "0101"  
x1 = int(str1)  
x2 = int(str1, base=2)  
print(x1, x2)
```

```
str1 = "{:b}".format(x1)  
str2 = "{:b}".format(x2)  
print(str1, str2)
```

```
101 5  
1100101 101
```

```
str1 = "{:o}".format(x1)  
str2 = "{:o}".format(x2)  
print(str1, str2)
```

```
145 5  
65 5
```

```
str1 = "{:x}".format(x1)  
str2 = "{:x}".format(x2)  
print(str1, str2)
```

https://www.w3schools.com/python/ref_string_format.asp

<https://docs.python.org/3.8/library/string.html>

String.format(): f, d, c, s

f: 浮点数 d: 10进制整数
c: 字符 s: 字符串

```
print("The lengths are {:.f}, {:.f}, {:.f}.".format(10.1, 8.5, 6.9))  
print("The lengths are {:.0f}, {:.0f}, {:.0f}.".format(10.1, 8.5, 6.9))  
print("The lengths are {:.3f}, {:.3f}, {:.3f}.".format(10.1, 8.5, 6.9))  
print("The lengths are {:.9f}, {:.9f}, {:.9f}.".format(10.1, 8.5, 6.9))
```

默认6位小数

```
print("The lengths are {:.10.3f}, {:.10.3f}, {:.10.3f}.".format(10.1, 8.5, 6.9))  
print("The lengths are {:.>10.3f}, {:.<10.3f}, {:.^10.3f}.".format(10.1, 8.5, 6.9))
```

```
print("The length is {:+d}".format(12))  
print("The length is {:+d}".format(-12))  
print("The length is {: -d}".format(12))  
print("The length is {: -d}".format(-12))  
print("The length is {: d}".format(12))  
print("The length is {: d}".format(-12))  
print("The length is {:=5d}".format(12))  
print("The length is {:=5d}".format(-12))
```

```
The lengths are 10.100000, 8.500000, 6.900000.  
The lengths are 10, 8, 7.  
The lengths are 10.100, 8.500, 6.900.  
The lengths are 10.100000000, 8.500000000, 6.900000000.  
The lengths are 10.100, 8.500, 6.900.  
The lengths are 10.100, 8.500, 6.900.
```

```
print("The length is {:12s}".format("Hello"))  
print("The ASCII/unicode character is {:c}".format(65))
```

```
The length is Hello  
The ASCII/unicode character is A
```

String: old style

“I am %s from %s”.%(name, city)

- %: placeholder 占位符
- f: 浮点数; d: 10进制整数; c: 字符; s: 字符串
- 手工——对齐

```
name = "Xiaoming"  
year = 1997  
score = 87.6  
print("I am %s, and I am %d years old. I got %.3f in CET6."%(name, year, score))
```

I am Xiaoming, and I am 1997 years old. I got 87.600 in CET6.

String.format(): format types

:<	Left aligns the result (within the available space)
:>	Right aligns the result (within the available space)
:^	Center aligns the result (within the available space)
=	Places the sign to the left most position
+	Use a plus sign to indicate if the result is positive or negative
-	Use a minus sign for negative values only
:	Use a space to insert an extra space before positive numbers (and a minus sign before negative numbers)
,	Use a comma as a thousand separator
_	Use an underscore as a thousand separator
b	Binary format
c	Converts the value into the corresponding unicode character
d	Decimal format

https://www.w3schools.com/python/ref_string_format.asp

String.format(): format types

:e	Scientific format, with a lower case e
:E	Scientific format, with an upper case E
:f	Fix point number format
:F	Fix point number format, in uppercase format (show inf and nan as INF and NAN)
:g	General format
:G	General format (using a upper case E for scientific notations)
:o	Octal format
:x	Hex format, lower case
:X	Hex format, upper case
:n	Number format
:%	Percentage format

https://www.w3schools.com/python/ref_string_format.asp

f-String: Formatted string literals

Repr, str

- Python version ≥ 3.6
- Formatted string literals (also called f-strings for short) let you include the value of Python expressions inside a string by prefixing the string with f or F and writing expressions as {expression}.

```
name = "Eric"
age = 74
txt = f"Hello, {name}. You are {age}."
num = f"{2 * 37}"
print(txt)
print(num)
```

```
Hello, Eric. You are 74.
74
```

```
exp1 = f"{70 + 4}"
exp2 = f"{{70 + 4}}"
exp3 = f"{{{70 + 4}}}"
exp4 = f"{{{70 + 4}}}"
```

```
print(exp1, exp2, exp3, exp4)
```

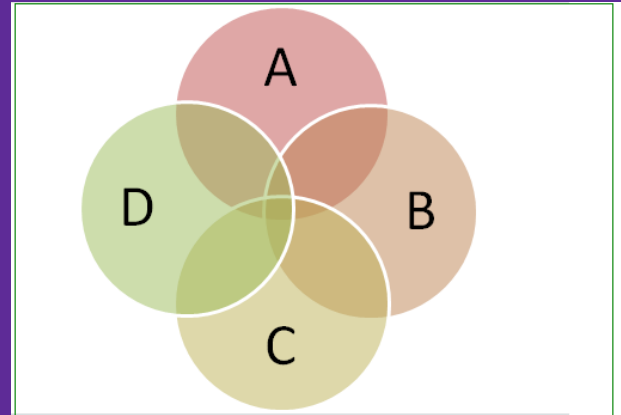
```
import math
print(f'The value of pi is approximately {math.pi:.3f}.')
```

```
The value of pi is approximately 3.142.
```

```
74 {70 + 4} {74} {{70 + 4}}
```

<https://docs.python.org/3/tutorial/inputoutput.html>

Set



Set: create

- A set is an unordered collection with **no duplicate** elements.
 - Basic uses include membership testing and eliminating **duplicate entries** (重复).
- Set objects also support **mathematical operations** like **union, intersection, difference, and symmetric difference**.
- **Curly braces {}** or the **set()** function can be used to create sets.

Note: to create an empty set you have to use `set()`, not `{}`; the latter creates **an empty dictionary**

```
fruit = {'apple', 'orange', 'pear', 'banana'}
```

```
my_set = {1, 2, 3}
print(my_set)

my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)

my_set = {} # it is a dict.
print(type(my_set))

my_set = set()
print(type(my_set))
```

```
{1, 2, 3}
{1.0, (1, 2, 3), 'Hello'}
<class 'dict'>
<class 'set'>
```

set()

- The parameter of set() can be type of string, list, dict
 - List, tuple: all the elements
 - Dict: all the keywords
 - String: all the characters
- You cannot access items in a set by referring to an index, since sets are unordered the items has no index
- To add one item to a set use the add() method
- To add more than one item to a set use the update() method
- To remove an item in a set, use the remove(), or the discard() method
- The clear() method empties the set
- All the above methods can be implemented by basic set operations

```
a = set([1, 2, 3])  
print(a)
```

```
a = set("123")  
print(a)
```

```
a = set((1,2,3))  
print(a)
```

```
a = set({1:2, 2:3, 3:4})  
print(a)
```

```
set([1, 2, 3])  
set(['1', '3', '2'])  
set([1, 2, 3])  
set([1, 2, 3])  
[Finished in 0.0s]
```

```
# a = set(123) #error  
# a = set(1,2,3) #error
```

```
4798 x = [[2]]  
4799 set(x)
```

TypeError: unhashable type: 'list'

```
a = set(range(10))  
print(a)
```

```
a.add(100) # single element  
a.remove(6) # single element
```

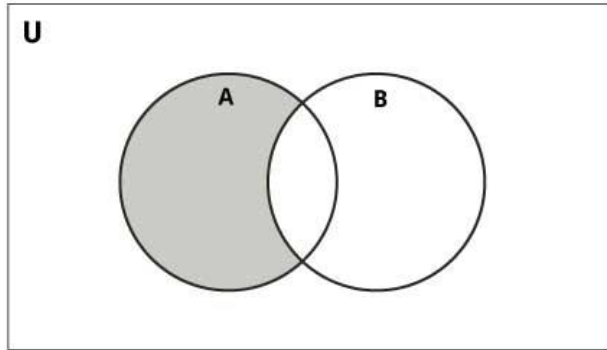
```
a.update([10,12]) # add multiple element  
print(a)
```

```
a.clear()  
print(a)
```

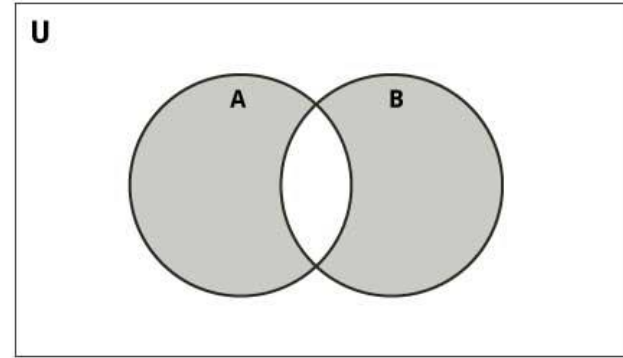
```
set([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])  
set([0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 12, 100])  
set([])
```

Set: operations

- $A|B$: Returns a set which is the **union** of sets A and B
- $A \& B$: Returns a set which is the **intersection** of sets A and B
- $A-B$: Returns the set **difference** of A and B (the elements included in A, but not included in B)
- $A^{\wedge}B$: Returns the **symmetric difference** of sets A and B (the elements belonging to either A or B, but not to both sets simultaneously)



$A-B$



$A^{\wedge}B$

- $A \leq B$: Returns true if A is a subset of B
- $A \geq B$: Returns true if B is a subset of A
- Similar, $A < B$, $A > B$

```

basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
print(basket,type(basket))

print("apple" in basket)
print("Frog" in basket)
print("pear" not in basket)

a = set("Hello world")
b = set("Shanghai Jiao Tong University")
print(a) # unique letters in a
print(b)

print(a-b) # set minus
print(a|b) # set union
print(a&b) # set intersection
print(a^b) # elements in only one set

```

```

{'apple', 'pear', 'banana', 'orange'} <class 'set'>
True
False
False
{'H', ' ', 'o', 'e', 'd', 'l', 'r', 'w'}
{'i', 'a', 'r', ' ', 'o', 'e', 'y', 'g', 'T', 'U', 'h', 's', 't', 'J', 'n', 'S', 'v'}
{'H', 'd', 'w', 'l'}
{'i', 'o', 'y', 'T', 'l', 't', 'U', 'n', 'v', 'H', 'a', ' ', 'e', 'S', 'd', 'g', 'h', 's', 'J', 'r', 'w'}
{'r', ' ', 'o', 'e'}
{'i', 'y', 'T', 't', 'l', 'U', 'n', 'v', 'H', 'a', 'S', 'd', 'g', 'h', 's', 'J', 'w'}

```

```

A = {1, 2, 3, 4, 5, 6, 7}
B = {1, 3, 5, 7, 0, -1, -3, -5}
print(A<=B)
print(A==B)

```

```

False
False

```

Code talks!


Built-in functions

- The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order

Built-in Functions				
<code>abs()</code>	<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>
<code>all()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>any()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>ascii()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>bin()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bool()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>breakpoint()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	

Tips: 当你需要实现一个基本的功能的时候, Python可能已经帮你实现了

How to program

- For most PLs like C++, Java, Python, etc..
- Basic grammar structure (语句的**意义** + 语句的**语法**)
 - Data types: inter, float, string
 - Expression: arithmetic, logic
 - Conditional expression
 - Loops
- Advanced
 - Files
 - Class
 - Standard library
- Complicated  Hard
 - There are thousand methods to implement a feature

Practice makes perfect!