Introduction to Python

Basic input and output

In [1]:

```
print("Hello world!")
print(1, "plus", 2, "equals", 1+2)

name=input("Give me your name: ")
print("Hello,", name)
```

```
Hello world!
1 plus 2 equals 3
Give me your name: Canhong Wen
Hello, Canhong Wen
```

Indentation

- The body of statements, such as bodies of functions, different branches of an `if` statement, `while` loops and `for` loop, is indented with **a tabulator or four spaces**.
- When the indentation stops, the body of the loop ends.

In [2]:

```
for i in range(3):
    print("Hello")
print("Bye!")
```

```
Hello
Hello
Hello
Bye!
```

Data type

The basic data types in Python are: `int`, `float`, `complex`, `str` (a string), `bool` (a boolean with values True and False), and `bytes`.

- `type()`, `dtypes()`

In [3]:

```
i=5
type(i)
```

Out[3]:

```
int
```

In [4]:

```
f=1.5
type(f)
```

Out[4]:

```
float
```

In [5]:

```
c=0+1j                                  # Note that j denotes the imaginary unit of complex numbe
rs.
print("Complex multiplication:", c*c)
print("The real part is", c.real, "and the imaginary part is", c.imag)
```

```
Complex multiplication: (-1+0j)
The real part is 0.0 and the imaginary part is 1.0
```

In [6]:

```
print(3==4)
```

```
False
```

In [7]:

```
s="conca" + "tenation"
print(s)
```

```
concatenation
```

In [8]:

```
len(s)
```

Out[8]:

```
13
```

Transformation of data types

In [9]:

```
print(int(-2.8))
print(float(2))
print(int("123"))
print(bool(-2), bool(0))  # Zero is interpreted as False
print(str(234))
```

```
-2
2.0
123
True False
234
```

Operators and Expressions

- An expression is a piece of Python code that results in a value.
- It consists of values combined together with operators.
- Values can be literals, such as `1`, `1.2`, `"text"`, or variables.
- Operators include arithmetics operators, comparison operators, function call, indexing, attribute references, among others.

In [10]:

```
1+2
```

Out[10]:

3

In [11]:

```
7/(2+0.1)
```

Out[11]:

3.333333333333333

In [12]:

```
2**3
```

Out[12]:

8

In [13]:

```
22%8
```

Out[13]:

6

In [14]:

```
22//8
```

Out[14]:

2

Comparison operators & Boolean operators

- `==`, `!=`, `<`, `>`, `<=`, `>=`.
- `and`, `or`, `not`

In [15]:

```
42==42
```

Out[15]:

True

In [16]:

```
(-1)**2 == 1
```

Out[16]:

True

In [17]:

```
c > 0 and c !=1
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-17-429c2b4174f0> in <module>()
----> 1 c > 0 and c !=1

TypeError: '>' not supported between instances of 'complex' and 'int'
```

In [18]:

```
not True
```

Out[18]:

False

Operators on strings

In [19]:

```
'Statistical' + 'software'
```

Out[19]:

'Statisticalsoftware'

In [20]:

```
'Statistical' + 'software' + 2020
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-111e21b133df> in <module>()
----> 1 'Statistical' + 'software' + 2020

TypeError: can only concatenate str (not "int") to str
```

In [21]:

```
'Statistical' + 'software' + str(2020)
```

Out[21]:

'Statisticalsoftware2020'

In [22]:

```
str2 * 5 ## replicate the string with 5 times
# str2 * 5.0
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-22-a37201cf2450> in <module>()
----> 1 str2 * 5 ## replicate the string with 5 times
      2 # str2 * 5.0

NameError: name 'str2' is not defined
```

In [23]:

```
str1 = 'Statistical'
str2 = 'software'
str_combine = str1 + str2
str_combine.upper()
```

Out[23]:

'STATISTICALSOFTWARE'

In [24]:

```
str_combine.islower()
```

Out[24]:

False

In [25]:

```
str_combine.lower().title()
```

Out[25]:

'Statisticalsoftware'

isX:

- isalpha()
- isalnum()
- isdecimal()
- isspace()
- istitle()
- ...

In [26]:

```
"hello123".isalpha()
```

Out[26]:

False

Paste or split strings

In [27]:

```
', '.join([str1, str2, "2020"])
```

Out[27]:

'Statistical, software, 2020'

In [28]:

```
' '.join([str1, str2, "2020"])
```

Out[28]:

'Statistical software 2020'

In [29]:

```
'Statistical, software, 2020'.split(',')
```

Out[29]:

['Statistical', ' software', ' 2020']

In [30]:

```
'Statistical, software, 2020'.split('s')
```

Out[30]:

['Stati', 'tical, ', 'oftware, 2020']

In [31]:

```
'Statistical software (2020)'.rjust(30)
```

Out[31]:

' Statistical software (2020)'

In [32]:

```
'Statistical software (2020)'.rjust(30, "*")
```

Out[32]:

'***Statistical software (2020)'

In [33]:

```
'Statistical software (2020)'.ljust(30)
```

Out[33]:

```
'Statistical software (2020)    '
```

In [34]:

```
'Statistical software (2020)'.center(50, "=")
```

Out[34]:

```
'==========Statistical software (2020)==========='
```

Data Structures

There are four built-in data structures in Python:

- `list`,
- `tuple`,
- `dictionary`,
- `set`.

In [ ]:

```

```

List

- A `list` is a data structure that holds an ordered collection of items i.e. you can store a sequence of items in a list.
- Specify items separated by commas.
- The list of items should be enclosed in square brackets `[ ]`.
- Create a numeric list automatically: `range(num)`
- Can add and remove items

In [35]:

```
list0=[] #Create an empty list
list1=['团购名','店名','团购活动ID','团购介绍','购买人数','团购评价','评价人数','地址','团购内容'] #创建列表火锅数据字段名列表
list2=['13377118',8,'随便退']
list3=['壹分之贰聚会吧桌游轰趴馆(旗舰店)',list2] #列表嵌套
print(list3)
```

```
['壹分之贰聚会吧桌游轰趴馆(旗舰店)', ['13377118', 8, '随便退']]
```

In [36]:

```
len(list3)
```

Out[36]:

2

In [37]:

```
list3[0]
```

Out[37]:

'壹分之贰聚会吧桌游轰趴馆(旗舰店)'

In [38]:

```
list3[-1]
```

Out[38]:

['13377118', 8, '随便退']

In [39]:

```
list3[1][0]
```

Out[39]:

'13377118'

Basic functions in list

- Type
  - `type(list)`
- Add an item:
  - `mylist.append(newelement)`
  - `mylist.insert(position, newelement)`
- Append two list
  - `list1 + list2`
- Length:
  - `len(list)`
- Element:
  - `list[i]`, `list[:length]`, `list[start:end]`, `in`
- Delete a list:
  - `del`
- Find an element:
  - `index()`
- Find more :
  - `help(list.extend)` or `help(list)`

In [40]:

```
list4=['到期时间','团购价','市场价','备注','购买须知']
print(type(list4))

list6 = list4
print(list6)
list6.append('团购名') #在列表的最后的位置加入"团购名"
print(list6)
list6.insert(0,'店名') #在列表的第一个位置加入"店名"
print(list6)
list6.append('团购名') #在列表的最后的位置再加入"团购名"
print(list6)
print(list6.count('团购名')) #计算有多少个元素是"团购名", 返回值为2
```

```
<class 'list'>
['到期时间', '团购价', '市场价', '备注', '购买须知']
['到期时间', '团购价', '市场价', '备注', '购买须知', '团购名']
['店名', '到期时间', '团购价', '市场价', '备注', '购买须知', '团购名']
['店名', '到期时间', '团购价', '市场价', '备注', '购买须知', '团购名', '团购名']
2
```

In [41]:

```
list5=list1+list4
print('List5 =', list5)
print('Sorted list5 =',list5.sort())
```

```
List5 = ['团购名', '店名', '团购活动ID', '团购介绍', '购买人数', '团购评价', '评价
人数', '地址', '团购内容', '店名', '到期时间', '团购价', '市场价', '备注', '购买须
知', '团购名', '团购名']
Sorted list5 = None
```

In [42]:

```
len(list5)
```

Out[42]:

```
17
```

In [43]:

```
print (list5[:])
print (list5[0])
print (list5[:3]) # length = 3
print (list5[2:7])
```

```
['到期时间', '团购介绍', '团购价', '团购内容', '团购名', '团购名', '团购名', '团购
活动ID', '团购评价', '地址', '备注', '市场价', '店名', '店名', '评价人数', '购买人
数', '购买须知']
到期时间
['到期时间', '团购介绍', '团购价']
['团购价', '团购内容', '团购名', '团购名', '团购名']
```

In [44]:

```
list5.remove('到期时间')
len(list5)
```

Out[44]:

16

In [45]:

```
del list5
list5
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-45-00cbe1778fa8> in <module>()
      1 del list5
----> 2 list5

NameError: name 'list5' is not defined
```

In [46]:

```
list4
```

Out[46]:

['店名', '到期时间', '团购价', '市场价', '备注', '购买须知', '团购名', '团购名']

In [47]:

```
list4.index('店名')
```

Out[47]:

0

In [48]:

```
list4.index('Name')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-48-4720ba27c483> in <module>()
----> 1 list4.index('Name')

ValueError: 'Name' is not in list
```

In [49]:

```
list4.index('团购名')
```

Out[49]:

6

Sorting

- The `sort` method modifies the original list
- The `sorted` function returns a new sorted list and leaves the original intact.

In [50]:

```
L=[5,3,7,1]
L.sort()      # here we call the sort method of the object L
print(L)
```

```
[1, 3, 5, 7]
```

In [51]:

```
L2=[6,1,7,3,6]
print(L2)
```

```
[6, 1, 7, 3, 6]
```

In [52]:

```
print(sorted(L2))
```

```
[1, 3, 6, 6, 7]
```

In [53]:

```
print(L2)
```

```
[6, 1, 7, 3, 6]
```

In [54]:

```
print(sorted(L, reverse=True)) # descending order
```

```
[7, 5, 3, 1]
```

Tuple

- Similar to `list`.
- Cannot modify tuples, can not add and remove items
- Specify items separated by commas within an optional pair of parentheses.
- Usually used in cases where a statement or a user-defined function can safely assume that the collection of values (i.e. the tuple of values used) will not change.

In [55]:

```
tuple0=() # create an empty tuple
tuple1=(2,4,5)
tuple2=('a','b','c')
tuple3=(21,'a','c')

print('Number of elements in the tuple1 is', len(tuple1))
print('The second element of tuple2 is', tuple2[1])
print('The first two elements of tuple2 are', tuple2[0:1])
```

```
Number of elements in the tuple1 is 3
The second element of tuple2 is b
The first two elements of tuple2 are ('a',)
```

In [56]:

```
tuple4=tuple2+tuple3
print('The combination of tuple2 and tuple3 is ', tuple4)
tuple5=tuple4[2:5]
print('Some elements of tuple5 are', tuple5)
```

```
The combination of tuple2 and tuple3 is  ('a', 'b', 'c', 21, 'a', 'c')
Some elements of tuple5 are ('c', 21, 'a')
```

In [57]:

```
tuple2[2]=4 # Cannot change the value
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-57-7ee9b6d2023c> in <module>()
----> 1 tuple2[2]=4 # Cannot change the value

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

Dictionary

- keys (name) with values (details): `d = {key1 : value1, key2 : value2 }`
- key-value pairs are separated by a colon and the pairs are separated themselves by commas and all this is enclosed in a pair of curly braces
- key-value pairs in a dictionary are not ordered in any manner
- `keys()` , `values()` , `items` , `get()`

In [58]:

```
dict0={'六婆串串香火锅':6,'嗲串串':14}
print('Type of dict0 is', type(dict0))
```

```
Type of dict0 is <class 'dict'>
```

In [59]:

```
print('嗲串串有',dict0['嗲串串'], '人团购')
```

```
嗲串串有 14 人团购
```

In [60]:

```
dict0['古城串串']=0
print('The new version of dict0 is',dict0)
```

The new version of dict0 is {'六婆串串香火锅': 6, '嗲串串': 14, '古城串串': 0}

In [61]:

```
dict0['古城串串']=2
print('The updated version of dict0 is',dict0)
```

The updated version of dict0 is {'六婆串串香火锅': 6, '嗲串串': 14, '古城串串': 2}

In [62]:

```
del dict0['古城串串']
print('The updated version of dict0 is',dict0)
```

The updated version of dict0 is {'六婆串串香火锅': 6, '嗲串串': 14}

In [63]:

```
print('商家有：', dict0.keys())
```

商家有： dict_keys(['六婆串串香火锅', '嗲串串'])

In [64]:

```
print('对应的团购人数为：', dict0.values() )
```

对应的团购人数为： dict_values([6, 14])

In [65]:

```
print(dict0.items()) #返回值[('六婆串串香火锅', 6), ('嗲串串', 14)]
```

dict_items([('六婆串串香火锅', 6), ('嗲串串', 14)])

In [66]:

```
print('嗲串串有',dict0.get('嗲串串'), '人团购')
```

嗲串串有 14 人团购

In [67]:

```
'嗲串串' in dict0.keys()
```

Out[67]:

True

In [68]:

```
'海底捞' not in dict0.keys()
```

Out[68]:

True

Set

Sets are unordered collections of simple objects.

In [69]:

```
set1 = {1, 2, 's', 1, 1}
set1
```

Out[69]:

```
{1, 2, 's'}
```

In [70]:

```
type(set1)
```

Out[70]:

```
set
```

In [71]:

```
set2 = set([1,3,4,5])
print(set2)
```

```
{1, 3, 4, 5}
```

In [72]:

```
set3=set((1,3,'e'))
print(set3)
```

```
{1, 'e', 3}
```

In [73]:

```
set('apple')
```

Out[73]:

```
{'a', 'e', 'l', 'p'}
```

In [74]:

```
set(['apple'])
```

Out[74]:

```
{'apple'}
```

In [75]:

```
A={1,2,3}
B={3,4,5}
```

In [76]:

```
A - B        ## set difference
```

Out[76]:

{1, 2}

In [77]:

```
A | B       ## union
```

Out[77]:

{1, 2, 3, 4, 5}

In [78]:

```
A & B        ## intersection
```

Out[78]:

{3}

In [79]:

```
set4 = set(['grape', 'apple', 'banana', 'pear'])
print(set4)
```

{'apple', 'pear', 'grape', 'banana'}

In [80]:

```
set4.add('orange')
print(set4)
```

{'apple', 'orange', 'banana', 'pear', 'grape'}

In [81]:

```
set4.remove('orange')
print(set4)
```

{'apple', 'banana', 'pear', 'grape'}

In [82]:

```
set4.clear()
print(set4)
```

set()