Python编程

Python Programing

数据类型

内置数据类型

- 数字：1， 3， 3.14
- 字符串：‘Python’, “MachineLearning”
- 列表: [1, 2, [1,34]]
- 字典: {‘food’:’spam’}
- 元组： (2, ‘U’, 3)
- 集合：{‘a’, ‘b’, ‘d’}
- 文件: f = open(‘eggs’, r)
- 其他类型：None, 布尔

数字

- 数字在Python中是不可变的对象，不可以改变它的值
- Python中有三种内置的数字类型
  - 整数
  - 浮点类型
  - 复数（在python中很少被用到）

数字

- 可以使用的算子

可以使用的算子	说明
M + N	加法
M - N	减法
M * N	乘法
M / N	除法
M % N	模运算
M ** N	幂运算

数字

- python整数上限是内存能接受的最大范围

```
>>> import sys
>>> sys.maxsize
9223372036854775807
>>> 2**63 -1
9223372036854775807
>>> 100000000000000000000000000000 > sys.maxsize
True
```

数字

- python float 可以用floa_info查看

```
>>> import sys
>>> sys.float_info
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308, min=2.2250738585072014e-308, min_exp=-1021, min_10_exp
=-307, dig=15, mant_dig=53, epsilon=2.220446049250313e-16, radix=2, rounds=1)
>>> sys.float_info.max
1.7976931348623157e+308
```

数字

- Python中是不可以使用自增运算的
- +=, -=, *=, /=, %=是可以的

```
Java

int i = 0;
i++;

--------------------
Python
i = 0
i += 1 <=> i = i + 1
```

字符串

- 字符串在Python中是不可变的对象，不可以改变它的值

```
>>> str= "strings are immutable!"
>>> str[0]="S"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

- 可以通过将字符串复制到另外一个变量来更新

```
>>> str1 = 'string1'
>>> str2 = str1 + '_string'
>>> str2
'string1_string'
>>>
```

字符串

- 3种字符串的创建方式

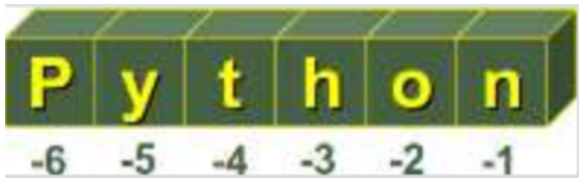
```
单引号方式
str1 = 'This is a python string.'

双引号
str2 = "This is a python string."

三个引号
str3 = """This is a
python
string."""
```

字符串

- Python 不支持字符类型，对于字符类型在python中被认为是长度为1的字符串
- Python的字符串可以看作一个数组，第一位的index是0。在Python中还可以从后向前遍历，最后一位的index是-1.



字符串格式化

There should be one-- and preferably only one --obvious way to do it.
— the zone of python
https://www.python.org/dev/peps/pep-0020/

字符串格式化

```python
name = 'Tom'
age = 3
# 方式1
print("His name is {}, he's {} years old.".format(name, age))

# 方式2
print(f"His name is {name}, he's {age} years old")

# 方式2 支持数学表达式
print(f"His name is {name}, he will be {age + 1} years old next year.")

# 方式2 支持对象操作
list_obj = ['a', 'b', 'c']
print(f'the length of list is {len(list_obj)}')
print(f'the elemets of list obj is {[_ for _ in list_obj]}')
```

字符串格式化（对数字的格式化）

```
1 x = 1234.567
2 #  对数字的格式化
3 #  显示小数点后两位
4 print("x is {:0.2f}".format(x))
5 print(f"x is {x:0.2f}")
6
7 # 向右移动满足10个字符
8 print("x is {:>10.1f}".format(x))
9 print(f"x is {x:>10.1f}")
10
11 # 向左移动满足10个字符
12 print("x is {:<10.1f}".format(x))
13 print(f"x is {x:<10.1f}")
```

字符串连接与复制

- Python的字符串使用加号进行连接，使用乘号进行重复

```
>>> print('I love ' + 'Python.')
I love Python.
>>> print('Hello Python ' * 3)
Hello Python Hello Python Hello Python
>>> print('I say ' + "'Hello Python'" * 3)
I say 'Hello Python''Hello Python''Hello Python'
>>> x = 'I love'
>>> y = 'Python.'
>>> print(x + ' ' + y)
I love Python.
```

字符串

- Python字符串的切片操作

```
string = 'abcdefg'
print('string[0] : {}'.format(string[0]))
print('string[1] : {}'.format(string[1]))
print('string[-1] : {}'.format(string[-1]))
print('string[-2] : {}'.format(string[-2]))
print('string[begin, end] 是截取从begin开始到end前一位的字符串')
print('string[0: 0] : {}'.format(string[0: 0]))
print('string[0: 1] : {}'.format(string[0: 1]))
print('string[0: 2] : {}'.format(string[0: 2]))
print('string[-2: -1] : {}'.format(string[-2: -1]))
```

```
root@c0f233ad99b1:/workspace# python test.py
string[0] : a
string[1] : b
string[-1] : g
string[-2] : f
string[begin, end] 是截取从begin开始到end前一位的字符串
string[0: 0] :
string[0: 1] : a
string[0: 2] : ab
string[-2: -1] : f
```

字符串

- 可以使用in关键字判断字符串是否包含某个字符

```
>>> 'a' in 'Python'
False
>>> 'p' in 'Python'
False
>>> 'P' in 'Python'
True
>>>
```

字符串

- 常用方法

str.count(sub, beg=0, end=len())	指定字符串在原字符串或原字符串中出现的次数
str.isalpha()	如果全是字母的话返回True，否则返回Flase
str.isdigit()	如果全是数字的话返回True，否则返回Flase
str.lower()	转换为小写字符
str.upper()	转换为大写字符
str.replace(old, new)	用new替换old
str.strip()	删除字符串首部和尾部的空格
str（x）	将x转换为string类型
len（x）	计算字符串x的长度

List

- 列表是一个有顺序的集合，列表中的元素类型不需要一致
- 列表是一个可变的对象，我们可以改变它的数
- 用中括号定义列表，列表的元素用逗号隔开。例如， [1, 2, 's']
- 可以通过索引来访问元素，第一个元素index是0，最后一个元素的index是-1
- 与字符串相似，列表也支持分片（[:]），连接（+），重复(*)和in操作

List

```
>>> l = ['physics', 'chemistry', 2017, 2019]
>>> print(l[0])
physics
>>> print(l[1:3])
['chemistry', 2017]
>>> l[2] = 'english'
>>> print(l)
['physics', 'chemistry', 'english', 2019]
>>> del l[0]
>>> print(l)
['chemistry', 'english', 2019]
>>>
```

List

- 常用方法

list.append(obj)	将obj添加到list的尾部
list.insert(index, obj)	将obj插入到list的index位置
list.count(obj)	统计列表中obj出现的次数
list.remove(obj)	从列表中删除obj
len(list)	统计list的长度
del list[index]	删除列表中的元素

List递推式构造列表

递推式构造列表由一个for关键字构成

```
>>> list1 = [1, 2, 3]
>>> list2 = [x ** 2 for x in list1]
>>> list2
[1, 4, 9]
>>> list3 = [x + 1 for x in [ x** 2 for x in list1 ]]
>>> list3
[2, 5, 10]
>>>
```

元组

- 元组是一个不可变的对象
- 元祖同样支持切片功能
- 使用小括号创建元组，元组的元素用逗号隔开

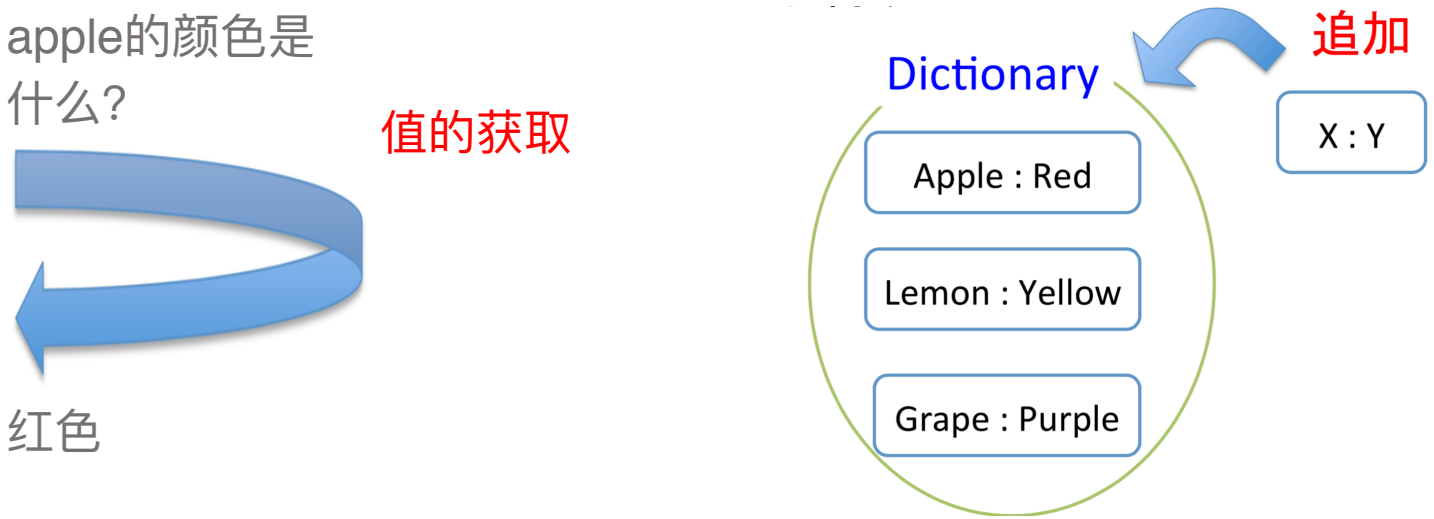
```
>>> t = ('tuples', 'are', 'Immutable')
>>> print('Access the element of tuples')
Access the element of tuples
>>> t[1]
'are'
>>> t[1] = 'is'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

- 如果创建一个元素的元组，也必须用一个逗号结尾

```
>>> a = (2)
>>> type(a)
<class 'int'>
>>> a = (2,)
>>> type(a)
<class 'tuple'>
```

字典

- Python中的字典是key-value的元素集合，并且是一个无序的结合
  - key：必须是不可以改变的数据类型，通常是数字或者字符串
  - Value：可以是任意的python类型
- 字典是一个可以改变的数据类型，可以添加新的元素与改变字典中的value
- Python的字典用大括号进行包裹，元素用逗号隔开，键与值之间用冒号(:)隔开
- 访问字典元素的时候使用 dict_name[key_name]的方式进行访问



字典

- 常用方法

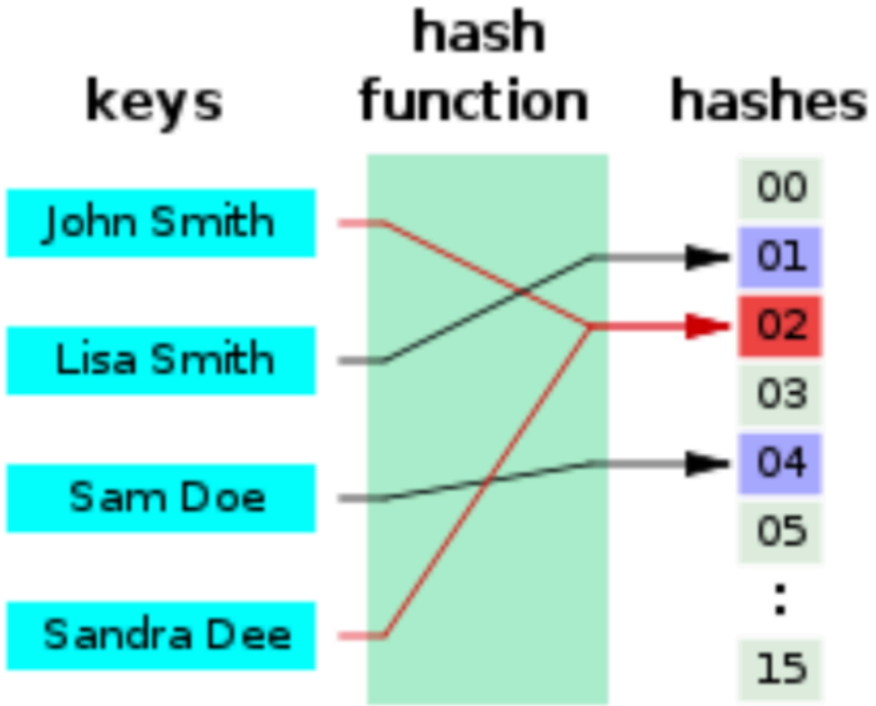
dict.keys()	以list的形式返回字典中的所有key
dict.values()	以list的形式返回字典中的所有value
dict.items()	以list的形式返回字典中的所有键值对，键值对以tuple的形式返回
dict.has_key()	如果字典存在key，则返回True，否则返回False
dict.update(dict2)	将dict2添加到dict中
dict.clear()	删除字典中的所有元素
len(dict)	获得字典的长度

例子

```
>>> my_dict = {'Name': 'Tome', 'Age': 9, 'Grade': '5th'}
>>> my_dict['Name']
'Tome'
>>> my_dict['Age']
9
>>> my_dict.keys()
dict_keys(['Name', 'Age', 'Grade'])
>>> my_dict.values()
dict_values(['Tome', 9, '5th'])
>>> my_dict.items()
dict_items([('Name', 'Tome'), ('Age', 9), ('Grade', '5th')])
>>> my_dict['Age'] = 10
>>> my_dict['School'] = 'School'
>>> my_dict
{'Name': 'Tome', 'Age': 10, 'Grade': '5th', 'School': 'School'}
>>> del my_dict['Name']
>>> my_dict
{'Age': 10, 'Grade': '5th', 'School': 'School'}
>>> my_dict.clear()
>>> my_dict
{}
>>> del my_dict['Age']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'Age'
```

字典与列表的不同

- 列表是以线性的顺序来保存数据
- 字典是采用hash的方式来保存数据
- 在数据量大的时候，字典会快于列表



集合（Set）

- set是无序的，其元素是一个无重复、可以哈希（元素为不可变对象）的集合。类似一个没有value的字典
- 因为是无序的，所以不支持索引和切片功能
- 创建方式：
    1. 使用关键字set, set_obj = set（序列对象）
    2. 使用大括号，元素之间用逗号隔开
    3. 空的集合只能用set()来创建，因为{}创建的是空的字典

集合-例子

```
>>> s_a = set()
>>> print(s_a, type(s_a))
set() <class 'set'>
>>> d = {}
>>> print(d, type(d))
{} <class 'dict'>
>>> s_b = {1, 2, '3'}
>>> print(s_b, type(s_b))
{1, 2, '3'} <class 'set'>
>>>
>>>
>>>
>>>
>>>
>>> s_c = set('boy')
>>> print(s_c, type(s_c))
{'o', 'y', 'b'} <class 'set'>
>>> s_d = set(['a', 'b', 1, 3])
>>> print(s_d, type(s_d))
{1, 3, 'a', 'b'} <class 'set'>
>>> s_e = set(('a', 'b', 1, 3))
>>> print(s_e, type(s_e))
{1, 3, 'a', 'b'} <class 'set'>
```

集合（Set）

- 常见的set操作
  - 比较（集合的减法）
    - set_1.difference(set_2)，找到set_1中存在，但是set_2中不存在的元素。将结果赋予新的变量
    - set_1.difference_update(set_2)，找到set_1中存在，但是set_2中不存在的元素。更新自己
  - 删除
    - set_obj.discard(val) 移除不存在元素不会报错
    - set_obj.remove(val) 移除不存在元素会报错
    - set_obj.pop(val) 移除末尾元素并赋给新的变量

```
>>> s_1 = {1, 2, 3}
>>> s_2 = {4, 5}
>>> temp_1 = s_1.difference(s_2)
>>> temp_1
{1, 2, 3}
>>> s_1
{1, 2, 3}
>>> temp_1 = s_1.difference_update(s_2)
>>> print(temp_1)
None
>>> s_1
{1, 2, 3}
>>>
```

```
>>> s_1.discard(11)
>>> s_1
{1, 2, 3}
>>> s_1.discard(1)
>>> s_1
{2, 3}
>>> s_1.remove(11)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 11
>>> temp_1 = s_1.pop()
>>> s_1
{3}
>>> temp
```

集合（Set）

- 常见的set操作
  - 取交集
    - set_1.intersection(set_2) 取交集赋给新的变量
    - set_1.intersection_update(set_2) 取交集改变自己
  - 取并集
    - set_1.union(set_2) 取并集赋予新的变量

```
>>> s_1 = {1, 2, 3}
>>> s_2 = {2, 3, 4, 5}
>>> temp_1 = s_1.intersection(s_2)
>>> temp_1
{2, 3}
>>> s_1
{1, 2, 3}
>>> temp_1 = s_1.intersection_update(s_2)
>>> temp_1
>>> s_1
{2, 3}
```

```
>>> s_1 = {1, 2, 3}
>>> s_2 = {2, 3, 4, 5}
>>> temp = s_1.union(s_2)
>>> temp
{1, 2, 3, 4, 5}
>>> temp = s_1.union_update(s_2)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'set' object has no attribute 'union_update'
```

None

- 表示一个空对象，不是0，也不是False，也不是空字符串
- 有自己类型NoneType， NoneType只有一个值是None
- 与C，Java中的NULL类似

Bool类型

- 不二类型只有两个值，True与False （注意大小写）
- 常用于if与while等控制语句中

变量

- 变量：是数据保存的容器
- Python 不需要声明变量的数据类型

```
int x = 5;
x = "Java" //Error
```

Java

```
x = 5;
print(x)
x = "Python" # OK
print(x)
```

Python