



人工智能之机器学习

零基础入门学习Python

产品研发中心 -- 李军



凯通科技

www.ctt-net.com

用软件重新定义世界，让世界更加智能互联

- ❑ Python是一种面向对象的解释型语言，底层是C语言编写，由荷兰人Guido van Rossum于1989 年发明，第一个公开发行版于1991年发布。
- ❑ Python语法简洁清晰，强制用空白符(white space)作为语句缩进。
- ❑ 共享Python开发的功能必须提供源码本身，可使用py2exe等包转换成系统执行的文件。
- ❑ 最大特点：简单、强大
- ❑ Python具有丰富的开源库 XGBoost/TensorFlow

尝试一点新的东西：

```
>>> print(5+3)
```

```
>>> 5+3
```

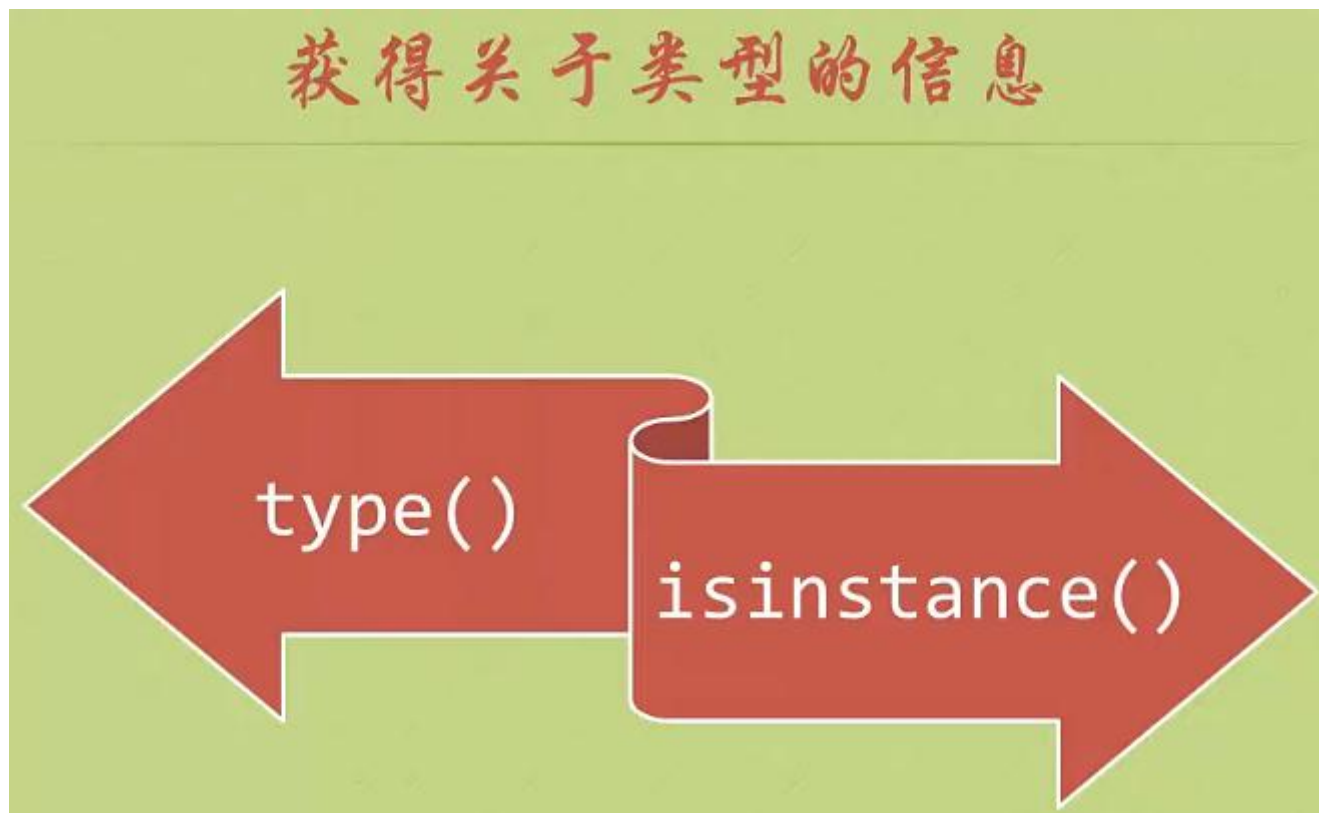
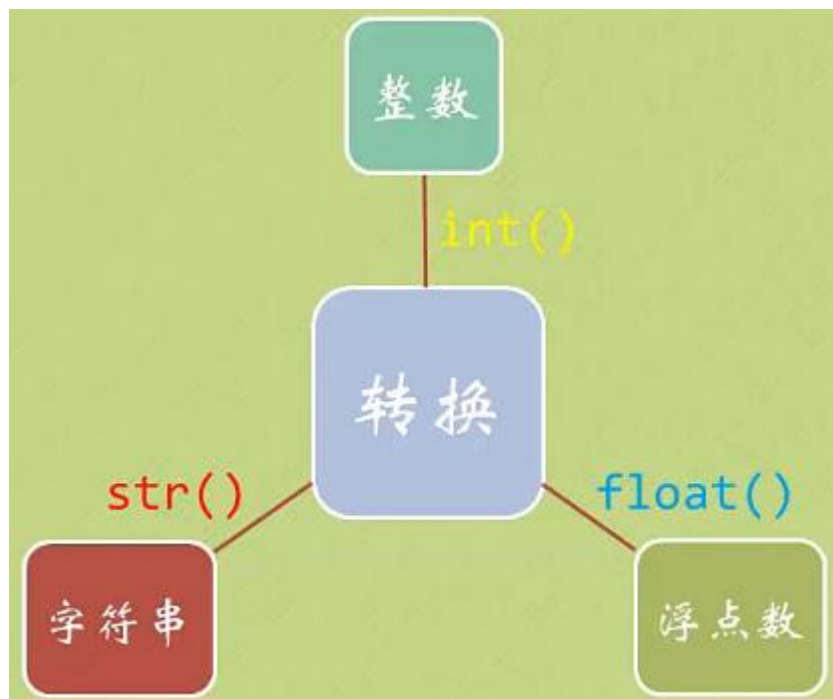
```
>>> 1234567890987654321 * 9876543210123456789
```

```
>>> print( "catt" + "soft" )
```

```
>>> print( "catt" + "soft" * 8)
```

- Pip
 - 安装Python包的推荐工具：<https://pypi.python.org/pypi/pip>
 - 更换国内源：pip install -i <https://pypi.tuna.tsinghua.edu.cn/simple> numpy
- Numpy：
- Pandas：Python Data Analysis Library
 - 在Numpy基础上提供了更多的数据读写工具
- Scipy
 - 在NumPy基础上添加了众多科学计算工具包
- Matplotlib
 - Python丰富的绘图库
- 官网：
 - Numpy/Scipy：<http://www.scipy.org>
 - Pandas：<http://pandas.pydata.org/>
 - Matplotlib：<http://www.matplotlib.org>
- Sklearn
 - 机器学习算法库
- Tensorflow
 - 深度学习库

- 数值类型：整形、浮点型、布尔型（True，False可运算）、E记法（15000 = 1.5E4）
- 字符串类型
- 复合类型



算术操作符



```
>>>a = 3
```

```
>>>a+=3
```

```
>>>a,b = 1,2
```

```
>>>a=b=c=3
```

```
>>>3/2
```

```
>>>1.5 #真正的除法运算
```

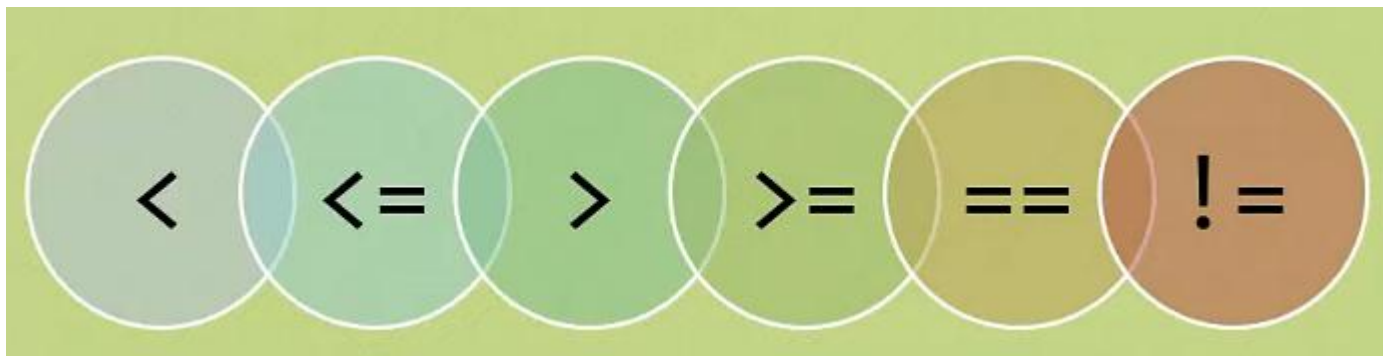
```
>>>3//2
```

```
>>>1 #取整除法
```

```
>>>3 ** 3
```

```
>>>27 #表示3的3次幂
```

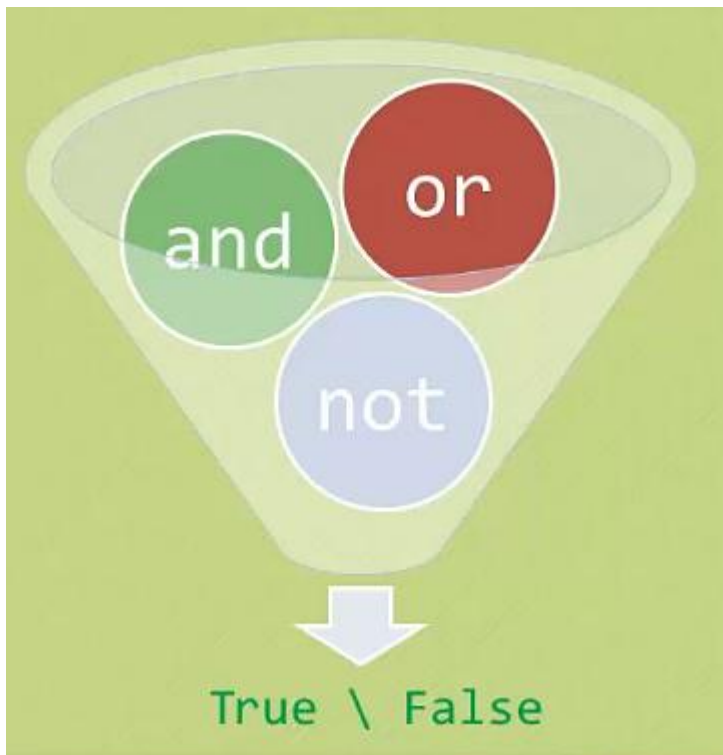
Python比较操作符与逻辑操作符



```
>>>3<2
```

```
>>>3==3
```

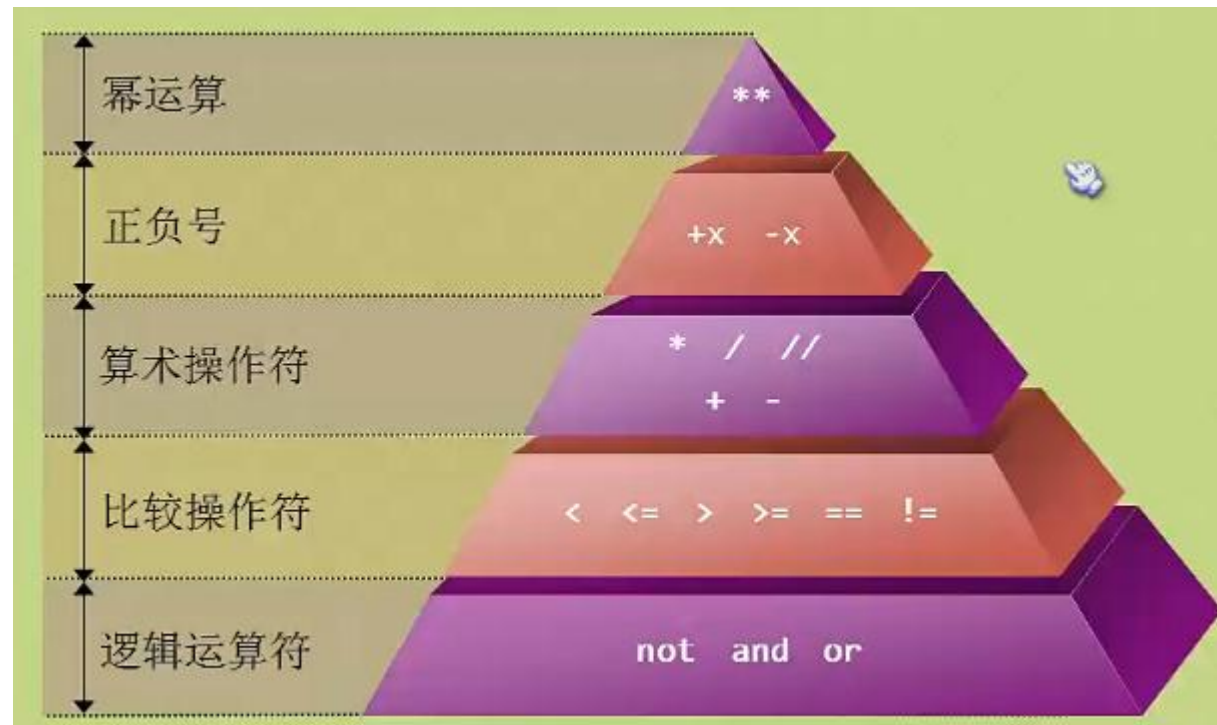
```
>>>3>1
```



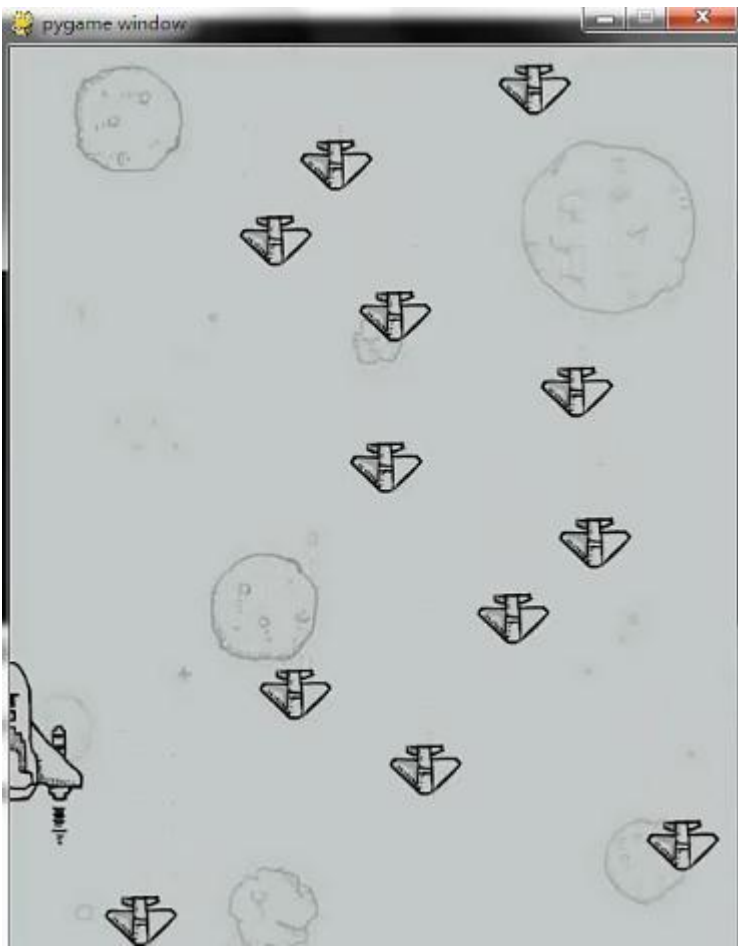
```
>>>not True
```

```
>>>not 0
```

```
>>>3<4<5
```



Python分支与循环



```
i = 1
```

```
while True:
```

```
    if i > 10:
```

```
        break
```

```
    print( 'aaaaaaaaaaaaaaaa' , i)
```

```
    i += 1
```

```
score = int(input('请输入一个分数: '))
if 100 >= score >= 90:
    print('A')
elif 90 > score >= 80:
    print('B')
elif 80 > score >= 60:
    print('C')
elif 60 > score >= 0:
    print('D')
else:
    print('输入错误!')
```

- 有了这个三元操作符的条件表达式，你可以使用一条语句来完成以下的条件判断和赋值操作：

```
x, y = 4, 5
```

```
if x < y:
```

```
    small = x
```

```
else:
```

```
    small = y
```

- 例子可以改进为：

```
small = x if x < y else y
```

for 循环

- 虽然说Python是由C语言编写而来的，但是Ta的for循环跟C语言的for循环不太一样，Python的for循环显得更为智能和强大！

```
>>> for i in range(1, 9, 2):  
>>>     print(i)
```

- 语法：

for 目标 in 表达式:
 循环体

```
>>> s = 'cattsoft'  
>>> for i in s:  
>>>     print(i, end= ' ')
```

断言 (assert)

- assert这个关键字我们称之为“断言”，当这个关键字后边的条件为假的时候，程序自动崩溃并抛出AssertionError的异常。
- 举个例子：

```
>>> assert 3 > 4
```
- 一般来说我们可以用Ta再程序中置入检查点，当需要确保程序中的某个条件一定为真才能让程序正常工作的话，assert关键字就非常有用。

列表：一个打了激素的数组




整数

浮点数

字符串

对象

```
>>>list1 = [ '乔峰' , '段誉' , '虚竹' ]
>>>list2 = [ '乔峰' , 3 , 1.2 , True , [1,2,3]]
>>>len(list2)
>>>list2.append( '凯通' )
>>>list2.extend([ '周伯通' , '洪七公' ])
>>>list2.insert(0, '黄药师' )
>>>list2.remove( '乔峰' )
>>>del list2[1] #del内置函数，会清空内存
>>>name = list2.pop() #默认从栈取最后一个元素
>>>name = list2.pop(1)
>>>list3 = list2[0:3] #切片生成一个新的列表
>>>list3 = list2[:3] 或 list2[:]
>>>list2 + list3
>>>list2 * 2
>>>'乔峰' in list2
>>>list2.count( '乔峰' ) #返回 '乔峰' 的个数
>>>list2.sort() #默认是升序，归并排序
>>>list2.sort(reverse=True) #降序
```

- 由于和列表是近亲关系，所以元组和列表在实际使用上是非常相似的。
- 我们这节课主要通过讨论元组和列表到底有什么不同来学习元组，酱紫大家就不会觉得老是重复一样的内容。
- 我们主要从以下几个点来讨论学习：
 - 创建和访问一个元组 
 - 更新和删除一个元组

```
>>>p1 = (1, 2, 'a' )
>>>p1[1]
>>>p2 = p1[:]
>>>p1 = 1, 2, 'a'
>>>p1 = ()
>>>p1 = (1,) #若p1 = (1)则不是元组而是整型
>>>p1 * 2
>>>p1 = p1[:2] + (2,) + p1[2:]
>>>del p1 #通常不需要显示删除对象，有垃圾回收机制
```

Python字符串



capitalize()	把字符串的第一个字符改为大写
casefold()	把整个字符串的所有字符改为小写
center(width)	将字符串居中，并使用空格填充至长度width的新字符串
count(sub[,start[,end]])	返回sub在字符串里边出现的次数，start和end参数表示范围，可选。
encode(encoding='utf-8', errors='strict')	以encoding指定的编码格式对字符串进行编码。
endswith(sub[,start[,end]])	检查字符串是否以sub子字符串结束，如果是返回True，否则返回False。start和end参数表示范围，可选。
expandtabs([tabsize=8])	把字符串中的tab符号（\t）转换为空格，如不指定参数，默认的空格数是tabsize=8。
find(sub[,start[,end]])	检测sub是否包含在字符串中end参数表示范围，可选。
index(sub[,start[,end]])	跟find方法一样，不过如果su

join(sub)	以字符串作为分隔符，插入到sub中所有的字符之间。
ljust(width)	返回一个左对齐的字符串，并使用空格填充至长度为width的新字符串。
lower()	转换字符串中所有大写字母为小写。
lstrip()	去掉字符串左边的所有空格
partition(sub)	找到子字符串sub，把字符串分成一个3元组（pre_sub,sub,fol_sub），如果字符串中不包含sub则返回（'原字符串'，' '，' '）
replace(old,new[,count])	把字符串中的old子字符串替换成new子字符串，如果count指定，则替换不超过count次。
rfind(sub[,start[,end]])	类似于find()方法，不过是从右边开始查找。

isalnum()	如果字符串至少有一个字符并且所有字符都是字母或数字则返回True，否则返回False。
isalpha()	如果字符串至少有一个字符并且所有字符都是字母则返回True，否则返回False。
isdecimal()	如果字符串只包含十进制数字则返回True，否则返回False。
isdigit()	如果字符串只包含数字则返回True，否则返回False。
islower()	如果字符串中至少包含一个区分大小写的字符，并且这些字符都是小写，则返回True，否则返回False。
isnumeric()	如果字符串中只包含数字字符，则返回True，否则返回False。
isspace()	如果字符串中只包含空格，则返回True，否则返回False。

所有的单词都是以大写开始，其余字母均小写），
alse。

一个区分大小写的字符，并且这些字符都是大写，
alse。

序列！序列！

列表、元组和字符串的共同点

- 都可以通过索引得到每一个元素
- 默认索引值总是从0开始
- 可以通过分片的方法得到一个范围内的元素的集合
- 有很多共同的操作符（重复操作符、拼接操作符、成员关系操作符）

`list()` 把一个可迭代对象转换为列表

```
>>> s1 = 'cattsoft'
>>> list1 = list(s1)
>>> t1 = (9,8,1,2,3,4,5)
>>> list1 = list(t1)
>>> len(list1)
>>> max(list1)
>>> min(list1)
>>> sum(list1)
>>> sorted(list1)
>>> sorted(reversed(list1))
>>> l1, l2, l3 = (1,2,3), (4,5,6), (7,8,9,10)
>>> l4 = list(zip(l1, l2, l3))
>>> for (x, y, z) in l4:
>>>     print(x, y, z)
>>> list3 = list(input())
```

Python函数



```
>>> def MyFirstFunction():  
    print("这是我创建的第一个函数!")  
    print("我表示很鸡冻.....")  
    print("在此我要感谢TVB, 感谢CCAV, 感谢小甲鱼  
老湿, 感谢各位鱼油.....")
```

```
>>> MyFirstFunction()  
这是我创建的第一个函数!  
我表示很鸡冻.....  
在此我要感谢TVB, 感谢CCAV, 感谢小甲鱼老湿, 感谢各位  
鱼油.....
```

```
>>> def MyFirstFunction(name):  
    '函数定义过程中的name是叫形参'  
    #因为Ta只是一个形式, 表示占据一个参数位置  
    print('传递进来的' + name + '叫做实参, 因为  
Ta是具体的参数值!')
```

```
>>> MyFirstFunction('小甲鱼')  
传递进来的小甲鱼叫做实参, 因为Ta是具体的参数值!
```

```
def add(num1, num2):  
    return num1 + num2
```

```
>>> add(1, 3)  
>>> add(num1=3, num2=2)  
>>> MyFirstFunction.__doc__  #函数文档
```

```
def add(num1=1, num2=2):  
    return num1 + num2
```

```
def test(*param):  
    print(len(param))  
    print(param[1])
```

```
>>> test(1,2,3, 'a')
```

```
def back():  
    return 1, 'catt', 3.14
```


变量作用域

```
count = 5
def MyFun():
    count = 10
    print(count)
```

```
>>>MyFun()
10
>>>print(count)
5
```

```
count = 5
def MyFun():
    global count
    count = 10
    print(count)
```

内嵌函数

```
def funX():
    x = 3
    def funY():
        nonlocal x
        x *= x
        return x
    return funY()
```

递归函数

```
def dg(n): #求阶乘
    if n == 1:
        return 1
    else:
        return n * dg(n-1)
>>>num = int(input( '请输入...' ))
>>>result = gd(num)
```

闭包函数

```
def funX(x):
    def funY(y):
        return x * y
    return funY
```

```
>>>i = funX(3)
>>>type(i)
>>>i(5)
>>>funX(3)(5)
```

Lambda表达式 (左 : 参数 , 右 : 返回值)

```
>>>g = lambda x, y : (x + y) * 2
>>>g(3, 2)
```

```
>>> dict1 = {'李宁': '一切皆有可能', '耐克': 'Just do it', '阿迪达斯': 'Impossible is nothing', '鱼c工作室': '让编程改变世界'}
>>> print('鱼c工作室的口号是:', dict1['鱼c工作室'])
鱼c工作室的口号是: 让编程改变世界
```

```
>>> dict2 = {1: 'one', 2: 'two', 3: 'three'}
>>> dict2
{1: 'one', 2: 'two', 3: 'three'}
>>> dict2[2]
'two'
```

```
>>> for eachKey in dict1.keys():
    print(eachKey)
```

```
>>> for eachValue in dict1.values():
    print(eachValue)
```

```
>>> for eachItem in dict1.items():
    print(eachItem)
```

```
>>> print(dict1[31])
赞
>>> print(dict1[32])
Traceback (most recent call last):
  File "<pyshell#17>", line 1, in <module>
    print(dict1[32])
KeyError: 32
>>> dict1.get(32)
None
>>> print(dict1.get(32))
None
>>> dict1.get(32, '木有!')
'木有!'
>>> dict1.get(31, '木有!')
'赞'
>>> 31 in dict1
True
>>> 32 in dict1
False
```

```
>>> a = {1: 'one', 2: 'two', 3: 'three'}
>>> b = a.copy()
>>> c = a
>>> c
{1: 'one', 2: 'two', 3: 'three'}
>>> a
{1: 'one', 2: 'two', 3: 'three'}
>>> b
{1: 'one', 2: 'two', 3: 'three'}
>>> id(a)
47496840
>>> id(b)
48239624
>>> id(c)
47496840
```

```
>>> a.pop(2)
'two'
>>> a
{1: 'one', 3: 'three', 4: 'four'}
>>> a.popitem()
(1, 'one')
>>> a
{3: 'three', 4: 'four'}
>>> a.setdefault('小白')
None
>>> a
{'小白': None, 3: 'three', 4: 'four'}
>>> a.setdefault(5, 'five')
'five'
>>> a
{3: 'three', 4: 'four', 5: 'five', '小白': None}
```

```
>>>S1 = {1,2,3,4,5,6,2,3}
```

```
#去掉重复元素
```

```
>>>S1.add(7)
```

```
>>>S1.remove(1)
```

```
>>>S1.pop() #先进先出
```

```
>>>for I in S1:
```

```
>>>    print(I)
```

封装

```
class Turtle: # Python 中的类名约定以大写字母开头
    """关于类的一个简单例子"""
    # 属性
    color = 'green'
    weight = 10
    legs = 4
    shell = True
    mouth = '大嘴'

    # 方法
    def climb(self):
        print("我正在很努力的向前爬.....")

    def run(self):
        print("我正在飞快的向前跑.....")

    def bite(self):
        print("咬死你咬死你!!")

    def eat(self):
        print("有得吃, 真满足^_^")

    def sleep(self):
        print("困了, 睡了, 晚安, Zzzz")
```

继承与多继承

```
>>> class MyList(list):
    pass

>>> list2 = MyList()
>>> list2.append(5)
>>> list2.append(3)
>>> list2.append(7)
>>> list2
[5, 3, 7]
>>> list2.sort()
>>> list2
[3, 5, 7]
```

构造函数 __init__

```
>>> class Ball:
    def __init__(self, name):
        self.name = name
    def kick(self):
        print("我叫%s, 该死的, 谁踢我..." % self.name)

>>> b = Ball('土豆')
>>> b.kick()
我叫土豆, 该死的, 谁踢我...
```

私有

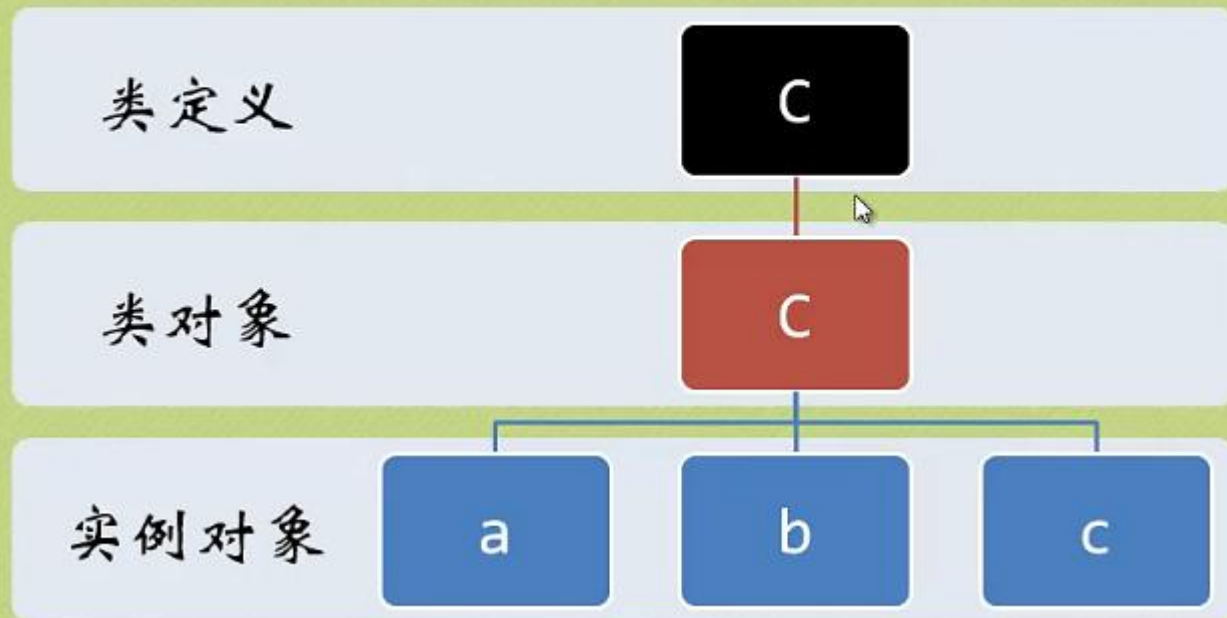
```
>>> p = Person()
>>> p.name
'小甲鱼'
>>> class Person:
    __name = "小甲鱼"
```

```
>>> p = Person()
>>> p.__name
Traceback (most recent call last):
  File "<pyshell#31>", line 1, in <module>
    p.__name
AttributeError: 'Person' object has no attribute '__name'
>>> p.name
Traceback (most recent call last):
  File "<pyshell#32>", line 1, in <module>
    p.name
AttributeError: 'Person' object has no attribute 'name'
>>> class Person:
    __name = "小甲鱼"
    def getName(self):
        return self.__name
```

伪私有

```
>>> p = Person()
>>> p._Person__name
'小甲鱼'
Traceback (most recent call last):
  File "<pyshell#38>", line 1, in <module>
    p._name
AttributeError: 'Person' object has no attribute '_name'
>>> p.getName()
'小甲鱼'
```

类、类对象和实例对象



```
class Test:
```

```
    count = 0
```

```
>>>a = Test()
```

```
>>>b = Test()
```

```
>>>a.count += 10 #仅修改实例变量
```

```
>>>Test.count += 5 #修改所有实例变量
```


导入模块

- 第一种: `import 模块名`
- 第二种: `from 模块名 import 函数名`

hello.py

```
def myFun():  
    print 'hello word'
```

test.py

```
import hello as h  
h.myFun()  
或  
from hello import myFun  
myFun()
```

```
from multiprocessing import Process
import os
import time

# 子进程要执行的代码
def my_proc(name):
    print('Run child process %s (%s)...' % (name,
os.getpid()))

if __name__ == '__main__':
    print('Parent process %s.' % os.getpid())
    p = Process(target=my_proc, args=('test',))
    print('Child process will start.')
    p.start()
    time.sleep(2)
    print('Child process end.')
```

```
from multiprocessing import Pool
import os, time, random

def long_time_task(name):
    print('Run task %s (%s)...' % (name, os.getpid()))
    start = time.time()
    time.sleep(random.random() * 3)
    end = time.time()
    print('Task %s runs %0.2f seconds.' % (name, (end - start)))

if __name__ == '__main__':
    print('Parent process %s.' % os.getpid())
    p = Pool(4)
    for i in range(5):
        p.apply_async(long_time_task, args=(i,))
    print('Waiting for all subprocesses done...')
    p.close()
    p.join()
    print('All subprocesses done.')
```



感谢您的聆听!

Thank you for your time!



凯通科技

www.ctt-net.com

用软件重新定义世界，让世界更加智能互联