本篇主要是 Python 的基本语法，重在应用，进阶版的语法细节请看下一篇博客.

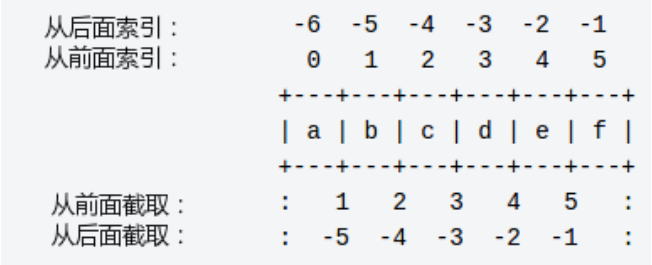
变量类型:

5 个数据类型：数字，字符串，列表，元组，字典.

数字类型(Numbers): int, float, complex(复数)

字符串(string)：可用单引号''和双引号""创建字符串，两者等价，没有 char 类型

列表(list)：索引方式为[], [start:step:end]，索引范围为[start,end), end 的下标不取到. 也可以从-1 开始，倒序索引的方式也很方便.



元组(tuple)：元组不能修改，用括号()构建, tuple = (1, 'aa','v')，用[]索引

字典(map)：字典是无序的，但是元素唯一，通过 key 索引 value. mp[key]=value

数据结构之间的**类型转换**:

```
x = 3.14
int(x);bin(3);hex(3);#转化为 10,2,16 进制
ord('a'); chr(97) #字符串的 ascii 码相互转化
str(x)#字符串转化数字
eval('1+1') #计算字符串的值
set(x) #转化为集合，元素的唯一化
dict((1,'value')) #字典必须传入 tuple 才能存储
```

操作符

算术运算符:

运算符	描述	实例
+	加　两个对象相加	a + b 输出结果 30
-	减　得到负数或是一个数减去另一个数	a - b 输出结果 -10
*	乘　两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果 200
/	除　x 除以 y	b / a 输出结果 2

%	取模　返回除法的余数	b % a 输出结果 0
**	幂　返回 x 的 y 次幂	2**3 = 8
//	取整除 - 返回商的整数部分（**向下取整**）	9//2 = 4; -9//2 = -5

比较运算符:

运算符	描述	实例
==	等于　比较对象是否相等	(a == b) 返回 False。
!=	不等于　比较两个对象是否不相等	(a != b) 返回 true.
<>	不等于　比较两个对象是否不相等	这个运算符类似 !=
>	大于　返回 x 是否大于 y	(a > b) 返回 False。
<	小于　返回 x 是否小于 y。所有比较运算符返回 1 表示真，返回 0 表示假。	(a < b) 返回 true。
>=	大于等于　返回 x 是否大于等于 y。	(a >= b) 返回 False。
<=	小于等于　返回 x 是否小于等于 y。	(a <= b) 返回 true。

赋值运算符:

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c *= a 等效于 c = c * a

/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c ** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

逻辑运算符:

运算符	逻辑表达式	描述	实例
and	x and y	布尔"与" - 如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。	(a and b) 返回 20。
or	x or y	布尔"或" - 如果 x 是非 0，它返回 x 的值，否则它返回 y 的计算值。	(a or b) 返回 10。
not	not x	布尔"非" - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。	not(a and b) 返回 False

成员运算符:

运算符	描述	实例
in	如果在指定的序列中找到值返回 True，否则返回 False。	x 在 y 序列中 ，如果 x 在 y 序列中返回 True。
not in	如果在指定的序列中没有找到值返回 True，否则返回 False。	x 不在 y 序列中 ，如果 x 不在 y 序列中返回 True。

位运算符:

运算符	描述	实例
&	按位与运算符：如果两个相应位都为 1,则该位的结果为 1,否则为 0	(0b110 & 0b101) 输出结果 0b100
|	按位或运算符：只要对应的二个二进位有一个为 1 时，结果位就为 1。	(0b110 | 0b101) 输出结果 0b111
^	按位异或运算符：当两对应的二进位相异时，结果为 1	(0b110 ^ 0b101) 输出结果 0b011
~	按位取反运算符：对每个二进制位取反,即把 1 变为 0,把 0 变为 1 。	在一个有符号二进制数的补码形式。~x 类似于 -x-1，补码 = 反码 + 1
<<	左移动运算符：由 << 右边的数字指定了移动的位数，高位丢弃，低位补 0。	1 << 2 输出结果 4
>>	右移动运算符：把">>"左边的运算数的各二进位全部右移若干位	3 >> 1 = 1

for while if 语句:

对于 for 和 while 循环，还可用 pass(空操作), continue(跳过单次循环), break(跳过全部循环)

```
x = 3
#缩进 Tap 就是 Python 的灵魂，缩进发挥了 C 语言的{}作用，限制作用域
if x < 2:
    print("x<2")
else:
    print("x>2")
for i in range(x):
    print(i)
while x > 0:
    print(x); x -= 1; #python 不能用 x--,
    #一行多个语句需要用;隔开，另外单行太长可以用\ 隔开
```

输出格式化:

python 输出格式化有两种重要方式: %s 与 format 函数

```
print("%03x"%a) #输出 16 进制的数字，占 3 位，不足补零
print("\n%2.2f"%11.3) #输出 2 位 2 小数的浮点数
```

format 大法好

```
a = 30
b = 10.265
#format 用{}来确定格式化输出的位置
print("first is {}, second is {} ".format(a,b))
print(" b is {1}, a is {0} ".format(a,b)) #还可以变换顺序
print(" {1:2.4f}, {0} ".format(a,b)) #利用:确定格式. 冒号:前表示 format 里面 tuple 的原数顺序
print(" {0:#x}, {0} ".format(a,b)) #d 十进制,x16 进制, #x 输出带前缀 0x
print("{:.2%}".format(0.9)) # 还可以输出%制
```

自定义函数

```
def myfunc(a): #不用定义参数类型，这是 Python 的一个特点
    return a+4
print(myfunc(4))
```

Python 中函数的参数都是引用，除了 list 类型是传入指针，因此 list 在函数内修改也会传递到外部，并没有像 C++中&引用的操作符，仅能通过 list 来传递引用参数.

```
#不定长函数参数
def myfunc2(*var):
    for i in var:
        print(i)
myfunc2(1,2,3)
```

全局变量也是可以直接在函数内索引，但是仍然是传递副本，不会改变外部的值.

匿名函数很好用: 传入参数和 return 结果，最好能够在一行内写完，省得占用空间.

```
sum = lambda arg1, arg2: arg1+arg2
print(sum(1,3))
```

class 类:

类(Class)算是面向对象编程(OOP)的一个重要特性, Python 本身可以是过程性编程也可以是面向对象编程，反正就是怎么方便怎么来.

Class 有几个专业术语，请看下面的语句:

```
class Circle:
    pi = 3.14 #类属性，全体实例共有的属性
    def __init__(self,r): #初始化方法函数
        self.r = r #实例属性，每个实例的属性不一定相同
    def add(self,r1,r2): #类方法，实例可以调用的函数
        self.r = r1 + r2
circle1 = Circle(3) #构建一个实例，初始化半径为 3
```

_foo 开头的方法和属性是伪私有，实例在外部可以调用，但约定以下划线开头的类不直接调用.

__foo 双下划线开头的方法和属性是类的私有成员，**不能**直接调用，会报错.

__foo__双下划线开头和结尾的方法是特殊方法. 特殊方法具体请看下一篇博客.

析构函数和作用域的问题:

在实例创建时, Class 会调用 __inti__方法来创建实例，当没有变量指向该实例时，系统的垃圾回收机制会调用析构函数删除该实例，析构函数也可以自定义: 同时__del__也是特殊方法，可以通过特殊语法调动, tom.del() 或是 del tom 都可以.

```
class Person:
    def __init__(self,name):#初始化方法
        self.name = name
    def __del__(self):#析构方法（销毁方法）
        print(self.name,'被销毁了...')
tom = Person('jack')
del tom#打印出被销毁
```

Class 的继承等细节可参考下一篇博客.

文件 IO

再加一个文件 IO 即可.

其中覆盖的意思表示如果源文件存在，则清空源文件，所以最好是使用 a+

模式	r	r+	w	w+	a	a+
读	+	+		+		+
写		+	+	+	+	+
创建			+	+	+	+
覆盖			+	+		
指针在开始	+	+	+	+		
指针在结尾					+	+

代码参考如下:

```
myfile = open("a.txt","w+") #文件路径和读写模式
#开始写
# myfile.write("hello world \n")
print(myfile.read(5)) #读取 5 个字节
position = myfile.tell() #当前读取的位置
myfile.seek(0,0) ##表示从头开始
myfile.close()
```

删除与重命名

```
import os
os.rename("a.txt","b.txt") #很简单的用法 重命名文件
os.remove("b.txt") #删除文件
os.getcwd() #的带当前文件夹
os.chdir("fisrt") #更改当前目录
```

参考文档: <Head first Python>，菜鸟基础教程—Python