一、python数据类型

**1、数字**

|  |  |
| --- | --- |
| int | 有符号整数(可表示最大值与物理内存有关)  0b或0B开头表示2进制数  0o或0O开头表示8进制数  0x或0X开头表示16进制数 |
| bool | 布尔值可以理解为特殊的整数，True:1/False:0 |
| float | 浮点数 |
| complex | 复数 |

1. **字符串**

1）在python中，单双引号没有任何区别

2）字符串中如果存在未确定字符，可使用占位符

>>> '%s is %s years old' % ('bob', 20)

'bob is 20 years old'

3）支持三引号，用来包含特殊字符

>>> var = '''hello

... world !'''

1. 字符串常用操作

>>> py\_str = 'python'

>>> len(py\_str) #返回6

>>> 'to' in py\_str #返回False

>>> 'to' not in py\_str #返回True

>>> py\_str[0] #返回'p'

>>> py\_str[-1] #返回'n'

>>> py\_str[2:4] #返回'th'，字符串切片(左闭右开区间)

>>> py\_str[1:4:2] #返回'hn'，2表示步长值

>>> py\_str[::-1] #返回'nohtyp'，步长值为负，表示自右向左取

5）字符串连接操作

>>> py\_str + ' ' + 'is cool' #返回'python is cool'

>>> py\_str \* 2 #返回'pythonpython'

1. **列表**

把任意类型数据集中存储到一个容器中，相当于"数组"

1）列表常用操作：

>>> alist = [10, 20, 'bob', 'alice', [1,2]]

>>> len(alist) #返回5

>>> alist[4][0] #返回1

>>> alist[-1] = 100 #修改列表最后一项

>>> alist.append('tom') #使用append方法追加一项

... ...与字符串的操作一致

**4、元组**

1）定义元组

>>> atuple = (1, "tom", 2, "alice") #使用()定义元组，不可修改

**5、字典**

{key: val}，key不能重复，val可以相同

1. 字典常用操作：

>>> adict = {'name': 'bob', 'age': 20}

>>> len(adict) #返回2

>>> 20 in adict #返回False

>>> 'age' in adict #返回True

>>> adict[1] #KeyError: 1报错

>>> adict['age'] #返回20

1. **数据类型比较**

1）按存储模型分类

标量类型：数值、字符串

容器类型：列表、元组、字典

python

[20, 'bob', 10]

**内存**

py\_str

alist

2）按更新模型分类:

可变类型：列表、字典

不可变类型(可重新定义)：数字、字符串、元组

3）按访问模型分类

直接访问：数字

顺序访问：字符串、列表、元组

映射访问：字典

二、判断语句

**1、if语句**

1）数据类型当作判断条件

-非零数字为True，否则为False

-空字符串、空列表、空元组、空字典、None的值均为False

2）案例：判断合法用户

import getpass #密文获取密码

var\_user = input('input your username:')

var\_pass = getpass.getpass('input your password:')

if var\_user == 'bob' and var\_pass == '123456':

print('Login stuccessful')

else:

print('Login inorrect')

**2、扩展if语句**

1）三元运算符

>>> x,y = 10, 20

>>> small = x if x < y else y

>>> print(small) #输出10

2）案例1：猜数

import random

number = random.randint(1,10)

print(number)

answer = int(input('your answer:'))

if answer > number:

print('\033[5;31m猜大了')

elif answer < number:

print('\033[5;31m猜小了')

else:

print('\033[5;32m猜对了')

三、循环语句

循环次数未知的情况下，建议采用while循环；

循环次数可以预知的情况下，建议采用for循环。

**1、while循环**

1）基本语法

result = 0

counter = 1

while counter <= 100:

result += counter

counter += 1

print('sum100 is %d' % result)

1. 语法进阶

break：结束当前循环，跳到下一条语句

continue：跳过本次循环，继续作下一次循环判断

else：循环执行后执行(break也会跳过)

3）案例：石头剪刀布

import random

all\_list = ['石头', '剪刀', '布']

win\_list = [['石头', '剪刀'], ['剪刀', '布'], ['布', '石头']]

counter = 0

while counter < 3:

index = int(input('''(0) 石头

(1) 剪刀

(2) 布

请出拳(0/1/2):'''))

com\_chose = random.choice(all\_list)

if com\_chose == all\_list[index]:

print('\033[5;33m平局\033[5;0m')

elif [all\_list[index], com\_chose] in win\_list:

print('\033[5;32m你赢了\033[5;0m')

break

else:

print('\033[5;31m你输了\033[5;0m')

counter += 1

else:

print('你尝试了%d次' % counter)

**2、for循环**

1）基本语法：

#for接受可迭代对象作为其参数(字符串、列表、元组、字典)

#支持break、continue、else

for ch in py\_str:

print(ch, end='\t')

1. range函数

#无需定义，可以节省内存空间；当需要使用再写入内存

语法：range(start, end, step=1)

range(5) --> [0, 1, 2, 3, 4]

range(1 ,6) --> [1, 2, 3, 4, 5]

range(5, 0, -1) --> [5, 4, 3, 2, 1]

1. 列表解析：列表推导式

#用于快速、便捷地生成列表

>>> print([10 + i for i in range(10)])

>>> print([10 + i for i in range(10) if i % 2 == 0])

4）案例1：9\*9乘法表

for i in range(1,10):

for j in range(1, i+1):

print('%d\*%d=%d' % (i, j, i\*j), end='\t')

print()

1. 案例2：使用列表解析，生成192.168.1.0/24网段中的IP地址

print(['192.168.1.' + str(ip) for ip in range(256)])

print(['192.168.1.%s' % ip for ip in range(256)])