nsd1909-py01-day01

准备开发环境

安装python

```
[root@localhost cloud5]# tar xf zzg_pypkgs.tar.gz
[root@localhost cloud5]# cd zzg_pypkgs/
[root@localhost zzg_pypkgs]# cd python3_pkg/
[root@localhost python3_pkg]# vim README
yum install -y sqlite-devel tk-devel tcl-devel readline-devel zlib-devel gcc gcc-c++
openssl-devel libffi-devel
tar xzf Python-3.6.7.tgz
cd Python-3.6.7
./configure --prefix=/usr/local/
make && make install
[root@localhost python3_pkg]# bash README
```

配置IDE

IDE：集成开发环境

python常用的IDE有：PyCharm，基于java

```
# pycharm必须有java支持
[root@localhost cloud5]# rpm -qa | grep java
java-1.8.0-openjdk-1.8.0.131-11.b12.el7.x86_64
# pycharm是绿色版的，解压后直接可用
[root@localhost cloud5]# mkdir ~/bin
[root@localhost cloud5]# tar xf pycharm2017.tar.gz -C ~/bin/
# 将pycharm配置到菜单中
[root@localhost bin]# yum install -y alacarte
# 点击 [应用程序]->[杂项]->[主菜单]->左窗格中的[编程]->[新建项目]
# Name: PyCharm2017  Command: /root/bin/pycharm2017/bin/pycharm.sh
# 点击左侧的图标，改为： /root/bin/pycharm2017/bin/pycharm.png
```

执行破解程序

```
[root@localhost nsd2019]# cp software/crack ~/bin/
[root@localhost ~]# /root/bin/crack &
```

配置vim为IDE：https://www.jianshu.com/p/29e7847f7298

配置python支持tab键补全：https://www.jianshu.com/p/b83a803cfc86

input函数

```
>>> input('username: ')  # username是屏幕提示语
username: tom
'tom'
>>> a = input('number: ')  # 注意，input读入的数据一定是字符类型
number: 10
>>> print(a)  # 变量直接使用，不用加 $
10
>>> a + 5  # 报错，不能把字符和数字直接运算
>>> int(a) + 5  # int函数可以将字符串 '10'转换成整数 10
15
>>> a + str(5)  # str函数可以将其他对象转换成字符
'105'
```

变量

变量：就是会变化的量。它和字面量literal相对，字面量如字符串'abc'，如数字100。变量，如上面例子中的a，它的值可以变化。

您应该拥有一双**透视**的眼睛，透过变量看到变量的值是什么。

合法的变量，需要满足以下要求：

- 首字符只能是字母或下划线
- 其他字符可以是字母数字下划线
- 区分大小写

变量赋值

```
# 变量赋值自右向左进行，将 =右边的表达式计算出结果，赋值给左边变量
>>> a = 5 + 5
>>> a
10
>>> a = a + 1   # 将a+1的结果11再赋值给变量a
>>> a
11
# 以上自增写法，可以简化为
>>> a += 1
>>> a
12
###############################################
>>> import this   # python之禅
The Zen of Python, by Tim Peters

Beautiful is better than ugly.      美胜丑
Explicit is better than implicit.   明胜暗
Simple is better than complex.      简胜繁
###############################################
>>> ++a   # 不会实现自增，也不报错，它是正正为正
```

运算符

- 算术运算符

```
>>> 5 / 3
1.6666666666666667
>>> 5 // 3   # 只得到商
1
>>> 5 % 3   # 模运算，只保留余数
2
>>> divmod(5, 3)
(1, 2)
>>> a, b = divmod(5, 3)  # 5除以3的商和余数，分别赋值给 a和b
>>> a
1
>>> b
2
>>> 2 ** 3  # 2的3次方
8
```

- 比较运算符

```
>>> 3 == 3   # 3等于3吗？
True
>>> 3 != 3   # 3不等3吗？
False
>>> 10 < 20 < 30   # python支持连续比较
True
>>> 10 < 20 > 15   # 相当于10 < 20 and 20 > 15，可读性差，不建议
True
```

- 逻辑运算符

```
>>> 10 > 5 and 5 > 8  # and两边全部为真，结果才为真，否则是假
False
>>> 10 > 5 or 5 > 8   # or两边，有一边为真结果就是真；两边全为假，结果才是假
True
>>> not 10 > 5   # not取反，将真变假，假变真
False
>>> not 10 > 50
True
```

数据类型

数字

- python中，没有小数点的是整型，有小数点的是浮点型。
- bool数True是1，False是0。

```
>>> True + 5
6
>>> False * 3
0
```

- 在python中，数字默认都是10进制数，如果采用非10进制数，需要加前缀

```
>>> 0o11   # 8进制以0o开头
9
>>> 0O11
9
>>> 0x11   # 16进制以0x开头
17
>>> 0X11
17
>>> 0b11   # 2进制以0b开头
3
>>> 0B11
3
>>> oct(10)  # 将10转为8进制
'0o12'
>>> hex(10)  # 将10转为16进制
'0xa'
>>> bin(10)  # 将10转为2进制
'0b1010'
```

字符串

- python中，字符串必须使用引号引起来，单双引号无区别
- python支持三引号，用于保留字符串格式

```
>>> words = """hello
... ni hao
... abc"""
>>> print(words)
hello
ni hao
abc
>>> danci = 'aaa\nbbb\nccc\nddd'
>>> print(danci)
aaa
bbb
ccc
ddd
```

- 字符串切片

```
>>> s1 = 'python'
>>> len(s1)   # 取长度
6
>>> s1[0]  # 取第1个字符
'p'
>>> s1[6]  # 报错，下标最大为5
>>> s1[5]  # 取最后一个字符
'n'
>>> s1[-1]  # 下标为负，表示从右向左取
'n'
>>> s1[-6]
'p'
>>> s1[2:4]  # 取切片，起始下标包含，结束下标不包含
'th'
>>> s1[2:6]  # 取切片时，下标超过范围也不会报错
'thon'
>>> s1[2:6000]
'thon'
>>> s1[2:]  # 结束下标不写，表示取到结尾
'thon'
>>> s1[:2]  # 开始下标不写，表示从开头取
'py'
>>> s1[:]  # 从开头取到结尾
'python'
>>> s1[::2]  # 2表示步长值
'pto'
>>> s1[1::2]  # 起始下标为1的字符开始取，步长为2
'yhn'
>>> s1[::-1]  # 步长值为负，表示自右向左取
'nohtyp'
```

- 字符串的拼接和重复

```
>>> 'abc' + '123'   # 字符串拼接
'abc123'
>>> '*' * 30   # '*'号重复30遍
'******************************'
>>> 'ab' * 5   # 'ab'重复5遍
'abababababab'
>>> '#' * 20
'####################'
```

- 成员关系判断

```
>>> s1
'python'
>>> 't' in s1   # t在s1中吗？
True
>>> 'th' in s1  # th在s1中吗？
True
>>> 'to' in s1  # to在s1中吗？
False
>>> 'to' not in s1   # to不在s1中吗？
True
```

列表

- 类似于shell中的数组
- 列表的大部分操作与字符串一样，可以取下标、切片等

```
>>> l1 = ['tom', 'jerry', 10, 20, [1, 2, 3]]
>>> len(l1)
5
>>> l1[-1]  # 取下标
[1, 2, 3]
>>> l1[-1] = 30  # 将最后一项重新赋值
>>> l1
['tom', 'jerry', 10, 20, 30]
>>> l1[:2]
['tom', 'jerry']
>>> 10 in l1
True
>>> l1.append(10)  # 在结尾追加10
>>> l1
['tom', 'jerry', 10, 20, 30, 10]
```

元组

- 跟列表类似，也支持像字符串一样的切片、下标等操作
- 元组可以认为是不可变的列表

```
>>> t1 = ('tom', 'jerry', 10, 20)
>>> len(t1)
4
>>> t1[2:]
(10, 20)
>>> t1[0] = 'bob'  # 报错，元组不可变
```

字典

- 字典没有顺序

```
>>> d1 = {'name': 'tom', 'age': 20}
>>> len(d1)
2
>>> 'tom' in d1  # 'tom'是字典的键(key)吗？
False
>>> 'name' in d1
True
>>> d1['name']  # 取出字典中'name'对应的值(value)
'tom'
```