Day 01

整型：

1. 对比：
    - 在python 2 版本中有整型，长整型long
    - 在python 3 版本中全部都是整型
2. 用于计算和比较
3. 整型和布尔值的转换
    - 二进制转换成十进制：
        - print (int("111010100",2))
        - ```
          # 2进制 - 10进制
          0 * 2**0 + 1 * 2 ** 1 + 1 * 2**2 + 1* 2**3 + 1 * 2 ** 4 + 1 * 2 **
          5
          0 + 2 + 4 + 8 + 16 + 32 = 62
          print(int("111110",2))
          ```
    - 十进制转换成二进制：
        - print(bin(30))
        - ```
          # 10进制 - 2进制
          30   0
          15   1
          7    1
          3    1
          1    1
          11110
          print(bin(30))
          ```
4. 整型中只要是非零的就是True
5. 在布尔值中：1--True，0--False
6. 在字符串中的内容非空就是True

1. ```
   int,str,bool
   a = bool(0)
   print(a)

   a = int(True)
   print(a)

   a = int(False)
   print(a)

   a = bool("a")
   print(a)

   a = bool("啊")
   print(a)
   ```
   ```

```
a = bool(" ")
print(a)

a = bool("")
print(a)

a = str(True)
print(type(a),a)

a = str(False)
print(type(a),a)
```

字符串详解：

1. **索引（下标）：**
    1. 从左往右，开头为0
    2. 从右向左，结尾为-1
    3. 索引的条件不能超出索引本身最大值

```
a = "alex_wu_sir,_tai_bai_日魔"
print(a[5])
print(a[6])
print(a[11])
print(a[-2])
print(a[-1])
```

2. **切片：**
    1. [起始位置：终止位置]----原则：顾头不顾尾
    2. 终止位置可以超出索引本身
    3. [::-1]----从头到尾字符串反转

```
a = "alex_wu_sir,_tai_bai_日魔"
print(a[21:100])
print(a[21:])  [21(起始位置):(默认到最后)]
print(a[:])    [(默认从最开始):(默认到最后)]
print(a[0:-2])
```

3. **步长：**
    1. 步长决定了查找时迈的步子
    2. 步长决定了查找的方向
    3. [起始位置：终止位置：步长]
    4. [::-1]----从头到尾字符串反转

```
name = "alex,wusir,太白金星,女神,吴超"
1.太金
print(name[11:15:2])
2.神女
print(name[-4:-6:-1])
3.星白
print(name[-7:-13:-2])
4."alex,wusir,太白金星,女神,吴超" 整体反转
print(name[::-1])
***** 面试题的答案print(name[10:10000:200000])
```

字符串的方法

1. upper()--全部大写
2. lower()--全部小写
3. startswith()--以.....开头
4. endswith()--以--结尾
5. replace()--把.....替换成.....
6. count()--统计字符出现的次数
7. strip()--脱（删除前后出现的空白）
8. split()--分割(默认空格,换行符\n,制表符\t)
9. format()--字符串格式化，填充

```
1. name = "{}今年:{}".format("宝元",18)     # 按照位置顺序进行填充
print(name)
输出：宝元今年：18
2. name = "{1}今年:{0}".format("宝元",18)     # 按照索引进行填充print(name)
输出：18今年：宝元
3. name = "{name}今年:{age}".format(name="宝元",age=18)
# 按照名字进行填充print(name)
输出：宝元今年：18
```

10. isdigit()--判断字符串中的内容是否全是数字
11. isdecimal()--判断是不是十进制数
12. isalnum()--判断是不是数字，字母，中文
13. isalpha()--判断是不是字母，中文

for 循环

1. for：关键字
2. i ： 变量名（可以随意更改）
3. in ： 关键字
4. msg：可迭代对象（python数据类型中，除了int，bool其余都可以迭代）

Day 02

列表

数据类型之一，存储数据，大量的，存储不同类型的数据

列表是一种有序的容器　支持索引

列表是一种可变数据类型 原地修改

列表中只要用逗号隔开的就是一个元素，字符串中只要是占一个位置的就是一个元素

1.1 列表的增加：

- lst.append()　追加（在最末尾的地方进行添加）
- lst. insert()　插入
- lst.extend()　迭代添加

1.2 列表的删除

- lst.clear()　清空
- lst.pop()　删除
- lst.remove()　移除

```
del lst[4]    # 通过索引删除
del lst[2:5]    # 通过切片删除
del lst[1:5:2]      # 通过步长删除
print(lst)
```

1.3 列表的修改：

- 通过索引进行修改：

```
lst = [1,2,3,4,5]
lst[2] = 80    通过索引进行修改
print(lst)
```

- 通过切片进行修改：

```
lst = [1,2,3,4,5]
lst[1:3] = "20"     # 通过切片进行修改，默认步长为1，修改的内容必须是可迭代的对象，修改的内容可多可少
print(lst)
```

- 通过步长进行修改：

```
lst = [1,2,3,4,5]
lst[1:5:2] = 100,100    # 步长不为1的时候，必须一一对应
print(lst)
```

1.4 列表的查：

使用for 循环索引

```
lst = [1,2,3,4,5]
for i in lst:
    print(i)
```

1.5 列表的嵌套：

```
lst = [1,2,[3,4,5,["alex[]",True[[1,2,]],90],"wusir"],"taibai"]
lst1 = lst[2]
[3, 4, 5, ['alex[]', True, [[1, 2]], 90], 'wusir']
lst2 = lst1[3]
['alex[]', True, [[1, 2]], 90]
str_1 = lst2[0]
print(str_1[-1])


print(lst[2][3][0][-1])

结果：输出"alex[]"的右边的中括号
```

元组

2.1 关键字及特性：

- 关键字：tuple
- 特性：有序、不可变
- 格式：（1，2，3） #使用圆括号以及逗号分隔开

2.2 元组的方法：

- 元组支持查询，元组就是一个不可变的列表
- 方法：统计、获取索引

```
tu = (1,2,3,4,5,1,2,1)
print(tu.count(1))   # 统计tu元组中“1”的个数


tu = (1,2,3,4,5,1,2,1)
print(tu.index(2))  # 通过元素查询索引“2”的位置
```

2.3 元组的用途：

在配置文件中使用

2.4 元组的嵌套：

```
tu = (1,2,3,4,(5,6,7,8,("alex","wusir",[1,23,4])))
print(tu[4][4][0])

结果：“alex”
```

range 范围

3.1 用途：为了解决不能循环数字、

3.2 range中py2与py3 的区别：

print(range(1,10)) Python3中打印range是自己range自己本身

print range(1,10) Python2中打印range获取的是一个列表，列表的元素是1-9

3.3range方法：

```
range(1,10)       [起始位置：终止位置]  顾头不顾尾
range(1,10,2)     [起始位置：终止位置：步长] 默认为 1
range(10)        10代表的是终止位置，起始位置默认为 0range是一个可迭代对象
```

```
for i in range(2,10,2):
print(i)      # 输出2 4 6 8

for i in range(100,-1,-1):
print(i)       # 输出100-0
```

Day 03

字典：

1. 定义：dict
2. dict = {"key":"value"} -- 键值对
3. 作用：存储大量数据，数据和数据起到关联作用
4. 所有的操作都是通过键来完成
5. 键：必须是不可变的数据类型（可哈希），且唯一不可变
6. 值：任意的数据类型
7. 字典是可变的数据类型，无序的
8. 字典的增：
    1. 暴力添加：dict["key"] = "value"
        - 添加一个键值对，可以是列表
    2. 有责添加，无则不添加
        - dict。setdefault（“key”，“value”）

```
 dic = {
        "日魔":["看动漫","健身","吃包子","吃大煎饼","吃大烧饼"],
        "炮手":"飞机",
        "豹哥":"贴膏药",
        "宝元":"宝剑",
        "alex":"吹牛逼"
 }
dic.setdefault("元宝",["唱","跳","篮球","喝酒"])
print(dic)
```

        - setdefault分为两步：
            - 先查看键是否在字典中
            - 不存在的时候进行添加
9. 字典的删
    1. dict.pop() -- pop删除字典中的键进行删除，返回的也是删除的值
    2. dic.popitem() -- 随机删除 python3.6是删除最后一个
    3. dict.clear（） -- 清空
    4. del dict -- 删除的是整个容器
    5. del dic["key"] -- 通过键进行删除
10. 字典的改

1. dic["key"] = "value" -- 有则覆盖，无则添加
2. dict.update() --- 当update中的字典里没有dic中键值对就添加到dic字典中,如果有就修改里边对应的值

11. 字典的查

1. dict.get("key") -- 查询不到就返回None
2. dic.get("key","定制内容") --- 查找不到就返回自己输入的定制内容
3. dict.setdefault("key") --- 查询不到返回None
4. dict.["key"] -- 查询不到就报错
5. 查看所有的键

```
for i in dic:   # 查看所有的键
print(i)
```

6. 查看所有的值

```
for i in dic:   # 查看所有的值   print(dic.get(i))
```

7. 获取到的是一个高仿列表

```
print(dic.keys())   # 获取到的是一个高仿列表
print(dic.values()) # 获取到的是一个高仿列表
```

8. 高仿列表支持迭代，不支持索引

```
面试题:
    a = 10
    b = 20
    a,b = b,a
    print(a)
    print(b)
```

```
    dic = {"key1":2,"key2":4}
    a,b = dic
    print(a)
    print(b)
```

12. 解构的作用:

1. 两个都是列表的时候才可以相加

```
lst = [1,2,3,4,5,6,7,8]
lst1 = lst[:2]
两个都是列表的时候才能够相加 lst1.append(lst[4])
for i in lst1:
print(i)
```

2. *是万能接受
3.

```
lst = [1,2,3,4,5,6,7,8]
a,b,c,d,e,*aa = lst     # *是万能接受print(a,b,e)
```

4. 字典嵌套查找的时候一定是按照键一层一层进行查

Day 04

小数据池

1. int： -5~256
2. str：
    1. 字母，数字长度任意符合驻留机制
    2. 字符串进行乘法时总长度不能超过20
    3. 特殊符号进行乘法时只能乘以0

代码块：

一个py文件，一个函数，一个模块，终端中的每一行都是代码块

1. int：-5~无穷大
2. str：
    1. 定义字符串的时候可以是任意的
    2. 字符串（字母，数字）进行乘法时总长度不得超过20
    3. 特殊字符（中文，符号）进行乘法时乘以0或者1
3. bool：
    1. True
    2. False

is是判断两边的内存地址是否相同

==判断两个值是否相等

代码块、小数据池同在的情况下先执行代码块。

驻留机制：节省内存空间，提升效率（减少了开辟空间和销毁空间的耗时）

集合

1. 集合是python中的数据类型之一
2. 定义方式：set
    - 集合就是一个没有值的字典
3. 字典的值：唯一，不可变
4. 集合：无序，可变
5. 集合天然去重

```
# s = {1,23,4,2,1,2,3,1}
print(s)
输出结果：{1, 2, 3, 4, 23}
```

```
# 面试题：
lst = [1,223,1,1,2,31,231,22,12,3,14,12,3] print(list(set(lst)))
输出结果：[1, 2, 223, 3, 231, 12, 14, 22, 31]
```

6. s = {} #空字典

s = set（） #空集合

7. 增：

```
 s = set()

s.add("alex")   #直接添加，

输出结果为：{'alex'}

 s.update("wusir")   #迭代添加，

输出结果为：{'r', 'i', 'w', 's', 'u'}

 set("wusir")    #迭代添加，

输出结果为：set()
```

8. 删

```
s = {100,0.1,0.5,1,2,23,5,4}
s.remove(4)    # 通过元素删除
print(s)
s.clear()  # 清空
s.pop()      # 随机删除 （最小的）
print(s)
```

9. 改：
    - 先删后加
10. 查：
    - for 循环
11. 其他操作：
    1. 差集--“-”
    2. 交集--“&”
    3. 并集--“|”-管道符
    4. 反交集--“^”
    5. 子集--“>”返回的是一个布尔值
    6. 父集（超集）--"<"返回的是一个布尔值
    7. 冻结集合：frozenset()

深浅拷贝

1. 赋值：将多个变量指向一个同一个内存地址就是赋值
2. 浅拷贝：
    1. 只拷贝第一层元素的地址，只有修改第一层的时候元数据不受影响
    2. 给可变数据类型进行添加的时候源数据会受影响
    3. = 是修改，.append是添加
    4. 可变数据类型能够添加和修改，不可变数据类型只能修改

```
a = [1,2,3,[4,5]]
b = a[:]  # 浅拷贝
print(id(a[-1][0]))
print(id(b[-1][0]))
a.append(9)
print(a)
print(b)
```

```
a = [1,2,3,[4,5,6,[9,10]],67]
b = a  # 赋值
b = a[:]
a[-2].append(10)#[1, 2, 3, [4, 5, 6, [9, 10], 10], 67]
a[-2][-1].append(10)#[1, 2, 3, [4, 5, 6, [9, 10, 10]], 67]
print(b)
```

3. 深拷贝：不可变数据类型内存地址共用,可变数据类型新开辟一个空间

```
# import copy  # 导入 copy模块
a = [1,2,3,[4,5],6]
b = copy.deepcopy(a)
print(id(a[-2]))#2812949845896
print(id(b[-2]))#2812949847304
print(a)
print(b)
print(a == b)       #True
print(id(a),id(b))
print(a is b)       #False
```

可变数据类型：list（列表），dict（字典），set（集合）

不可变数据类型：int（整型），str（字符串），tuple（元祖），bool（布尔值）

Day 05

基础数据了类型补充：

1. 字符串：str
    1. 首字母大写：capitalize
    2. 每个单词的首字母大写：.title()
    3. 大小写转换：.swapace()
    4. 居中及填中：.center(数字，''参数'')
    5. 居中：.center(数字)
    6. 查找：.find()--通过元素查找索引，查找不到时返回-1
    7. 查找：.index()--通过元素查找索引，查找不到时报错
    8. 列表+列表 str + str
    9. 列表相乘 str * 5
    10. 字符串相加或者相乘，都是开辟一块新的空间
2. 列表：list
    1. 通过元素查找索引：.index()
    2. 排序：
        1. .sort（）--默认从小到大，升序
        2. .sort（reverse = True）-- 降序

3. 人工降序：

```
lst = [1,23,4,5,7,8,9]
lst.sort()
lst.reverse()              #人工降序      print(lst)
```

4. 反转：.reverse（）

```
反转：.reverse()
lst = [1,23,4,5,7,8,9]          lst.reverse()
print(lst)          # 将源数据进行反转
```

5. 不修改元数据进行反转：

```
lst = [1,23,4,5,7,8,9]
lst1 = lst[::-1]
print(lst)
print(lst1)         # 不修改源数据进行反转
```

6. 列表相加：

```
lst = [1,2,3,4]
lst = lst + [1,2,3]
print(lst)  # [1,2,3,4,[1,2,3],]   [1,2,3,4,1,2,3]
```

7. 列表相乘：列表进行乘法运算时，元素是共用的

```
lst = [1,2,3] * 5
print(lst)              print(id(lst[0]),id(lst[3])
```

3. 元祖：tuple

```
tu = (1,[]) * 3
print(tu)    #输出结果：(1, [], 1, [], 1, []) tu[-1].append(10)
print(tu)  #输出结果：(1, [10], 1, [10], 1, [10])
```

4. 字典：dict

1. dict.popitem() 随机删除，3.6版本默认删除最后一位

```
dic = {}
dic.fromkeys("abc",[])   # 批量创建键值对 "a":[],"b":[],"c":[]
print(dic)              #输出的结果为：{}
```

```
dic = {}
dic = dic.fromkeys("abc",[])
print(dic)
dic["b"] = 11
dic["a"].append(10)
print(dic)
输出结果为：{'a': [10], 'b': 11, 'c': [10]}
```

2. fromkeys第一个参数必须是可迭代对象，程序会将可迭代对象进行迭代成为字典的键，第二个参数是值（这个值是共用的）

3. fromkeys 共用的值是可变数据类型就会有坑，不可变数据类型就没事

5.
```
可变数据类型：列表（list），字典（dict），集合（set）
不可变数据类型：字符串（str），整型（int），布尔值（bool），元祖（tuple）
有序数据类型：字符串（str），列表（list），元祖（tuple）
无序的数据类型：字典（dict），集合（set）
可以用索引取值：列表（list），元祖（tuple），字符串（str）
可以用键取值：字典（dict）
直接取值：整型（int），布尔值（bool），集合（set）
```

6. 数据类型转换：

```
1. 字符串转整型：int（str）
2. 整型转字符串：str（int）
3. 字符串转布尔值：bool（str）
4. 布尔值转字符串：str（bool）
5. 整型转布尔值：bool（int）
6. 布尔值转为整型：int（bool）
7. 列表转元祖：tuple（list）          print（tuple（list））
8. 元祖转列表：list（tuple）          print（list（tuple））
9. 列表转集合：set（list）
print (set(list))
10. 集合转列表：list（set）
print(list(set))
11. 元祖转集合：set（tuple）
12. 列表转字符串：使用join，
如下所示：
list -- str
lst = ["1","2","3"]              print("".join(lst))
13.字符串转列表：使用分割
str -- list
s = "alex wusir 太白"             print(s.split())
```

7.
```
重点：  列表乘法，
    find用法
        join用法
        元祖形式：（1，）--有逗号的才是元祖                    元祖乘法
        字符串转列表
        列表转字符串
```

以后遇到的坑

1. 死循环：

```
lst = [1,2,3]
for i in lst:
lst.append(4)
print(lst)
```

2. 使用for循环清空列表元素内容

    1. 从前向后删除

    2. 创建一个新的容器，循环新的容器删除旧的容器内容

3.
```
lst = [1,[2]]
lst[1] =lst
print(lst)
答案：[1,[...]]
```

4. 字典和集合：

    1.
```
dic = {"key":1,"key1":2}
for i in dic:
    if i == "key1":
        dic[i] = dic[i] + 8
        dic[i] = 10
        else:
            print("没有这个键!")
print(dic)
```

    2. 字典和集合在遍历(循环)时不能修改原来的大小(字典的长度),可以进行修改值

```
s = {1,2,3,4,5,6}
for i in s:
    s.pop()
print(s)
```

二次编码

1. 编码：
    1. ascii码：
        1. 支持英文，数字，符号　1字节
        2. 不支持中文
    2. gbk（国标）
        1. 支持英文，数字，符号 1字节
        2. 支持中文　2字节
    3. unicode（万国码）：
        1. 支持英文，数字，符号 4字节
        2. 支持欧洲　　4字节
        3. 支持亚洲　　4字节
    4. utf-8：
        1. 支持英文，数字，符号　--1字节
        2. 支持欧洲　---　2字节
        3. 支持亚洲 ----　3字节
2. 字节：存储和传输
    1. 编码

```
今天是个好日子
s = "今天"    # b'\xe4\xbb\x8a\xe5\xa4\xa9'
s1 = s.encode("utf-8") # 编码
print(s1)
```

    2. 解码

```
print(b"meet")  #只有字母才能这么搞
s2 = s1.decode("utf-8") #解码
print(s2)
```

3.
```
# 必会     #
python3: 默认编码unicode
pyhton2:默认编码ascii
python2不支持中文     # 重要:
encode()  # 编码     # decode()  # 解码
用什么编码就要用什么解码     # 网络传输一定是字节
```