Python基础回顾

数据类型(5类)

整型

- 即数学概念中的整数
- 十进制/二进制/八进制/十六进制，数据类型都是int

```
# 整型
print(1, 100, -21, 0, type(100))
# 二进制，0b,
print(0b10, type(0b10))
# 八进制  0o10
print(0o10, type(0o10))
# 十六进制，开头：ox ，0 1 2 3 4 5 6 7 8 9 10 A B C D E F
print(0xff, type(0xee))

# import random
```

```
1 100 -21 0 <class 'int'>
2 <class 'int'>
8 <class 'int'>
255 <class 'int'>
```

- 进制转换：

```
其他进制转十进制
1. 将字符串转换成整型，10进制
print("123", type("123"))
print(int("123"), type(int("123")))
2. 其他进制，转换成10进制
print(int('0b11', 2))
# 7 * 8 + 7 = 63
print(int('0o77', 8))
# 15 * 16 + 15
print(int('0xff', 16))
# 十进制转其他进制
bin 10转2
print(bin(100))
# oct 10转8
print(oct(100))
# hex 10 转 16
print(hex(100))
```

```
0b1100100
0o144
0x64
```

- 应用场景：RSA非对称加密，AES加密

浮点数

- 即数学概念中的小数：`3.14`，`-5.89`
- 科学计数法：`1.23e6` `2.1e-5`
- 浮点数计算时会四舍五入：

```
# 浮点型
print(3.14, type(3.14))
# 除法，四舍五入
print(3.14 / 3)

print("3.14", type("3.14"))
print(    float(3),      type(float(3))       )
# 不是四舍五入，而是直接去掉小数点后的数
print(    int(3.14),      type(int(3.14))       )
```

```
3.14 <class 'float'>
1.0466666666666666
3.14 <class 'str'>
3.0 <class 'float'>
3 <class 'int'>
```

字符串

- 定义：单/双/三 引号括起来的**任意文本**，该文本本身是就是字符串

```
# 字符串
print( 'aslkdfakjsasdkfhasjk*&^*&@^#!(*@#&(SDJFHSDKJFH第三方空间啊还是独家开发))'
)
print('hello', type('hello'))
print("hello", type("hello"))
print('''hello''', type('''hello'''))

# 预期：输出：  I'm ok
# print('I'm ok')
print("I'm ok")
print('我是"外国"人')

# 转义
```

```
print('I\'m ok')
```

```
aslkdfakjsasdkfhasjk*&^*&@^#!(*@#&(SDJFHSDKJFH第三方空间啊还是独家开发))
hello <class 'str'>
hello <class 'str'>
hello <class 'str'>
I'm ok
我是"外国"人
I'm ok
```

- 注意点：
    1. 若在控制台输出，不会输出引号，即：括起来的那对引号只是字符串的标识，而不是字符串本身，字符串本身只是扩起来头的那部分内容
    2. 不区分引号类型，即：单/双/三引号的用法效果是一样的
    3. 若文本内容包含单引号，"括号"要用双引号；若文本内容包含双引号，"括号"要用单引号

```
# 内容包含单引号，括号要用双引号；print("I"m ok") 则非法
print("I'm ok")
# 内容包含双引号，括号要用单引号；print('She is my 'girls'.') 则非法
print('She is my "girl".')
# 内容即包含单引号，又包含双引号，需要使用转义字符
print('I\'m ok. She is my \'girl\".')
```

```
I'm ok
She is my "girl".
I'm ok. She is my 'girl".
```

- 注意点：
    4. 常用转义字符：`print("\n 1\t1 \\")`
    5. 强制不转义：`print(r'\n\t\\')`
    6. 三引号常用场景：函数注释说明，sql语句

```
# 回车/tab 缩进
print('1\n2\t3 \\')
```

```
1
2 3 \
```

布尔型

- 布尔类型仅两种情况：True真/False假，注意开头字母必须大写
- 布尔值产生的2种方式：
    1. 直接赋值
    2. 逻辑运算：数值比较/and与/or或/not非

```
# 布尔类型
print(True, False, type(True))
print("True", "False", type("True"))
# # 逻辑运算
print(3 > 1)
print(1 == 1)
print(3 < 1)
# and，全为真则真
print(3 > 1 and 2 > 1)
# or，一个真则真
print(3 < 1 or 1 == 1)
# # 取反
print(not False)
print(not 3 > 1)
```

```
True
True
True
False
```

3. 条件判断：True则会进入逻辑分支，False则不进入

```
# 条件判断
a = True
b = False
if a:
    print("我是真")
if b:
    print("我是假")
if 3 > 1:
    print("3大于1")
if 1 in [1, 2]:
    print("1在列表里")
```

```
我是真
3大于1
1在列表里
```

空值

- `None`是一个特殊值，不能理解为`0`，`0`是有意义的，是一个整型数

```
print(0, type(0))
print(None, type(None))
```

```
0 <class 'int'>
None <class 'NoneType'>
```

变量

- 类比初中方程中的x变量，但是在python的x不仅可以是数字，也可以是任意数据类型
- 变量名称有规范：英文、数字、下划线`_`，而且不能用数字开头
- 赋值语句：`a = 1`，`=`符号完成了一个赋值操作，即：把**右边**的值传递给左边的变量
    1. 左边必须是变量，而不能是一个确定的值：`1=3.14`则非法
    2. 变量的值是可以被覆盖的，并且可以变更变量类型，是动态语言的特性，区分java和c静态语言

```
# 变量覆盖
apple_helllo_asdjkfhk = 1123
a = 1
# a = False
# a = None
print(a)
```

```
1
```

3. 内部实现：先在内存中创建了一个变量`1`，然后再在内存中创建了一个名为`a`的变量，并让这个`a`指向`1`

格式化输出

占位符	替换内容
%d	整数
%f	浮点数
%s	字符串
%x	十六进制整数

- %s表示用字符串替换，%d表示用整数替换，有几个%?占位符，后面就跟几个变量或者值，顺序要对应好

```
# 一个变量
# print('hello python')
# print('Hello, %s' % 123)
# 两个不同变量
print('Hi, %s, you have $%d.' % ('duoceshi', 1000000))
```

```
Hi, duoceshi, you have $1000000.
```

- 格式化补齐，整型和浮点型才可以补
- 三种补齐方式：补空格/补0/指定位数

```
# 补充空格/补充0
print('%2d-%04d' % (3, 1))
# 指定位数
print('%.7f' % 3.1415926)
```

```
 3-0001
3.1415926
```

- 不确定什么类型的时候，就用 %s，它会把任何数据类型转换为字符串

```
# 任意类型都可以转义
print('Age: %s. Gender: %s' % (25, True))
```

```
Age: 25. Gender: True
```

- format()格式化，用法类似

```
print( 'Hello, {0}, 成绩提升了 {1:.1f}%'.format('小明', 17.125))
```

```
Hello, 小明, 成绩提升了 17.1%
```

```
print( 'Hello, {1} {0}'.format('xiaobai', "xiaohei"))
```

```
Hello, xiaohei xiaobai
```

小结

- python中，任何数据都是一个对象，变量就是用来指向这些数据对象的，对变量赋值，就是把数

据和变量关联起来
- 对变量`x = y`，只是把y指向的对象，也赋给了x，之后x和y都指向同一个数据，并不会影响y自己的指向

```
print('hello')

a = 10
b = 20
c = a
print(c)
```

```
hello
10
```

列表和元祖

列表list

- 有序集合，支持列表索引：从0开始

```
# a = [1 , 2]
# b = [2 , 1]
# if a == b:
#     print("is ok")
# else:
#     print("not ok")
a = [1, 2, "hello", '中文', True, 4.24, None, 77]
# 依次索引
# print(a[0])
# print(a[1])
# print(a[2])
# print(a[3])
# print(a[4])
# print(a[5])
# print(a[6])
# print(a[7])
# print(a[8])
# 反向索引，反向索引从-1开始
# print(a[-3])
# 越界报错，最常见的报错
# print(a[8])
```

```
4.24
```

- 可增删元素，可变长度

```
a = [1, 2, "hello", '中文', True, 4.24]
print(a)
# 末尾增加元素
a.append("xiaobai")
print(a)
# # 指定位置插入元素
a.insert(3, "新插入的元素")
print(a)
# 删除元素，不传，默认删最后
a.pop(0)
print(a)
```

```
[1, 2, 'hello', '中文', True, 4.24]
[1, 2, 'hello', '中文', True, 4.24, 'xiaobai']
[1, 2, 'hello', '新插入的元素', '中文', True, 4.24, 'xiaobai']
[2, 'hello', '新插入的元素', '中文', True, 4.24, 'xiaobai']
```

- 替换元素，直接赋给索引位置

```
a = [1, 2, "hello", '中文', True, 4.24]
a[3] = "英文"
print(a)
```

```
[1, 2, 'hello', '英文', True, 4.24]
```

- 列表中也可以包含列表，称二维列表，甚至可以多维

```
# 初始化
a = [1, 2, "hello", ["name", "sex", "中国"], True]
print(a)
```

```
[1, 2, 'hello', ['name', 'sex', '中国'], True]
```

- 字符串和列表的转换：split和join

```
# 字符串切割成列表
# a = 'apple,pen,hello'
# print(a.split('l'))
# 列表合并成字符串
b = ['我', '是', '中国人']
print('*'.join(b))
```

```
我*是*中国人
```

元祖tuple

- 和列表类似，但是tuple一旦初始化就不能修改
- 用得较少，测试尽量用列表

```
a = (1, 2, 3)
print(type(a))
# 元组不能修改
# a[1] = 2
print(a[-1])
# a.pop()不支持
```

set减法

- set是一个集合
- 比较两个字符，并把不同的部分输出

```
a = [1 ,1, 2, 2, 3, 3]
# print(set(a))
h1 = '1,2,3'
h2 = '3,4,5'
# # 从前一个集合中，剔除前面包含后面包含的元素
b = set(h2.split(','))
c = set(h1.split(','))
print(b, c)
print(b-c)
print(set(h1.split(',')) - set(h2.split(',')))
```

```
{'4', '5', '3'} {'1', '2', '3'}
{'4', '5'}
{'1', '2'}
```

条件判断

1. if的判断结果是Ture，则去执行内容，并忽略剩下的所有条件判断，有次序性
2. 如果前一个判断为False，则执行下一个判断
3. 所有判断都为False，则执行else内容

```
# 完整的条件结构
if <条件判断1>:
    <执行1>
elif <条件判断2>:
    <执行2>
elif <条件判断3>:
    <执行3>
else:
    <执行4>
```

```
# 忽略第二个判断
age = 20
if age >= 6:
    print('teenager')
elif age >= 18:
    print('adult')
else:
    print('kid')
```

```
teenager
```

4. 简写的判断：非零数值、非空字符串、非空list等

```
x = -1
y = '阿斯顿发撒地方'
z = ['123']
# x不为0，则为True
if x:
    print(True)
# 字符串不为空，则为True
if y:
    print(True)
# 列表不为空列表，则为Ture
if z:
    print(True)
```

```
True
```

循环

for循环

- 遍历列表，依次执行循环体

```
sum = 0
# range(10)
for x in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
    sum = sum + x
print(sum)
```

```
55
```

while循环

- 只要条件满足，就不断循环，条件不满足时退出循环

```
sum = 0
n = 99
while n > 0:
    sum = sum + n
    n = n - 2
print(sum)
```

跳出循环

- break结束所有循环

```
sum = 0
# range(10)
for x in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
    if sum > 20:
        break
    sum = sum + x
print(sum)
```

```
21
```

- continue结束本次循环

```
sum = 0
# range(10)
for x in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
    if x == 5:
        continue
    sum = sum + x
print(sum)
```

```
50
```

小结

1. break跳出所有循环，continue跳出本轮循环并开始下一轮循环，这两个语句一般都配合if语句
2. break和continue容易出错，尽量少用

课后题

1. 计算1-100的偶数和

字典和集合

字典

- 一个键值对，key-value形式存储，本质上就是一种映射关系表，和顺序无关，在java中叫map
- 特点：
    1. 无论字典多大，查询速度都不会变慢，利用索引来检索(key -> dict计算索引 -> value)
    2. key不能相同，重复赋值会覆盖

```
# 必须先申明，否则会报错
a = {}
a["name"] = "duoceshi"
print(a)
# 覆盖
a["name"] = "duoceshi2"
print(a)
```

```
{'name': 'duoceshi'}
{'name': 'duoceshi2'}
```

3. 索引时，key不存在则会报错，通过get()，或者in

```
# 直接初始化
a = {"name": "duoceshi"}
# print(a["age"])
# 判断方法1
# print("name" in a)
# print("name1" in a)
# # 判断方法2
print(a.get("name"))
print(a.get("name1"))
```

```
duoceshi
None
```

4. 删除key时，也会删除值

```
# 直接初始化
a = {"name": "duoceshi", "age": 4}
# 删除name
a.pop("name")
print(a)
```

```
{'age': 4}
```

5. 字典的key是不可变的，因为字典通过key去计算位置(hash算法)，如果每次key都不一样，则字典内部会错乱，所以字符串、整数等都是可以当作key，但可变的list不能作为key

```
# 列表不能当key
# key = [1, 2, 3]
# a[key] = "hello"
# # 元组tuple可以当key
d = {}
a = (1, 2, 3 ,4 ,5 ,"sdhkjf ")
d[a] = "duoceshi"
print(d)
# # 元组tuple中包含list也不能当key
d = {}
a = (1, 2, ["hello"])
d[a] = "duoceshi"
```

- 字典与列表对比：

字典dict	列表list
插入速度快	插入速度慢
需占用太大内存，浪费内存	占用内存少，不浪费内存

结论：dict是以时间换取空间的方法，测试时一般不用过于关注，但开发时需要注意区别使用

集合

- 一组key的集合，但不存value
- 特点：
    1. key不能重复
    2. key是无序的

```
# 用列表初始化，重复的key自动过滤
s1 = set([1,2,3])
s1.add(4)
# 重复添加
s1.add(4)
print(s1)
# 删除key
s1.remove(2)
print(s1)
```

```
{1, 2, 3, 4}
{1, 3, 4}
```

3. 符合数学运算的交集，并集

```
s1 = set([1,2,3])
s2 = set([2,3,4])
print(s1 & s2)
print(s1 | s2)
```

```
{2, 3}
{1, 2, 3, 4}
```

4. 集合中不能放入可变对象，如：列表，因为无法保证set内部元素不重复

不可变对象

```
# 列表是可变对象
a = [3, 2, 1, 34,23,123,323,2,3,3,4,54,5,6,]
# 排序
a.sort()
print(a)
# 字符串是不可变对象
b = "abc apple "
# # a替换成A
c = b.replace("a", "A")
print(b)
print(c)
```

```
[1, 2, 2, 3, 3, 3, 4, 5, 6, 23, 34, 54, 123, 323]
abc apple
Abc Apple
```

理解：

1. 变量指向对象，类似指针
2. 赋值后，操作变量，就相当于操作对象
3. 可变长的对象调用方法后，直接被修改了，如：sotr()方法
4. 不可变对象调用方法后，不会修改原对象，而是生成一个新对象返回，如：replace()

小结

1. 为了避免麻烦，尽量使用str当key