5分钟读懂python,走进python工程

day01

1.01 安装软件

windows

mac

linux

1.02 配置python环境变量

在安装python时，应在添加路径上打勾

1.03 pycharm的安装和基本配置

选用专用版，不要汉化

可破解使用。详见破解文档。

1.04 xmind软件的使用

好用的一款思维导图软件，可以在每日学习完成，列出一个思维导图，自己回想添加学习的内容，细化到每一小的分支。

1.05 sublime和npp的安装使用

sublime好用的一款文档编辑器

1.06 markdown语法的使用

软件 typora，好用的文本编辑软件

day02

2.01计算机的介绍

硬件

软件

2.02 什么是编程语言

基于二进制

将人类的语言，转换为机器能识别人语言。

编程语言写程序，程序编码器

语言分类

编译性语言

C语言，运行快

解释性语言

python

2.03 python的历史

什么是python

解释性语言，是当今世界上最流行的编程语言之一。

python发展背景

吉多，荷兰人，1989年，写出了python

语言诞生

1991年，第一个编译器诞生。很多来自于C，但又受到ABC影响。

版本

多版本，2.7或3.7

优缺点

2.04 python的使用场景

web应用方向

操作系统的运维

网络爬虫

科学计算

桌面软件

服务器软件

游戏

2.05 pycharm的基本使用

软件的安装

专业版，英文，不要汉化

2.06 交互式编程介绍

直接在终端中运行的解释器

IEPL环境

```
>>>print("hello,world")
hello,world
>>>2**10
1024
>>>exit()#退出
```

读，运行，打印，循环

大段代码不适合用交互编程

2.07 注释的使用

几种分类

ctrl+/加减注释

ctrl+鼠标，点击，查看注释。

```
# 1.需要注释的内容，位于代码行的上面#为了对代码说明，是为了给人来看的。表示的是单行注释
for i in range(10:)  #代码行注释，位于语句的后面
    print(i)
#2.如果多行注释，则要用"""    """
3.函数的注释，三个双引号
def func(*arg,*avrg):
    """函数功能描述
    :a  参数
```

```
    :b  参数
    :c  参数
    return  返回值说明
    """
4.类注释
class Foop(object):
    """对这个类进行注释和描述"""
5.文件的注释：写于文件头部，"""内容"""
6.对文件夹的注释
__init__.py文件中，"""当前文件夹是做什么用的"""
```

TODO注释

下一步做什么？

```
def func():
    for i in range(10):
        print(i)
        #TODO 下一步做什么
        #
        #
```

2.08 pycharm的虚拟环境

2.09 常见的数据类型

变量及数据类型

```
>>>a="你好，世界"#定义一个变量，并赋值
>>>print(a)
```

数据类型

在python中数据都有各自对应的类型

1.数字类型： 整型 int 浮点型 float 复数complex

```
>>>print(45)  #整型
>>>print(3.14159265358)#浮点型
>>>print((-1)*0.5)#复数
```

2.字符串string

一段普通的文字，由引号引起来

```
>>>print("今天是个好天气，天天好风光")
>>>print("56")
```

3.布尔bool

真和假，True　False　 1,0(注意首字母大写)

```
>>>print(4>3)
>>>print(5<10)
```

4.元组tuple

5.列表list

```
names=["jack","tony","david","alex","paul"]
```

6.字典dict

```
nums={'name':"jack",'age':"29"}
```

7.集合set

```
car={'buick','3','name'}#自动去重
```

2.10查看数据类型

```
>>>a=34
>>>b='hello'
>>>c=Ture
>>>d = ['2','6']
>>>type()#查看数据类型，使用type内置类
```

在python中，变量是没有数据类型的，我们所说的变量的数据类型，其实是变量对应的值的数据类型。

2.11 标识符的命名规则与规范

规则

1. 由数字，字母及下划线组成，但不能以数字开头。
2. 严格区分字母大小写
3. 不能使用内置关键字

关键字，在python语言里，有特殊含义的单词

规范

建议遵守 PEP 8

遵守一定的命名规则。

两种命名方法：

1. 大驼峰，类名使用，每个单词的首字母大写
2. 小驼峰，第一个字母小写，后面每个单词的首字母都大写
3. 字母小写加下划线或数字

2.12 print语句的使用

在pycharm中，将光标点在print上，按住ctrl键，点击。重点理解几个参数的使用

```
def print(self, *args, sep=' ', end='\n', file=None): # known special case of
print
    """
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.   # 这个位置，打印结果
隔开使用符号，一般为空格，也可以填入其他的符号
    end:   string appended after the last value, default a newline.#末尾一般为换
行，也可以填用其他符号
    flush: whether to forcibly flush the stream.
    """
    pass
```

2.13 input语句的使用

输入语句的使用

python里，使用内置input函数来接收用户的输入,不管用户输入的是什么，保存的都是字符串，需要时，要对类型改变。

```
>>>pword=input("请输入你的银行卡密码：")
>>>age=int(input("你的年龄："))
```

day03

3.01 教辅系统的使用

3.02 使用系统环境变量

pycharm安装位置，program files中 jetbrain bin文件夹中，创建快捷方式。

如何使用pycharm来创建新的项目。

3.03 不同进制数据的表示方法

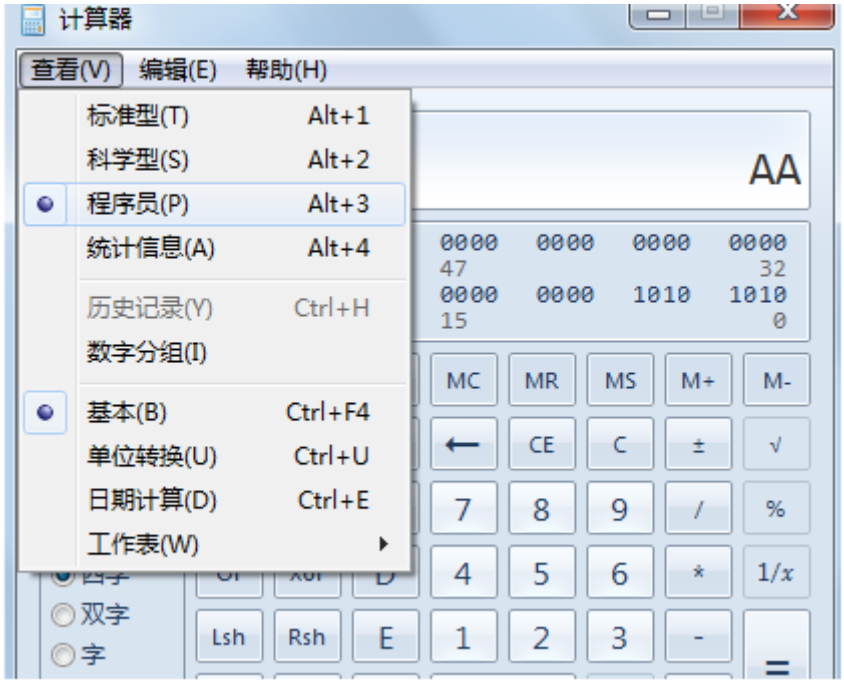
数据的类型

整型int

整数，计算机内只能保存0和1，生活中我们用的是十进制。最大单个数字为9，二进制中最大1，同时计算机也支持八进制和十六进制。

二进制，八进制，十六进制

```
>>>a=98  #十进制98

>>>b=0b010101010  #以0b开头的是二进制

>>>print(b)#当用print打印一个数字的时候，打印出的结果也是十进制

>>>c=0034#以0o开头的为八进制，最大个位数7

>>>d=0x25#以0x开头的为十六进制，最大个位数为F  0--9  a b c d e f
```



python2和python3的区别

八进制数据表示方式，0o开始，还可以0开始的数字，也是八进制，python3中八进制只能以0o开头。

进制间的转换

浮点型float

复数complex

字符串string

元组tuple

列表list

字典dict

集合set

3.04 十进制转为二进制

十进制转换为二进制

128 64 32 16 8 4 2 1　一直除以2，余数

小学学的短除法，倒排序

3.05 二进制转八进制十六进制

程序中进制转换的不多

```
>>>a=23
>>>a=0b111110101#一个二进制是一位，一个byte,一个字节是八位B，KB,MB,GB

#三个一组，转成八进制0BB27
#四个一组，转成十六进制0X17
```

3.06 使用内置函数实现进制转换

代码进行进制转换

```
a=12
bin(a)--->转成二进制#内置函数bin转换成二进制
oct(a)--->转换成八进制#oct
hex(a)--->转换成十六进制#hex
```

3.07 为什么要类型转换

数据类型的转换

不同的数据类型进行运算时，运算规则是不同的。

```
#进制转换是把int类型用不同的进制表现出来
#类型转换，是把一个类型的数据转换成其它类型的数据
#int-->str   str---->int   int--->float
>>>age=input("请输入你的年龄：")
>>>print(age+1)----->#会报错，input接收到的是字符串类型，不能进行加法运算，不同类
改正为：
>>>age=int(input("请输入你的年龄："))#使用内置类可以将其他类型的数据转换成为整数
```

3.08 转换为整数

使用int内置类，可以将数据转换为整数。

```
>>>a='31'
>>>b=int(a)#转换成整数型
>>>print(a)#字符串
>>>print(b)#整数
============================================================
>>>x="hello"
>>>y=int(x)
>>>print(y)------>会报错，如果字符串不是一合法的数了，会直接报错。
============================================================
>>>x='1a2c'
>>>y=int(x，16)----》当作16进制
>>>print(y)-------->正常打印结果，十进制输出
```

3.09 转换为字符串和浮点数

```
#转换成浮点数,使用内置类，可以将其经数据转换为浮点数
a='12.34'
b=float(a)--------------->     12.34
================================================
c=float("hello")
print(c)---------->会报错
================================================
 c=101
 print(float(c))------->101.0
================================================
a="12"
float(a)----->将字符串转换为浮点数
float(12)------》将整型转换为浮点数，和上面的过程是不同的。
```

```
#使用str类，将其他数据转换为字符串
a=34
b=str(a)
print(b)
print(type(a))-------->数字34
print(type(b))-------->字符串34
```

3.10 转换为布尔值

使用bool内置函数可以将其他数据类型转换成布尔值

在python中，0,空字符串，空元组，空列表，空字典，空集合set，转换为bool型，为False

python中，True 各False数字表示为1,0来使用。

```
>>>print(bool(100))#转换成布尔值
>>>bool(-1)------->True
>>>bool(0)--------->False #数字里，只有0转换，为False
==================================================
>>>bool("hello")----->True#字符串
>>>bool("False")----->True
>>>bool("")----------->False#空字符串为 False
==================================================
>>>bool(None)----------->False
==================================================
>>>bool([])------------>False
>>>bool([1,2])---------->True
==================================================
>>>bool(())---------->False
>>>bool((1,2))------->True
==================================================
>>>bool({})---------------->False
==================================================
```

隐式类型转换

```
if 3>2:
    print("hello")---------->hello
====================================
if 3:
    print("hello")----------->hello
====================================
```

3.11 算数运算符的基本使用

运算符

算术运算符

+-*/ // %**

加减乘除，幂，求余，整除

```
print(1+2)---->3
print(4-2)---->2
print(3*2)---->6
print(3**2)--->3*3
print(6/2)---->3.0
print(9/2)---->4.5#两个整数相除，得到结果有可能是浮点数。
print(81 ** 1 / 2)
print(81 ** (1 / 2))
print((10 / 3))
print(10 // 3)
print(10 % 3)
print(1928 % 9826)
```

除法运算符在2和3中区别，2 中是整数，3中结果是浮点数。

3.12 字符串里的算数运算符

```
"""
字符串里的算术运算符
有限
加法+
乘法*
"""
# 加法运算符，只能用于两个字符串的数据类型，用来拼接两个字符串
print("hello"+"world")------>

#用来将一个字符串重得多次
print("hello"*3)--------->
```

3.13 赋值运算符的使用

```
"""赋值运算符，在计算机编程中，”=“称为赋值运算符
和数学中的=是有区别的:
1+1=2
4=4
在计算机中，等号的作用是，将等号右边的值赋给等号左边，从右向左。
4=b则不行。
"""
# 等号的左边一定不能是常量，表达式  3+3=m 则不可

a = 3
b = 3 + 3 + 2
# ====================高级点的复合赋值运算符
c = 1
c = c + 1

c += 1
# ====================还有加减乘除
```

3.14 赋值运算符的特殊场景

```
"""
赋值运算符的特殊使用场景
"""
# 等号的连接可以传递数据值
a = b = c = d = "hello"
print(a, b, c, d)
================================================================================
# 以下写法错误
# x="yes"=y=z
================================================================================
m, n = 3, 5  # 拆包，相当于是一个元组

print(m, n)

x = "hello", "world", "good"
print(x)
================================================================================
#  y, z = 1, 2, 3, 4, 5 #给的值太多，不能拆包 too many values to unpack (expected
2)

================================================================================

o,p,q=4,2  #not enough values to unpack (expected 3, got 2)
print(o,p,q)
================================================================================
o,*p,q = 1,2,3,4,5,6,7 #加了*为可变量，赋值式就成立了
print(o,p,q)------->1 [2, 3, 4, 5, 6] 7
```

3.15 比较运算符的使用

```
"""
比较运算符
大于，大于等于  >   >=
小于，小于等于  <   <=
不等于  !=
等等于  ==
"""
print(2>1)
print(2<7)
print((4 <= 9))
print((5 != 6))
print(("hello" == "hello"))

# 比较运算符在字符串中的运用,会根据各个字符的编码值来逐一比较
# ascii码表
print(('a' > 'b'))---->  97  98
print("abc">'b')----->

print("a">90)------>报错
# 数字和字符中之间，做==运算的结果是False，做！=结果是True，不支持其它运算
```

3.16逻辑运算的基本使用

```
"""
逻辑运算符
与 and
或 or
非 not

"""
# 只要有一个运算数是False，结果就是False，只有所有的运算数都是Ture，结果才是Ture
print(2 > 1 and 5 > 3)
print(3 > 2 and 5 < 4 and 6 > 1)
# 只要有一个是Ture，结果就是Ture，只有所有的是False，结是才是False

print(3 > 9 or 10 < 3)
print(3 > 5 or 4 < 2 or 8 > 7)

#非运算，Ture变成False，False变成True
```

3.17逻辑运算的短路和取值

```
"""
逻辑运算的短路

"""
# 逻辑运算与，只有所有的都是True，结果才为True
# 只有一个是False，则False

4 > 3 and print("hello world")
4 < 3 and print("你好世界")

# 逻辑或运算，只有都是False，才是False
# 只要有一个是True，结是术是True

4 > 3 or print("hahahaha")
4 < 3 or print("ohohoh")

# 逻辑运算的结果一定是布尔值么？

print(3 and 5 and 0 and 'hello')#false
```

3.18 位运算（了解）

用的不多，了解。

```
"""
```

```
按位与 &
按位或  |
按位异或 ^
按位左移 <<
按位右移 >>
按位取反 ~
"""
======================
a = 23
b = 15
print(a & b)
print(a | b)
print(a ^ b)
=======================
x = 5
print(x << 2)  # 左移2位，在右边数字位补2个0
y = 15
print(y >> 2)  # 会丢失数据
```

day04

4.01 回顾总结

不能止步于听懂，要会动手才成。

```
"""
color=0xF0384E

F0   38    4E


"""
color = 0XF0384E
red = color >> 16
print(hex(red))

green = color >> 8 & 0xff
print(hex(green))

blue = color & 0xff
print(hex(blue))
```

4.02 运算符的优先级

幂运算

```
# 逻辑运算符 not > and > or
print(True or False and True)
print(False or not False)
print(True or True and False)
# 强烈建议，在开发中，使用括号来说明运算符的优先级。
print(True or (True and False))
```

逻辑与运算

只要有一个为false,结果为false,只有所有的运算数都是TURE,结果才是true

短路：只要遇到了false，就停止了，不再继续运行。

取值：取第一个false,如果都是true，取最后一个值

逻辑或运算

只要有一个运算数为true，结果就是true，只有所有的运算数为false，结果才是false

短路：只要遇到了true，就停止，不再继续执行了

取值:取第一个为true的值，如果所有的运算数都是false，取最后一个运算数。

4.03 作业评比

4.04 if....else..的使用

条件判断语句

```
"""
条件判断语句
if
if else
if elif elif  else

"""
================================================================================
age=int(input("请输入你的年龄："))

if age < 18: #字符串和数字做比较，==结果是False，！=结果是true，其它比较会报错。
    print("未18岁，禁止入内。")

else:
    print("you can get in.")
```

4.05 条件判断语句练习

```
score=float(input("请输入你的成绩>>>"))

if score > 60:
    print("ok,你已合格了。")
else:
    print("你出局了。")
```

4.06 if ...elif...语句使用

```
score=float(input("请输入你的成绩>>>"))

if score >= 90:
    print("你真棒，成绩优。")
elif score >= 80:
    print("成绩良，再接再励。")
elif score >=60:
    print("恭喜你，通过。")
else:
    print("你未通过是低合格线，失败了。")
```

4.07 if语句嵌套

```
# 买票进站

# 是否进入候车大厅

ticket = input("确定是否已买票，y/n:")

if ticket == "y":
    print("你已购买车票，请进入大厅，等待安检。")

    safe = input("安全检查是否通过，y/n:")
    if safe == "y":
        print("安全检查通过，请进入发车区域等待。")
    else:
        print("安全检查未通过，请您进一步检查所带物品是否安全。")

else:
    print("请您先至售票口购买所要乘坐的车票后，再至大厅，等待安检。")
```

4.08 pass关键字使用

pass关键字的作用，占位

没有其它的意义，在程序未完成的语句情况下，可以让程序继续运行下去，多出现在if　for　while 语句内。

4.09 猜拳游戏

```
"""
猜拳游戏
三种姿势 ：0:剪刀　1：石头　2：布
思路是：
    电脑随机一种
    人员要求输入自己的选择
    电脑对比后
    得出结论：输　赢
"""
import random

computer = random.randint(0, 2)

print(computer)
```

```
player = int(input("请输入你要伸的手势，0:剪刀  1：石头  2：布  >>>"))#注意问题，比较的数字类型。

if (player == 0 and computer == 1) or (player == 1 and computer == 2) or (player == 2 and computer == 0):
    print("电脑胜了。")
elif player == computer:
    print("小子，真幸运，咱俩平局。")
else:
    print("玩家胜了。")
```

4.10 if语句的注意点

1. 隐式类型的转换

4.11 while语句的基本使用

loop 循环，生活中的，同一件事情不停的做，一直不停的做。

```
"""
python中的循环

1.while  有条件的，有结束标识的，如果无，则会死循环。

2.for


不支持do ...while...

"""

print("hello\n"*50)
print("good")
# ======================================
str1="hello"
str2="good"
print(str1*50,str2*50)
# ======================================
# while的使用，条件
i=0
while i<50:
    print(str1,str2)
    i+=1  #缩进这一段内的变量使用，python里没有i++这种表示。
```

4.12 while语句的练习

4.13 for..in循环的使用

```
"""
for in  循环
1.指的是for ...in...循环
2.格式：  for x in iterable:(后同是可迭代的数据）
        pass
3.range 内置类，用来生成指定区间的整数序列
4.目前可迭代对象：字符串，列表，字典，元组，集合，range
```

```
"""
for i in range(0,11):#注意数据的后面必须是一个可迭代对象
    print(i)
================================================================================
z=0   #求0--100 相加之和
for i in range(0,101):
    z+=i
print(z)
```

4.14 break和 continue的使用

```
"""
break和continue
两个只用于循环语句中
break
    用来结束整个循环
continue
    用来结束本轮循环，开启下轮循环

"""
i=0
while i<5:
    if i==3:
        i+=1
        break#continue结束本轮循环，进入下一轮
    print(i)
    i+=1
```

4.15 嵌套打印矩形

```
"""
嵌套循环

"""
j = 0
while j < 5:  # 控制行
    j += 1

    i = 0
    while i < 5:  # 控制列
        i += 1
        print("*", end="  ")
    print()  #用来换行的
============================================================
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
```

4.16 嵌套打印三角形

```
j = 0
while j < 9:  # 控制行
    j += 1

    i = 0
    while i < j:  # 控制列  改成i<j,通过分析*变化，找规律。
        i += 1
        print("*", end=" ")
    print()
```

```
==========================================================
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
```

4.17 打印九九乘法表

```
"""
打印九九乘法表
主要思路：
从打印星号开始
1.先打钱矩形
2.调整打印三角形
3.打印乘法表
"""

j = 0
while j < 9:  # 控制行
    j += 1

    i = 0
    while i < j:  # 控制列  改成i<j,通过分析*变化，找规律。
        i += 1

        print(f"{i}*{j}={i*j}", end=" ")
    print()
```

```
========================================================================
1*1=1
1*2=2 2*2=4
1*3=3 2*3=6 3*3=9
1*4=4 2*4=8 3*4=12 4*4=16
1*5=5 2*5=10 3*5=15 4*5=20 5*5=25
1*6=6 2*6=12 3*6=18 4*6=24 5*6=30 6*6=36
1*7=7 2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49
1*8=8 2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64
1*9=9 2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81
```

day05

5.01 基础题

5.02 进阶题

5.03 for...else语句的使用

5.04 使用假设成立法求质数

5.05 使用计数方法求质数

5.06 求斐波那契数列

5.07 挑战练习

5.08 字符串的表示方法

字符串，一段普通文本内容。

```
"""
 字符串的表示方法
 使用一对单引号，一对双引号
 或一对三双引号，一对三单引号
 注意：
 如果字符串里还有双引号，外面则使用单引号。
 如果字符吕里有单引号，则外面使用双引号。

"""
a = 'hello'  # 最常用的方式，用单引号
b = "hello"
c = """呵呵"""
d = '''嘿嘿'''
print(a, b, c, d)
m = 'xiao ming said: "i am xiao ming."'  # 双引号开始结束都是一样的，没法区分开始结束属于哪一对。
print(m)
# 三个引号的使用。
n = """
xiao ming said :"i am xiao ming."
"""
print(n)

# 字符串里的转义字符 \   表示转义，作用是对\后面的字符进行转义。
x = 'i\'m xiao ming.'
print(x)
y = "xiao ming said:\" i am xiao ming \""
print(y)
z = "xiao ming\n zhang san"  # 换行
print(z)

x1 = '你好\t\t世界'  # 制表符
```

```
print(x1)
x2 = 'good mor\\ning'  # 表示一个普通的反斜线
print(x2)
x3 = r'hello \teacher'  # 前面加r,表示原生字符 ， 大写的R 也可以
print(x3)
```

5.09 字符串的下标和切片

```
"""
字符串的的下标及切片
1.下标
    a.下标又叫索引，也是编号，相当于存储的位置，通过这个编号找到相应的存储空间。
    b.可迭代对象，str list tuple dict set range
    c.str list tuple可通过下标来访问下标。
在计算机里，下标都是从0开始的。
字符串是不可变的类型。对字符串的任保操作，都不会改变原有的字符串。
2.切片
    从字符串里，复制一段指定的内容，生成一新的字符串

"""
s = "hello world,it's great."

print(type(s))
for i in s:
    print(i)

print(s[3])
print('============================')
m = 'abcdefghijklmnopqrstuvwxyz'
print(m[5])  # m[index]===>获取指定下标的数据

# 切片的语法m[start:end:step] step  指的是步长，可以理解为间隔
print(m[2:9])   # 包含头，不含尾。包含start ,不含end.
print(m[2:])  # 如果只给开头，不给尾，则会一直到尾部。
print(m[:9])  # 从头开始，到所给的位置

print(m[3:15:2])  # 2在这里是步长，间隔，每隔2取1
print(m[3:15:1])  # 步长默认为1，步长不能为0，会报错。步长可以为负数，
print('======================')
print(m[3:15:-1])  # 没有数据
print(m[15:3:-1])  # ponmlkjihgfe
print(m[::]) # 从头到尾
rint(m[15:3:-1])  # ponmlkjihgfe
print(m[::])
print(m[::-1])  # 倒序
```

5.10 查找相关的方法

ctrl + 单击 点击左侧十字瞄准，展开

```
```

```
x = 'abcdefghijklmnopqrs'

# 获取字符串长度
print(len(x))

# 查找内容的相关方法 find index rfind rindex
print(x.find('l'))  # 找到下标
print(x.index('l'))
print(x.find('z'))  # 如果不存在，返回为负一
print(x.index('z'))  # 如果不存在，则会报错ValueError: substring not found
```

5.11 字符串查找判断和替换

```
"""
字符串的常见操作


"""

x = 'abcdefghijklmnopqrs'

# 获取字符串长度
print(len(x))

# 查找内容的相关方法 find index rfind rindex
print(x.find('l'))  # 找到下标
print(x.index('l'))
print(x.find('z'))  # 如果不存在，返回为负一
# print(x.index('z'))  # 如果不存在，则会报错ValueError: substring not found
print('-----------------------------------')
# 判断相关的 startwith endwith isalpha isdigit isalnum isspace
# is 开头是用来判断的，结果是一个布尔类型的。
print('hello'.startswith("he"))
print('hello'.endswith("o"))
print('hel56lo'.isalpha())
print('hello456'.isdigit())
print('1456'.isdigit())
print('3.1415926'.isdigit())  # 显示错误
print('hello2314321'.isalnum())  # 判断是否是由数这和字母组成的
print('helool321432'.isalpha())
print('   '.isspace())
'''
num = input("请输入一个数字：")
if num.isdigit():
    num=int(num)
else:
    print("你输入的不是一个数字，请重新输入。")
```

```
'''

# count 计数的，统计的

mystr = '今天天气真的很好，是个好天气，可以外出玩的开心了，好好噢'

print(mystr.count('好'))  # 统计有4个好字

#  字符串的替换replace 用来替换字符串

word = 'hello'
word1=word.replace('l', 'x')
print(word1)  # 原字符串是不可变的，
```

5.12 字符串分割相关的方法

```
"""
字符串的分割相关操作
split
rsplit
splitlines
partition
rpartition

"""
# split
x = 'zhangsan-lisi-david-herny-merry-jack-tony'
y = x.split('-')
y1 = x.split('-', 2)
print(y)  # y切割后便是一个列表。
print(y1)
print('-------------------------------------------------')
z = x.rsplit('-', 2)
print(z)
print('-------------------------------------------------')
# patition指定一个字符串作为分隔符，分为三部分，
# 前面   分隔符   后面
a = 'abcdXefghiXjklmn'
print(a.partition("X"))

file_name = '不要打开.mp3'
print(file_name.partition('.'))
```

5.13 快捷键的使用方法

```
"""
pycharm中的快捷键

双击 shift 会弹出全局搜索功能， Jetbrains 开发的工具还部有这个功能。
关闭这个功能。---->双击，打开全局搜索功能----》actions----->registry
打到ide.suppress.double.click.handler  选项，勾选。
```

```
ctrl + shift +a  同样可以打开全局搜索。

常见快捷键

    # 1.操作符两侧要加空格    PEP8,规范  -----》快速格式化代码  ctrl+shift +l
    # 2.快速复制粘贴选中代码  ctrl +d
    # 3.移动一行代码  ctrl+shift+上下箭头  alt+shift+上下箭头
    # 4.删除 ctrl+y  一行代码
    # 5.注释  ctrl+/
    key map中可以更改快捷键    format
"""
```

day06

6.01 字符串的常见操作

```
"""
6.01 字符串的常见操作
    大小写--只在英文中，中文中无用。
        capitalize
        upper
"""
# capitalize()第一个字母大写
print('hello the world,are you ok?'.capitalize())  # 第一个单词字母大写
print('hello world'.upper())  # 所有字母大写
print('Hello World'.lower())  # 所有字母变小
print('hello world'.title())  # 首字母大写
'''
content = input("请输入内容，输入exit退出：")
while True:
    print("你所输入的内容是：", content)
    if content.lower() == 'exit':
        break
'''
# ljust(width,fillchar)填充字符
print('Monday'.ljust(10,'#'))  # 占有十个字节长度，如果不够，后面加空格补齐。右侧

print('Tuesday'.rjust(12,'|'))  # 左侧加

print('Wednesday'.center(20,'*'))

#  strip()去空格左，右,左右去。
print('    strip    '.lstrip())
print('    strip   '.rstrip())
print('    strip    '.strip())

# 以某种固定格式显示的字符串，我们可以将它切割成为一个列表
x = 'zhangsan=lisi=jack=tony=chris=herry'
names = x.split('=')
print(names)

# 将列表变成一个字符串

fruit = ['apple','pear','peach','banana','orange','grape']
print('-'.join(fruit))
```

```
print('*'.join('hello'))
```

6.02 字符串的编码

```
"""
字符集
数据处理或存储数据时，只能识别0和1
文本要先转换成二进制，才能处理。8位一个字节
ascii码表 使用一个字符，最多只能表示128个
最大  1111  1111    255
最小  0000  0000    0

unicode
utf-8
字符和数字编码存在一个对应关系
使用函数chr和ord能够查看数字和字符的对应关系。

"""
print(bin(97))

print(ord('a'))  # 查找对应的数字

print(chr(65))  #  查找对应的字符

print(ord('你')) # 查打相应的编码
```

6.03 字符串的编码集

```
"""
字符串的编码集
# 使用chr 和 ord函数能够查看数字和字符的对应关系
# ord获对字符对应的编码
# chr根据编码获取对应的字符串

GBK国标，中国用，简体中文
BIG5  繁体中文
UTF-8 统一编码
    使用encode方法，可以将字符串转换成为指定编码集结果
    如果有一个编码集的结果，想把他转换成为对应的字符，使用decode方法

"""
# 如何将字符串转换成编码集
print('你'.encode('gbk'))
print('你'.encode('utf-8'))

x=b'\xe4\xbd\xa0'
print(x.decode())
print("============================")
y='你好'.encode('gbk')
print(y)
print(y.decode('GBK'))
print("============================")
y='你好'.encode('UTF8')
```

```
print(y)
print(y.decode('UTF8'))
```

6.04 in 或outin的使用

成员运算符

in 或 not in 的使用
用来判断一个内容在可迭代对象中是否存在

```
word = "hello"
x = input("请输入一个字符：")

# 判断用户输入的字符是否在字符串中存在
# for c in word:
#     if x == c:
#         print("你输入的内容存在。")
#         break
#     else:
#         print("你输入的内容存在")
# ========================================
# if word.find(x) ==-1:
#     print("你输入的内容不存在")
# else:
#     print('你输入的内容存在。')

if x in word:
    print('存在')
else:
    print('不存在')
```

6.05 格式化输出字符

```
"""
字符串的格式化输出
两种方式
1.百分号占位符 %nd打印n位。 %.nf保留小数点n位
2.
"""
name = 'zhangsan'
age = 18

print("大家好，我的名字是%s,我的年龄是%d" % (name, age))
print("大家好，我的名字是：", name, "我的年龄是：", age)
print('================================================')
print(f"大家好，我的名字是{name}，我的年龄是{age}")
print('================================================')
print("大家好，我的名字叫{}，我的年龄是{}大".format(name, age))
print('================================================')

print("大家好，我是%3d号男嘉宾" % 5)
```

```
print("大家好，我是%03d号男嘉宾" % 5)
print("大家好，我是%-3d号男嘉宾" % 5)

print("大家好，我是%03d号男嘉宾" % 15)

print('我今天挣了%.2f块钱。' % 3.151458255) #    特别注意此处，%.nf
```

6.06 字符串的format方法的使用

```
"""
字符串format方法
在字符串中也可以用{}占位
{}什么都不用写，会读取后面的内容，一一对应
"""
# 什么都不写，只占位，读取后面内容一一对应
print("我的名字叫{}，我今年{}岁了。".format('zhangsan', 19))  # 一 一对应的位置。
# 花括号内填写数字，数字从0开始，按位读取后面内容
y = '大家好，我是{1},我今年{0}岁了'.format(20, 'jeerry')
print(y)
# 花括号内填写参数，后面赋值
z = '大家好，我的名字叫{name},我来自{address},我今年{age}岁了'.format(age=18,
address='sandong', name='alex')
print(z)
# 混合式 {数字}{变量}
a = '大家好，我的名字叫{name},我来自{1},我今年{0}岁了'.format(18, 'sandong',
name='alex')  # 赋值变量的一定要写在后面
print(a)
# 什么都不写，{数字}不能混合使用。
# 数据拆包，传入列表数据，顺序不能变
d = ['zhangsan', 18, '上海', 180]
b = '大家好，我是{},我今年{}岁了，我来自{}，身高{}'.format(*d)
print(b)

# 字典拆包,顺序无所谓，通过KEY值来找的
info = {'name': 'chris', 'age': 20, 'addr': '北京', 'height': 190}

c = '大家好，我叫{name},我今年{age}岁了，我来自{addr}，我身高
{height}cm'.format(**info)
```

6.07 列表的基本使用

```
"""
列表的基本使用
[]

"""
name_list = ['zhangsan', '李四', '王五', 'jack', '张飞', '光羽', '马超', '王昭君']
# print(type(name_list))
# print(help(name_list))
```

```python
# 可迭代对象
names = list(('李四', '王五', 'jack', '张飞', '光羽', '马超', '王昭君'))
print(names)
# 有序，可以使用下标获取对应元素,切片，及通过下标修改元素。
print(names[2])
print(names[:3])

names[3]="eric"
print(names)
```

6.08 列表增加元素

```python
"""
列表增加元素
列表是用来保存多个数据的
列表操作
    增加
    删除
    修改
    查询

"""
heros = ['阿珂', '嬴政', '后羿', '李元芳', '亚瑟']

# 添加元素的方法append insert extend
heros.append('黄忠')  # 在列表的最后面追加一个元素
print(heros)

heros.insert(1, '女娲')  # 索引之前插入一个对象
print(heros)

heros.extend(('马可波罗', '狄仁杰'))  # 可迭代对象元组，列表
print(heros)

heros.extend(['马可波罗', '狄仁杰'])
print(heros)
```

6.09 列表的修改查询和删除

```python
"""
列表的修改查询和删除

"""
heros = ['阿珂', '嬴政', '后羿', '李元芳', '亚瑟','马可波罗','狄仁杰']


# clear remove pop三个相关方法

# x=heros.pop()  # 有返回值，默认删除最后一个数据
# print(x)  # 结果返回这个数据
```

```
# y= heros.pop(2)  # 指定位置上的删除
# print(y)
# print(heros)

# remove删除

# heros.remove('亚瑟') #删除指定元素
# print(heros)
#
# heros.remove('妲己')  # 如果数据不在列表中，则会报错
# print(heros)

# clear
# heros.clear()
# print(heros)     # 清空了

# del
# del heros[1]
# print(heros)

tanks = ['亚瑟','张飞','程咬金','盾山','廉颇']

print(tanks.index('盾山'))
# print(tanks.index('庄周'))  # 如果元素不存在，则直接报错

# count计数
print(tanks.count('张飞'))

# in 运算符
print('张飞' in tanks)
print('苏烈' in tanks)

# 修改元素，通过下标，直接修改
tanks[2]='铠'
print(tanks)
```

6.10 列表的遍历

```
"""
列表的遍历
将所有的数据都访问一遍，遍历针对的是可迭代对象
两种：
    for....in...
    while...

"""
tanks = ['亚瑟', '张飞', '程咬金', '盾山', '廉颇']

for i in tanks:  #  本质是不断的调用迭代器的next方法查找下一个数据。
    print(i)

print('----------------------------------------')
```

```
count = 0
while count < len(tanks) - 1:  # 注意范围
    count += 1
    print(tanks[count]) # 通过下标来访问每一个数据

print('----------------------------------------')
```

6.11 交换两个变量的值

交换两个变量的值

```
"""
交换两个变量的值

"""
a=13
b=20
print('-'*30)  #方法一
c=a  # 找个中间变量
a=b
b=c
print(a,b)
print('-'*30)
a,b=b,a  # python中特有的方式方法二
print(a,b)

print('-'*30)  # 方法三
a=a+b
b=a-b
a=a-b
print(a,b)
```

6.12冒泡排序

```
"""
冒泡排序
让一个数字和它相邻的下一个数字对比，交换位置
"""
nums = [6, 5, 3, 1, 8, 7, 2, 4]
# 思路，先找出一个数字的比较，然后再做一次比较，直到排序成功

i = 0
while i < len(nums) - 1:  # 每次比较得到一个最大排序的数。
    i += 1

    n = 0
    while n < len(nums) - 1:  # 比较得到最大一个数

        if nums[n] > nums[n + 1]:
            nums[n], nums[n + 1] = nums[n + 1], nums[n]
            # print(nums)
```

```
        n += 1

print(nums)
```

6.13 列表的排序和反转

```
"""
列表的复制
可变数据类型

"""
# 直接对原有列表进行排序，sorted()内置函数
nums = [6, 5, 3, 1, 8, 7, 2, 4]
nums.sort(reverse=True)  # 调用列表sort方法，直接排序。
print(nums)
# 二者之间有区别sorted不会改变原有的数据列表，会生成一个新的列表
x = sorted(nums)
print(x)

names=['zhangsan','lishi','wangwu']
names.reverse()
print(names)

print(names[::-1])
```

6.14 可变数据类型和不可变数据类型

可变数据类型

不可变数据类型

```
"""
可变数据类型和不可变数据类型
数据都是存在内存里的
1.可变数据类型：列表，字典，集合，内存地址不会发生改变
2.不可变数据类型：  字符串，数字，元组，如果修改值，内存地址会发生改变。
"""
a=10
b=a

print('修改前，a={},b={}'.format(id(a),id(b)))
a=6
print(b)
print('修改后，a={}，b={}'.format(id(a),id(b)))
================================================
x=[100,200,300]
y=x
print(f'修改前a={id(x)},b={id(y)}')
```

```
x[0]=1
print(y)
print(x)
print(f'修改后a={id(x)},b={id(y)}')
================================================

修改前，a=8791472697904,b=8791472697904
10
修改后，a=8791472697776，b=8791472697904
----------------------------------------
修改前a=1790536,b=1790536
[1, 200, 300]
[1, 200, 300]
修改后a=1790536,b=1790536
# 指向相同的空间地址。
```

6.15 列表的浅复制

```
"""
列表的复制

"""
x = [100, 200, 300]
y = x
z = x.copy()
print(id(x), id(y), id(z))  # x,y地址相同，z指向另一空间，但内容相同
print('-----------------------------------')
x[0] = 1
print(x, y, z)
print(id(x), id(y), id(z))
print('-----------------------------------')
# copy模块
import copy

a = copy.copy(x)
print(a)
# 切片就是一个浅拷贝
names1 = ['zhangsan', 'lishi', 'wangwu', 'liuqi']
names2 = names1[::]
names1[0] = "alex"
print(names1, names2)
print(id(names1), id(names2))
================================================

30298696 30298696 30299208
-----------------------------------
[1, 200, 300] [1, 200, 300] [100, 200, 300]
30298696 30298696 30299208
-----------------------------------
[1, 200, 300]
['alex', 'lishi', 'wangwu', 'liuqi'] ['zhangsan', 'lishi', 'wangwu', 'liuqi']
40625096 40625608
```

day07

7.01 冒泡排序优化

```
"""
冒泡排序的优化

"""

# names = ['zhangsan', 'lisi', 'wangwu', 'liuqi', 'erci', 'alex']
#
# new_name = input("请输入一个名字>>>")

# if new_name in names:
#     print("用户名已存在。")
# else:
#     print("新的用户名，添加中...")
#     names.append(new_name)

# print(names)

# for name in names:
#     if new_name==name:
#         print("用户名已存在。")
#         break
# else:
#     print("新用户名，添加中...")
#     names.append(new_name)
# print(names)
# 统计列表中出现次数最多元素


# 冒泡排序完善
nums = [6, 5, 3, 1, 8, 7, 2, 4]
count = 0
j = 0
# 第一趟比较时，j=0,多比较了0次
# 第二趟比较时，j=1多比较了1次
while j < len(nums) - 1:
    # 在每一趟定义个flag,作标识
    flag = True

    i = 0
    while i < len(nums) - 1 - j:
        count += 1
        if nums[i] > nums[i + 1]:
            flag = False
            nums[i], nums[i + 1] = nums[i + 1], nums[i]
        i += 1
    if flag:
        #     #  这一趟走完后，flag依然是True，说明这一趟没有进行过数据交换
        break
    j += 1
print(nums)
print(f'比较了{count}次')
```

```
# 求列表里的最大数

# 删除列表里的空字符串
```

7.02 求列表的最大数及坐标

```
"""
求列表的最大数以及下标

"""
# 求列表中的最大数
nums = [3, 1, 9, 8, 4, 2, 0, 7, 5]
# nums.sort()
# print(nums[-1])
#
# nums.reverse()
# print(nums[0])
#
# nums.sort(reverse=True)
# print(nums[0])

# x = nums[0]
#
# for num in nums:
#     if num > x:  # 如果发现列表存在比假设的大
#         x = num  # 说明假设不成立，要交换数据
# print("发现的最大数为", x)

i=0
x=nums[0]
while i<len(nums)-1:
    if nums[i] > x:
        x = nums[i]
    i += 1

print("发现的最大数为",x)
```

7.03 移除空字符串

```
"""
移除空字符串

"""

x = ['hello', 'good', '', 'yes', 'no', '']
# 在for in循环中，最好不要对列表元素进行增删操作
# for word in x:
#     if word =='':
#         x.remove(word)
#
```

```
# print(x)
i = 0
while i < len(x):
    if x[i] == '':
        x.remove(x[i])
        i -= 1
    i += 1
print(x)
==================================================
words2 = []
for word in x:
    if word != '':
        words2.append(word)
words = words2
print(words)
```

7.04 列表的嵌套

```
import random

"""
列表的嵌套
三个房间，8个老师，随机分配工位
"""

teachers = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
rooms = [[], [], []]

for teacher in teachers:
    room = random.choice(rooms)
    room.append(teacher)

print(rooms)
# 打印下第0个房间
for i in range(len(rooms)):
    print(f'第%s个房间的人员共有%d人' % (i, len(rooms[i])))

for i, room in enumerate(rooms):
    print('房间%d里一块有%d个老师' % (i, len(room)),end='\t')
    for teacher in room:
        print(teacher,end='\t')
```

7.05 列表推导式的使用

```
"""
列表推导式
使用简的语法创建一个列表

"""
nums = [i for i in range(10)]
```

```
print(nums)

nums = []
for i in range(10):
    nums.append(i)
print(nums)
print('----------------------------')

nums=[i for i in range(10) if i%2==0]
print(nums)
print('----------------------------')
nums=[i for i in range(10) if i%2]
print(nums)
print("----------------------")
x=[]
for i in range(10):
    if i%2:
        x.append(i)

print(x)

points=[(x,y) for x in range(5,9) for y in range(10,20)]

print(points)
=========================================
C:\Users\Administrator\AppData\Local\Programs\Python\Python37\python.exe
C:/Users/Administrator/PycharmProjects/pythonProject/千峰/day7/05列表推导式.py
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
----------------------------
[0, 2, 4, 6, 8]
----------------------------
[1, 3, 5, 7, 9]
----------------------
[1, 3, 5, 7, 9]
[(5, 10), (5, 11), (5, 12), (5, 13), (5, 14), (5, 15), (5, 16), (5, 17), (5,
18), (5, 19), (6, 10), (6, 11), (6, 12), (6, 13), (6, 14), (6, 15), (6, 16), (6,
17), (6, 18), (6, 19), (7, 10), (7, 11), (7, 12), (7, 13), (7, 14), (7, 15), (7,
16), (7, 17), (7, 18), (7, 19), (8, 10), (8, 11), (8, 12), (8, 13), (8, 14), (8,
15), (8, 16), (8, 17), (8, 18), (8, 19)]
```

7.06 列表推导式的练习

```
"""
推导式练习

"""

nums = [i for i in range(1,101)]
print(nums)
x = []
# 分隔nums,
# 尝试2个一组，再转成3个一组
```

```
m=[nums[x:x+2] for x in range(0,100,2)]
print(m)

n=[nums[j:j+3] for j in range(0,100,3)]
print(n)
```

7.07 深拷贝和浅拷贝的介绍

```
"""
列表的浅拷贝和深拷贝

"""
nums=[1,2,3,4,5,6,7,8,9]
nums1=nums
nums2=nums.copy()

import copy
# nums3=copy.copy(nums1)
# nums4=copy.deepcopy(nums1)
# print(nums3,nums4)
# nums1[0]=1000000
# print(nums2,nums4)

words=['hello','good',[100,200,300,400],'alex','eric','bob']

words1=words.copy()
print(words1)
x=print(copy.deepcopy(words))
words[0]='你好'
print(words1)
words[2][0]=1
print(words,words1)
print(x)
```

7.08 元组的使用

```
"""
元组的使用
元组和列表很象，都是用来保存多个数据
（）表示
tuple
元组中的数据是不可变的。列表中的数据是可变的。
"""

words = ['hello', 'yes', 'good', 'hi']
nums = (9, 4, 3, 1, 7, 6)

# 和列表一样，是有序的储存数据的容器，可以通过下标来获取元素
```

```
print(nums[3])  # 可以获取，但不能改变数据，是不可变数据类型

# 只有2个查找
# nums.index()  # 得到元素的下标
# nums.count()  # 对其中的元素出现次数的统计

# 特殊情况 如何表示只有一个元素的元组？
# age = (18)  # 不是一个元组，是一个数字
age = (18,)
print(type(age))

# tuple  是内置类
print(tuple('hello'))  # ('h', 'e', 'l', 'l', 'o')

# 怎么样把一个列表转换成元组，或元组转换成列表

print(tuple(words))  # tuple list set 都可以如此使用
print(list(nums))

height=('133','147','023')
print('-----'.join(height))
print(''.join(('h', 'e', 'l', 'l', 'o')))
# 元组也可以遍历
for i in nums:
    print(i, end='\t')
```

7.09 字典的基本使用

```
"""
字典的基本使用
# 列表一般情况下可以存储任意类型
但一般情况下，我们通常都存储同一类型的数据

"""
# 这个列表里的每个元素代表什么，我们并不知道，这个值，我们无法进行描述

# person = ['zhangsan', 18, 98, 97, 95, 93, 180, 150]
# 字典不仅可以保存值，还可以对值进行描述
# 使用大括号来表示一个字典，不仅有值value，还有键值key,是以键值对形式来保留的，中间用冒号

person={'name':'zhangsan','age':18,
        'math':98,'chinese':93,'english':95,
        'gym':92,'height':180,'weight':150
        }
print(person)
```

7.10 字典使用时注意事项

```
"""
字典使用注意事项

"""
person = {'name': 'zhangsan',  'height': 180, 'age': 20, 'chinese': 93,
'english': 95,
          'gym': 92,  'weight': 150, 'hobbies': ['jump', 'rap']}
print(person)
# 1.字典里的KEY值不允许重复，如重复，后一个KEY值会覆盖前一个
# 2.字典里的value值可以是任意类型，但KEY值只能是不可变数据类型，一般使用字符串
```

7.11 从字典里获取数据

```
"""
从字典里获取数据
字典的增删改查

"""
person = {'name': 'zhangsan',  'height': 180, 'age': 20, 'chinese': 93,
'english': 95,
          'gym': 92,  'weight': 150, 'hobbies': ['jump', 'rap']}

# 查找 使用key得到数据,但不能根据value找到KEY
print(person['name'])
# print(person[age])
x='age'
print(person[x])
# print(person['heig'])  # 如果查找的KEY不存在，则会报错。

# 需求，获取一个不存在的KEY值 ，不报错，如果不存在，使用默认值。
# 使用字典的get方法
print(person.get('heig'))
print(person.get('gender', 'female'))
print(person.get('name', 'list'))
print(person)
```

7.12 修改和新增数据

```
"""
字典的修改和新增数据

"""
person = {'name': 'zhangsan', 'height': 180, 'age': 20,
          'chinese': 93, 'english': 95,
          'gym': 92, 'weight': 150,
```

```
          'hobbies': ['jump', 'rap']
          }
person['name'] = 'alex'  # 修改名字，数据的修改
print(person)
person['gender'] = 'male'  # 如果KEY不存在，则会在字典里添加一个新的VALUE
print(person)
================================================
{'name': 'alex', 'height': 180, 'age': 20, 'chinese': 93, 'english': 95, 'gym':
92, 'weight': 150, 'hobbies': ['jump', 'rap']}
{'name': 'alex', 'height': 180, 'age': 20, 'chinese': 93, 'english': 95, 'gym':
92, 'weight': 150, 'hobbies': ['jump', 'rap'], 'gender': 'male'}
```

7.13 字典的增删改操作

```
person = {'name': 'zhangsan', 'height': 180, 'age': 20,
          'chinese': 93, 'english': 95,
          'gym': 92, 'weight': 150,
          'hobbies': ['jump', 'rap']
# 删除
print(person.pop('name'))  # 删除并返回值
print(person)

print(person.popitem())  # 删除并返回删除值
print(person)
==============================
person.clear()   # 清空字典
print(person)
==============================
```

7.14update方法的使用

```
"""
update方法的使用
列表可以使用extend方法将两个列表合并在一起
"""
nums1 = [1, 2, 3, 4, 5, 6]
nums2 = [88, 66, 99, 77, 55]
nums1.extend(nums2)
print(nums1)

person = {'name': 'zhangsan', 'height': 180, 'age': 20,
          'chinese': 93, 'english': 95,
          'gym': 92, 'weight': 150,
          'hobbies': ['jump', 'rap']
          }
person1 = {'address': '山东省宁阳县'}
person.update(person1)  # 将两个列表合并成一个列表
print(person)
```

```
words1 = ('hello', 'good')
words2 = ('yes', 'no')
print(words2+words1)
# 字典不能用+
```

7.15 字典的遍历

```
"""
字典的遍历

"""
person = {'name': 'zhangsan', 'height': 180, 'age': 20,
          'chinese': 93, 'english': 95,
          'gym': 92, 'weight': 150,
          'hobbies': ['jump', 'rap']
          }
# for i in person:  #得到的是key
#     print(i)

for key, value in person.items():
    print(f'{key}:{value}')

# 特殊情况，列表和元组是单一的数据，字典是键值对保留的形式 key:value
print('-----------------------------------1')
# 第一种遍历方式
for x in person:  # for in 循环取出的的是key
    print(x, '=', person[x])  # x 一定不能加引号
print('-----------------------------------2')
# 第二种遍历方式,获取所有的key,再遍历key,根据key，再获取value
print(person.keys())  # 保存了所有的KEY

for k in person.keys():
    print(k, '=', person[k])
print('-----------------------------------3')
# 一二种方法一样，第二种相当于多了一步，取KEY值
# 第三种方式
for key,value in person.items():
    print(f'{key}:{value}')
print('------------------------4')
# 第四种 取value值
for v in person.values():
    print(v)
print('----------------------5')
#  整体来拿值
print(person.items())
for item in person.items():
    print(item[0],'=',item[1])
print('--------------------6')
# 拆包
for k,v in person.items():
    print(k,'=',v)
```

7.16 字典的练习

```
"""
字典的练习

"""
chars = ['a', 'b', 'd', 'p', 'q', 'a', 'd', 'd', 's', 'm']
# for char in chars:
#     print(char)

char_count = {}
for char in chars:
    char_count[char] = chars.count(char)

print(char_count)
vx = char_count.values()
print(max(vx))   # 取最大值
for k ,v in char_count.items():
    if v==max(vx):
        print(k)

# 方法2
print('--------------------')
for char in chars:
    if char in char_count:
        char_count[char] += 1

    else:
        char_count[char] = 1
print(char_count)
```

7.17 字典的练习2

```
"""
练习
"""
persons = [
    {'name': 'alex', 'age': 18},
    {'name': 'zhangsan', 'age': 25},
    {'name': 'lisi', 'age': 22},
    {'name': '王二麻子', 'age': 16}
]
x = input("请输入一个名字>>>")

for person in range(len(persons)-1):
    # print(persons[person])
    if persons[person]['name'] == x:
        print("输入的名字已存在，请重新输入")
        break
else:
    age = input('请输入年龄>>>')
    persons[person]['name']=x
    persons[person]['age']=age
```

```
print(persons)
```

7.18 字典的推导式

```
"""
字典推导式  ------>key value值相互转换
"""
dict = {'a': 100, 'b': 200, 'c': 300}
dict2 = {}

for k,v in dict.items():
    dict2[v] = k
dict = dict2
print(dict)

print('-------------------------1')
dict = {v:k for k,v in dict.items()}
print(dict)
```

day08

8.01 作业讲解1

8.02作业讲解2

8.03集合的基本使用

```
"""
集合的基本使用
    1.是一个不重复的无序数据
    2.使用{}  set来表示
        >>{}内放的是键值对，则是字典
        >>{}内如果放的是单个的词，则是集合


"""
name = {'zhangsan','lisi','wangwu','lisi','zhangsan'}  #自动去重
print(name)

# 如何增删改查----位置会改变，一般不用
name.add('阿珂')
print(name)
# name.clear()  # 清空集合
# pop  remove 方法
# pop随机删除一个数据
# remove删除指定的数据，如果没有数据，则会报错。
```

```
# union将多个集合合并生成一个集合
# a.update()将B拼接到A里
# 无序，没有查询方法。
# in是可以用的。
```

8.04集合里运算符使用

```
"""
集合里的高级使用方式

"""
a = {'张三', 'lisi', 'wangwu', 'mingming', 'lili', 'gaoyang', 'alex', 'eric',
'aiqi'}
b = {'mingming', 'lili', 'gaoyang', 'kuku', 'liuyu', 'xiangxiang'}

# print(sing+rap)  不及持加法

print(a - b)
print('----------------------------------------1')  # 交集
print(a & b)
print('----------------------------------------2')
print(a ^ b)
print('----------------------------------------3')

print(a or b)

print('----------------------------------------4')  # 并集
print(a | b)
```

8.05 eval和json的使用

```
"""
eval和json的使用

"""
# 去重,排序，全数字，自动排序
nums = [5, 8, 7, 6, 4, 1, 3, 5, 1, 8, 4]

x = set(nums)
print(x)
y = list(x)
print(sorted(x, reverse=True))

# 转换的相关方法list set tuple dict
nums = [9, 8, 4, 3, 2, 1, ]
x = tuple(nums)  # 转换成元组
print(type(x))
print(type(set(x)))
```

```
# eval有一个强大的内置函数，可以执行字符串里的代码
a = "input('请输入你的用户名>>>')"  # 字符串
# eval(a)  # 执行字符串里的代码
# json 把列表元组，字典，转换成json字符串json里面的字符串只能用双引号
#
person = {'name': 'zhangsan', 'age': 28, 'gender': 'male'}
# 转换成 '{'name':'zhangsan','age':28,'gender':'male'}'
import json

m = json.dumps(person)
print(m)
print(type(m))  # m变成了字符串，操作也只能按字符串来操作
# json.loads()  将一json字符串转换成python里的数据。
# python      json
# True        True
# 字符串       字符串
# 字典         对象
# 列表元组集合    数组
a_tupe = ('hello', 'good', 'ni', 'hao')
print(json.dumps(('hello', 'good', 'ni', 'hao')))
print(json.dumps(['hello', 'good', 'ni', 'hao']))
# print(json.dumps({'hello', 'good', 'ni', 'hao'}))---》集合不能转
```

8.06公共方法总结

```
"""
公共方法总结

算术运算符
+可以用来拼接，char tuple list
-只能用于集合，求差集
*可以用于字符串元组，列表，表示重复多次，不能用于字典和集合
成员运算符in可用字符串，列表，元组，字典，集合
都可以用for in 遍历
"""
# +可以用来拼接，char tuple list
print('hello' + 'good')
print([1, 2, 3, ] + [99, 88, 77, 55, ])
print({1, 2, 3, 5, 8, 9, } - {1, 8, 9})
print('------------------------------1')
# *可以用于字符串元组，列表，表示重复多次，不能用于字典和集合
print('hello' * 3)
print([1, 2, 3] * 3)
print((1, 2, 3) * 3)
print('------------------------------2')
# 成员运算符in可用字符串，列表，元组，字典，集合
print('a' in 'apple')
print(1 in [1, 2, 5, 9])
print('pen' in ('pen', 'book', 'rubber'))

print('------------------------------3')
# in用于字典来判断KEY值是否存在
print('name' in {'name': 'alex', 'age': 18})  # 当判断value值时，则会返回false
```

```
print('-------------------------------4')
print(3 in {3, 4, 5})
print('-------------------------------5')
# 带下标的遍历，元组，列表，集合，但集合是无序的，随机的
nums = [19, 82, 39, 12, 34, 58]
for x in enumerate(nums):  # 返回带下标的数组
    print(x)
print('------------------')
en = enumerate(nums)
for i, e in en:
    print(f'第{i}个数字是{e}')

person = {'name':'zhangsan','age':19,'adress':'泰山宁阳'}    # 打印出了KEY值
for k,v in enumerate(person):
    print(k,v)
```

8.07函数的介绍

```
"""
函数的介绍
一堆准备好的代码，需要时，随时调用
使用关键字，来声明一个函数

def 函数名（参数）：
    函数要执行的操作
    pass
函数定义好后，并不执行，在需要时调用
调用，直接调用函数名（参数）
"""
print('从前有座山')
print('山里有座庙')
print('庙里有个老和尚')
print('还有一个小和尚')
print('老和尚在给小和尚讲故事')
print('故事的内容是')

print('-------------------------')


# 重复性，可读性，维护性差
def story():
    print('从前有座山')
    print('山里有座庙')
    print('庙里有个老和尚')
    print('还有一个小和尚')
    print('老和尚在给小和尚讲故事')
    print('故事的内容是')


age = int(input("请输入年龄>>>"))
```

```
if 0 < age < 6:
    for i in range(5):
        print(f'--------------{i}')
        story()


else:
    for j in range(3):
        print(f'------------{j}')
        story()
```

8.08函数的参数

```
"""
函数的参数
定义  def 函数名（参数）
函数名（参数）

"""
# 函数声明时，括号内的参数我们称之为形参，形参用来占位的，是不确定的。

def story(place, person1, person2):
    # place = '庙'
    # person1 = '老和尚'
    # person2 = '小和尚'
    print('从前有座山')
    print('山里有座', place)
    print('庙里有个', person1)
    print('还有一个', person2)
    print(person1, '在给', person2, '讲故事')
    print('故事的内容是')
    print('----------------------')


# 调用函数时传递参数，才是真正参于运算的数据。称之为实参。

story('道观', '老道', '小道士')

#
story('niguan','shitai','xiaonigu')
# 什么时候定义参数？需要改变的地方，可以定义参数。
```

8.09函数的返回值

```
"""
函数的返回值
有的
没有的
定义函数需不需要返回值，根据实际需求来决定
"""
def sum(a,b):
    c=a+b
    return c


print(sum(2, 3))
```

8.10函数的文档说明

```
"""
函数的文档说明
1.基本使用
2.使用help()来查看函数使用说明
3.定义时，使用“”“”“来对函数说明，方便，自己和别人查看功能说明。
"""


def add(a:int, b:int):   # 在此建议类型
    """
    这两个函数用来将两个数字相加
    : a
    : b
    : return a+b

    """
    return a + b


print(add(3, 2))
help(add)
print('---------------1')
# 参数类型没办法规定
x=add(2,3)
y=add('hello','world')
print(x,y)
```

8.11函数调用函数

```
"""
函数调用函数

调试的断点使用
步进查看
"""
```

```
def text1():
    print("test1开始了")
    print("test1结束了")


def text2():
    print('test2开始了')
    text1()
    print('test2结束了')


text2()


# 求N到M之间所有整数之和
#
def add(n, m):
    x = 0
    for i in range(n, m):
        x += i
    return x


print(add(0, 101))
```

8.12全局变量和局部变量

```
"""
全局变量和局部变量

用的区域
    内置函数globals()
    locals()查看变量的使用域

"""
a = 100  # 全局变量，上下全文可用

#
# def book():
#     x = 'hello'  # 局部变量
#     print(x)
#
#
# book()

x = 'good'


def test1():
    x = 'hello'
    print('x={}'.format(x))
```

```
    a = 10  # 又声明了一个新的局部变量，如果和全局同名，选用时，会先调用局部变量，而不是去修改全局变量。
    print('函数的内部a={}'.format(a))

    global word
    word = 'ok'

test1()
print('函数的外部a={}'.format(a))
# 如何在函数内部修改外部变量

print(f'函数word变量的值为{word}')
print(locals())
print(globals())
#在python中，只有函数可以分割作用域
```

8.13多个返回值

```
"""
多个返回值

"""

def test1(a, b):
    x = a // b
    y = a % b
    return[x,y]
    return(x,y)
    return x, y  # 看到return就表示结束了，不可写多个 return，一般情况下只执行一个

result = test1(13, 5)
print(result)

# 特殊情况下，一个函数可能会执行多个return语句
```

day09

9.01 函数的回顾

```
"""
函数的回顾
1.声明    def 函数名（参数）：
              pass

2.函数的调用
3.函数的返回值，使用return 语句返回函数的执行结果
4.函数返回多个结果，将多个数据打包成一个整体tuple list dict
```

```
函数名也是标识符    数字，字母，下划线组成，区分大小写，不能使用关键字，尊守命名规范

"""
def get_sum(a,b):
    return a+b

x=get_sum(2,3)

print(x)

def calc(a,b):
    shang=a//b
    yushu=a%b
    return shang,yushu

m,n= calc(5,2)
print(m,n)
```

9.02 函数的缺省参数

```
"""
函数的缺省参数
要放在后面写
"""
def say_hello(name, age, addr='东北'):
    print(f'大家好，我叫{name}，我今年{age}岁了，我来自{addr}')


say_hello('alex', 19, )
say_hello('tony', 30, '山东泰山')  # 如果传了，则用传递的参数，如果没有传，则用默认的

# 有些函数的参数，如果你传递了参数，就使用传递的参数，如果没有传递参数，就使用默认的
print('hello world', end='')  # end就是默认缺省参数
print('hi')
print('hello world', 'nihao', sep="_____")  # 函数定义时，直接给个值就可以了
```

9.03 多个参数

```


def add_two(a, b):
    return a + b


print(add_two(2, 4))


# 任意数之和
```

```python
def add_many(mum):
    x = 0
    for n in mum:
        x += n
    return x

#
# nums = [1, 2, 3, 4, 5, 6, 7]
# print(add_many(nums))
# print(add_many((5, 8, 9, 6, 7, 5, 4, 2, 3, 4, 1, 4)))
# print(add_many({9, 5, 3, 4, 7, 8}))

# 一次input只能接收一次用户输入
nums=[]
while True:
    num=input("请输入>>>")
    if num =='exit':
        break
    nums.append(num)
```

9.04 可变参数

```python
"""
可变参数
*args  表示可变位置参数
**kwargs 表示可变的关键字参数
变量名，可以改，但不要改，习惯。
"""
print('hello')
print('good', 'alex')


# 可变参数
def add(a, b, *args,**kwargs):  # args 可变参数 多出来的关键字参数，会以字典的形式保存起来。
    print('a={},b={}'.format(a, b))
    print('args={}'.format(args))  # 多出来的可变参数会以元组的形式保存在args中


add(1, 2, 3, 4, 5, )


def add(*args):
    pass
```

9.05 可变类型和不可变类型的传参

9.06 函数的注意事项

9.07 递归的基本使用

```
"""
函数的基本使用

函数在使用过程中自己调自己。


"""
# count = 0  # 定义变量并赋初始值
#
#
# def tell_story():
#     global count  # 全局变量
#     count += 1
#     print('从前有座山')
#     print('山上有座庙')
#     print('庙里有个老和尚')
#     print('还有一个小和尚')
#     print('老和尚给小和尚讲故事')
#     print(f'----------------{count}')
#
#     if count < 3:  # 条件设置
#         tell_story()
#
#
# tell_story()

# 求1到N的和------》问题的关键是要有边界，什么情况下停止下来。

def get_sum(n):
    if n == 0:  # N为0时，停止递归，停止条件
        return 0
    return get_sum(n - 1) + n


print(get_sum(10))
```

9.08 递归的练习

```
"""
递归的练习
"""

# 使用递归求n!  n!=(n-1)!*n
# def factorial(n):
#     if n == 0:
#         return 1
#     return n*factorial(n-1)
#
#
# print(factorial(6))   # 6!=6*factorial(5)=6*5*factorial(4).....*1

# 使用递归求斐波那契数列的第n个数字

def fibonacci(n):
    if n == 2 or n == 1:
        return 1

    return fibonacci(n - 2) + fibonacci(n - 1)


print(fibonacci(5))
```

9.09 匿名函数的使用介绍

```
"""
匿名函数的使用价绍
创建:
    使用lambda关键词创建小型匿名函数，这种函数得名于省略了用def声明函数的标准步骤
    只包含一个语句
    lambda 参数列表：运算表达式

"""


def add_(a, b):
    return a + b


x = add_(4, 5)
print(x)

fn = add_  # 函数的存储位置，算是一个别名，指向同一地址
print(fn)
print(fn(2, 3))

# 除了def关键字定义外，还可以使用lambda来定义
lambda a, b: a + b

# 和上面表达示的效果一样的，两变量，返回和
# 更多的时候用来表达一个简单的函数，调用次数很少，只用一次，调用方式有两种
```

```
# 1.给他定义个名字
mul = lambda a, b: a * b
print(mul(3, 4))


# 2.这种方式用的多，把函数当作参数传给另一个函数使用。

def calc(a, b):
    pass
================================
def calc(a, b, fn):
    c = fn(a, b)
    return c


def add(x, y):
    return x + y


def minus(x, y):
    return x - y


x1 = calc(1, 2, add)
print(calc(8, 3, minus))

x3=calc(1,2,lambda x,y:x+y)
print(x3)
x4=calc(19,3,lambda x,y:x-y)
x5=calc(2,6,lambda x,y:x*y)
x6=calc(12,3,lambda x,y:x/y)

print(x3,x4,x5,x6)
```

9.10 sort方法的使用

```
"""
sort方法的使用

"""
# 有几个内置函数和内置类，用到了匿名函数
nums = [4, 6, 8, 2, 5, 5, 1]
# nums.sort()
# print(nums)

sorted(nums)  # 不会改变原有的列表，生成一个新的列表
x = sorted(nums)
print(x)

ins = (5, 6, 9, 8, 2, 1, 4, 5, 6, 7)
y = sorted(ins)  # 生成一个新的有序列表
print(y)
```

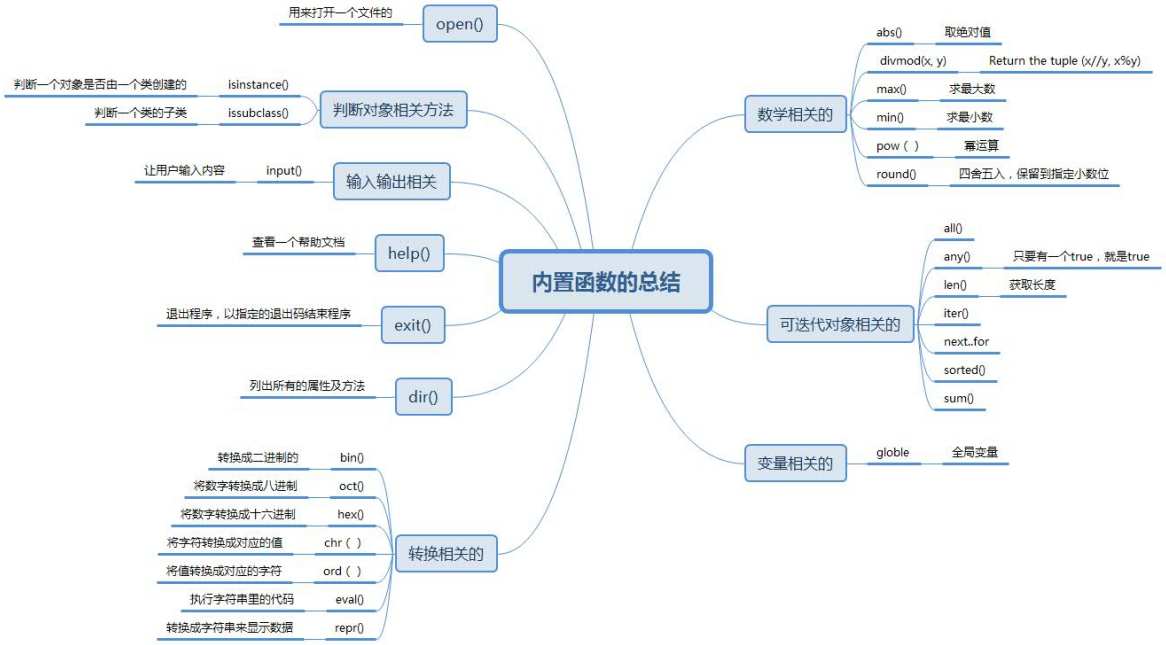
```
stu = [
    {'name': 'zhangsan', 'age': 18, 'score': 90, 'height': 188},
    {'name': 'alex', 'age': 20, 'score': 95, 'height': 168},
    {'name': 'xiaomi', 'age': 33, 'score': 60, 'height': 178},
    {'name': 'wangxi', 'age': 25, 'score': 88, 'height': 169},
    {'name': 'eric', 'age': 40, 'score': 40, 'height': 158},
    {'name': 'libai', 'age': 16, 'score': 99, 'height': 190},
    {'name': 'tony', 'age': 42, 'score': 100, 'height': 178},
]


# stu.sort()  # 报错，不能用于字典排序比较
# TypeError: '<' not supported between instances of 'dict' and 'dict'
# 缺少比较规则
# stu.sort(self,key,reverse)
# 需要传递参数KEY，指定比较规则
# key的参数类型是函数
def foo(ele):
    return ele['height']



stu.sort(key=foo)  # 在sort内部实现的时候，调用了foo方法，并传入了一个参数,参数就是列表里的元素字典

# TypeError: foo() takes 0 positional arguments but 1 was given 需要0个参数，但在调用时传入了一个参数
print(stu)

# 简化下
stu.sort(key=lambda ele:ele['score'])
print(stu)
```

9.11 filter&map&reduce方法

```
"""
filter()内置类的使用方法
过滤的作用，是内置函数，在 python3中，修改成了一个内置的类
  filter(function or None, iterable) --> filter object

    Return an iterator yielding those items of iterable for which function(item)
    is true. If function is None, return the items that are true.
对可迭代对象进行过滤，得到一个filter对象
需要2个参数，第一个参数是函数，第二个参数为可迭代对象，通过函数对可迭代对象过滤

"""
# ages = [13, 45, 63, 55, 23, 22, 14, 2, 5, 9, 8, 20]
#
# x = filter(lambda ele: ele > 18, ages)
# adult = list(x)
# print(adult)
# print(x)
```

```
# for a in x:
#     print(a)
# # map的使用
"""
class map(object):

    map(func, *iterables) --> map object

    Make an iterator that computes the function using arguments from
    each of the iterables.  Stops when the shortest iterable is exhausted.
    指定行为，
"""
# m = map(lambda ele: ele + 2, ages)
# print(list(m))
from functools import reduce

# reduce()以前是个内置类，现在变成一个模块内functools内
# builtins.py文件中
# scores = [100, 80, 70, 50]
# print(reduce(lambda ele, el1: ele + el1, scores))
#

# 使用过程 def foo(x, y):  # x=100,y=80;x=180,y=70;x=250,y=50;x=300
#       return x + y

stu = [
    {'name': 'zhangsan', 'age': 18, 'score': 90, 'height': 188},
    {'name': 'alex', 'age': 20, 'score': 95, 'height': 168},
    {'name': 'xiaomi', 'age': 33, 'score': 60, 'height': 178},
    {'name': 'wangxi', 'age': 25, 'score': 88, 'height': 169},
    {'name': 'eric', 'age': 40, 'score': 40, 'height': 158},
    {'name': 'libai', 'age': 16, 'score': 99, 'height': 190},
    {'name': 'tony', 'age': 42, 'score': 100, 'height': 178},
]
# 求所有学生的总年龄
# 1，先把年龄弄成一个列表
print(stu[0]['age'])
age = []
sum = 0
for i in range(len(stu) - 1):
    age.append(stu[i]['age'])
    sum += int(age[i])  # 把各项相加
print(age)
print(sum)


# 2把年龄相加得和
def bar(x, y):
    return x + y['age']
print(reduce(bar, stu, 0)

print(reduce(lambda x,y:x+int(y['age']),stu))
```

9.12 内置函数总结



9.13 高阶函数

day10