python2基础

- 条件判断语句：根据条件的成立与否，选择相应的逻辑分支
- 关键字：if...elif...else
- 最简格式
  - if 布尔表达式：
    - 待执行逻辑分支
  - 当布尔表达式结果为真，则执行待执行逻辑分支
-
- 对立条件判断
  - if 布尔表达式：
    - 待执行逻辑分支1
  - else：
    - 待执行逻辑分支2
-
- 多分支条件判断
  - if 布尔表达式1：
    - 待执行逻辑分支1
  - elif 布尔表达式2：
    - 待执行逻辑分支2
  - elif 布尔表达式3：
    - 待执行逻辑分支3
  - ...
  - else:
    - 以上条件均不成立时执行
  - **越特殊的情况越提前判断
  - 多条件分支只会执行一个分支
- *只有if关键字可以独立引领一个完整的条件判断逻辑链；else会和同层最近的if组成完整的逻辑链

```
44  i=int(input('请输入一个整数：'))
45  if i>100:
46      print('输入的数大于100')
47  if i==100:
48      print('输入的数等于100')
49  else:
50      print('输入的数小于100')
```

```
test01
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
请输入一个整数：130
输入的数大于100
输入的数小于100

Process finished with exit code 0
```

-

- 条件判断的扩展
  - 三目运算：三目运算是一个表达式，表达式的结果取决于布尔值的真假
    - 三目运算的格式：i = x if 布尔表达式 else y
      - 布尔表达式为真则值为x，否则为y
      - i=10 if 0 else 20
      - print(i)

```
51  i=10 if 0 else 20
52  print(i)
```

```
test01
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
20

Process finished with exit code 0
```

  - 多重条件判断：if语句中的待执行逻辑分支，也是一个条件判断语句
    - 内层if的条件判断一定是外层条件判断的扩展
      - 你大我更大，你小我更小
      - 两个毫不相干的条件组合

```
53  i=10
54  if i<100:
55      if i<30:
56          print(i)
```

```
test01
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
10

Process finished with exit code 0
```

- 根据三整数，判断该三个数能否组成三角形，如果能组成，输出三角形的类型
  - 提升：两边之和大于第三边：三角形的类型有：直角三角形、等腰三角形、等边三角形、等腰直三角、钝角三角形和锐角三角形

```
a=int(input('请输入第一个数:'))
b=int(input('请输入第二个数:'))
c=int(input('请输入第三个数:'))
#a,b,c=1,1,2**(1/2) #能组成等腰直角三角形
l=[a,b,c]
l.sort()
if l[0]+l[1]>l[2]:
    if l[0]**2+l[1]**2==int(l[2]**2):
        if l[0]==l[1]:
            print('能组成等腰直角三角形')
        else:
            print('能组成直角三角形')
    elif l[0]==l[1] or l[1]==l[2]:
        if l[0]==l[2]:
            print('能组成等边三角形')
        else :
            print('能组成等腰三角形')
    elif l[0]**2+l[1]**2<l[2]**2:
        print('能组成钝角三角形')#4 5 7
    else:
        print('能组成锐角三角形')#7 8 9
else:
    print('输入的数不能组成三角形')
```

- 循环语句
  - for循环：可以理解成为了达到某种目标的循环，目标达成循环停止
  - while循环：基于条件成立的循环，条件成立循环开始，直到条件不再成立
- **for循环：迭代变量，在集合中依次迭代，直到遍历完整个集合
- for循环的格式
  - for 变量 in 集合
    - 待循环的语句组
  - **变量从集合中第一个元素开始取值，每取值一次称为迭代一次，待循环语句组就执行一遍
  - 难点：变量可以参与到待循环语句组中，也可以不参与
    - 如果只是单纯的循环做某事，通常变量不参与；如果待循环的逻辑中需要这个变量，则变量需要参与
- 范围函数：range(x,y,z)
  - x,y,z分别代表起始值、终止值（不含）和步长
  - 如果只有两个参数，则默认为起始值和终止值（不含），步长默认为1
  - 如果只有一个参数，则为终止值（不含），起始值默认为0，步长默认为1

```
for i in range(0,5,1):
    print(i)
```

```
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
0
1
2
3
4

Process finished with exit code 0
```

-
- for循环的扩展：
  - ***列表推导式：列表中的元素，来自符合某种条件的集合中的迭代变量
    - l=[i for i in range(10) if i%3==0]
    - print(l)

```
l=[i for i in range(10) if i%3==0]
print(l)
```

```
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
[0, 3, 6, 9]

Process finished with exit code 0
```

```
# for循环的扩展:
    # ***列表推导式:列表中的元素，来自符合某种条件的集合中的迭代变量

l = [i for i in range(100) if i%3==0]


# 输出100以内的偶数/奇数，不能使用步长
# oushu = []
```

    - 极大提升代码的效率
  - 多变量迭代多集合：通过多个变量同时迭代多个集合
    - 格式 for i,j,k,... in zip(集合1，集合2，集合3，...):
      - zip():将多个集合统一打包处理
      - 执行次数取决于木桶原理

```
oushu=[]
jishu=[]
for i in range(10):
    if i%2==0:
        oushu.append(i)
    else:
        jishu.append(i)
print('偶数:',oushu)
print('奇数:',jishu)
for x,y in zip(oushu,jishu):
    print(x,y)
```

```
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
偶数: [0, 2, 4, 6, 8]
奇数: [1, 3, 5, 7, 9]
0 1
2 3
4 5
6 7
8 9

Process finished with exit code 0
```

```
for x,y in zip([1,2,3],['a','b','c','d']):
    print(x,y)
```

```
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
1 a
2 b
3 c

Process finished with exit code 0
```

- 字典查询元素,可以双变量for循环迭代items()

```
d={'name':'张三','age':18,'hobby':'钓鱼','info':{'a':1,'b':2,'c':[1,2,3,4,5]}}
for k,v in d.items():
    print(k,v)
```

```
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
name 张三
age 18
hobby 钓鱼
info {'a': 1, 'b': 2, 'c': [1, 2, 3, 4, 5]}

Process finished with exit code 0
```

- 多重for循环：for循环语句中的待循环语句组，也是一个for循环
  - ***外层循环迭代一次，内层循环执行一轮

```
for i in range(1,3):#i=1
    for j in range(1,5):#j=1,2,3,4
        print('%s:%s' %(i,j))
```

```
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
1:1
1:2
1:3
1:4
2:1
2:2
2:3
2:4

Process finished with exit code 0
```

- 练习
  - 输出1加到100的总和
  - sum=0
  - for i in range(101):
  - sum=i+sum
  - print(sum)

```
sum=0
for i in range(101):
    sum=i+sum
```

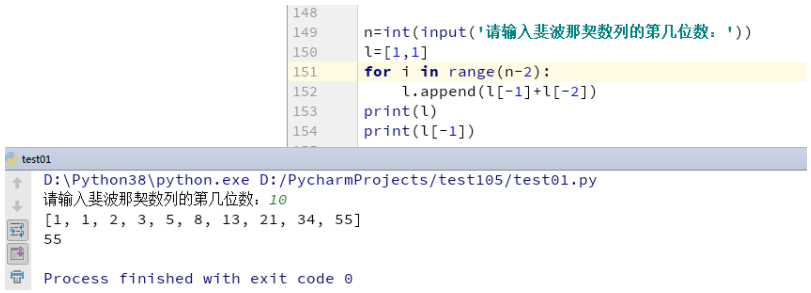
```
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
5050

Process finished with exit code 0
```

  - 不使用数据类型强制转换，输出100以内包含7的数。例如：7、17、71等
  - seven=[i for i in range(100) if i%10==7 or i//10==7]
  - print(seven)

```
seven=[i for i in range(100) if i%10==7 or i//10==7]
print(seven)
```

```
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
[7, 17, 27, 37, 47, 57, 67, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 87, 97]

Process finished with exit code 0
```

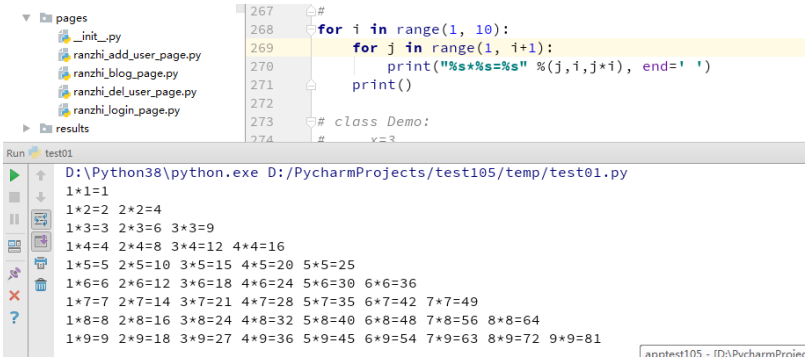
-
- 已知数列1,1,2,3,5,8...,观察规律，输出该数列指定任意位置的数
- n=int(input('请输入斐波那契数列的第几位数：'))
- l=[1,1]
- for i in range(n-2):
- l.append(l[-1]+l[-2])
- print(l)
- print(l[-1])



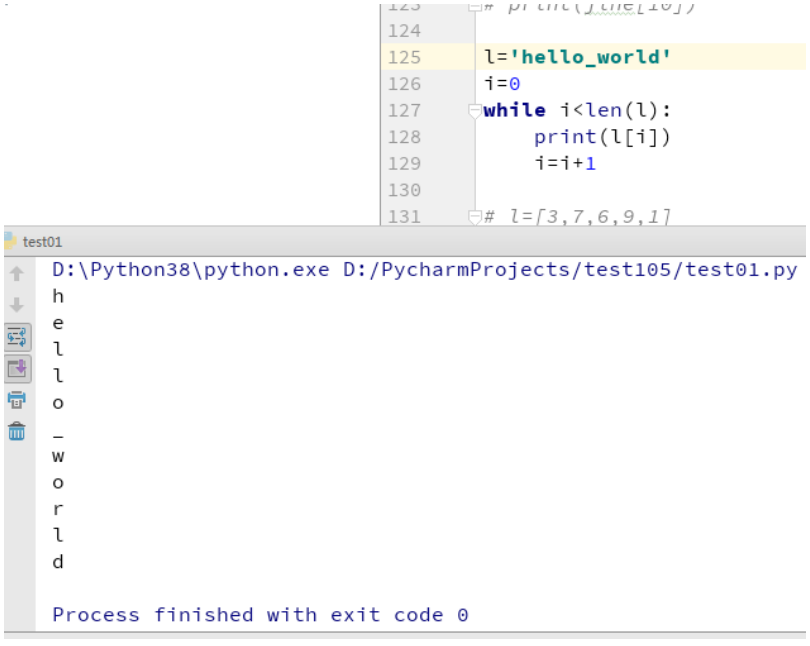
-
- 冒泡排序：一组数据，经过多轮多次相邻元素两两之间的比较，小的靠前，大的排后，最后得到一组有序数据
- 规律：
  - 1、比较的总轮次数为元素的个数减一
  - 2、随着轮次的递增，每轮比较的次数递减
  - l=[3,7,6,9,1]
  - for i in range(len(l)-1):#i:0,1,2,3
  - for j in range(len(l)-1-i):#5-1-0=4;5-1-1=3;5-1-2=2;5-1-3=1
  - if(l[j]>l[j+1]):
  - # temp=l[j]
  - # l[j]=l[j+1]
  - # l[j+1]=temp
  - l[j],l[j+1]=l[j+1],l[j]
  - print(l)

-
- 九九乘法表

- for i in range(1, 10):
- for j in range(1, i+1):
- print("%s*%s=%s" %(j,i,j*i), end=' ')
- print()

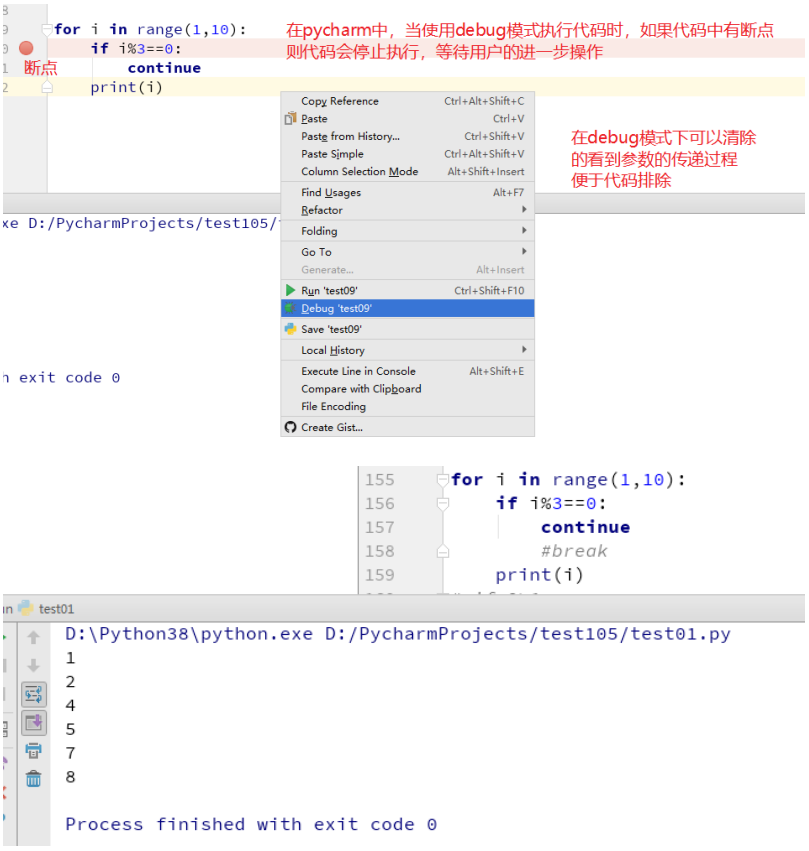


- while循环：基于条件成立的循环，条件成立循环开始，直到条件不再成立
  - 如果没有外力干涉或内部崩溃，while通常都是死循环
- while循环的格式
  - while 布尔表达式：
    - 待循环语句组



- while循环和for循环的区别
  - for循环有迭代变量和集合；while循环只有条件
  - *当范围确定通常使用for循环；当条件确定通常使用while循环
  - for循环通常可以改写成while循环

- continue语句：配合循环使用，当代码执行到continue语句时，立刻回到循环开始的地方，跳过本次循环，执行下一次循环
  - 通常在循环中充当例外的处理



- break语句：配合循环使用，当代码执行到break语句时，立刻摧毁循环，跳出循环之外
  - 通常在循环中发现特征数据

- pass语句：没有任何逻辑的语句，通常充当代码的补齐和占位

```
155  if 2>1:
156      print(1)
157  else:
158      pass
159
```

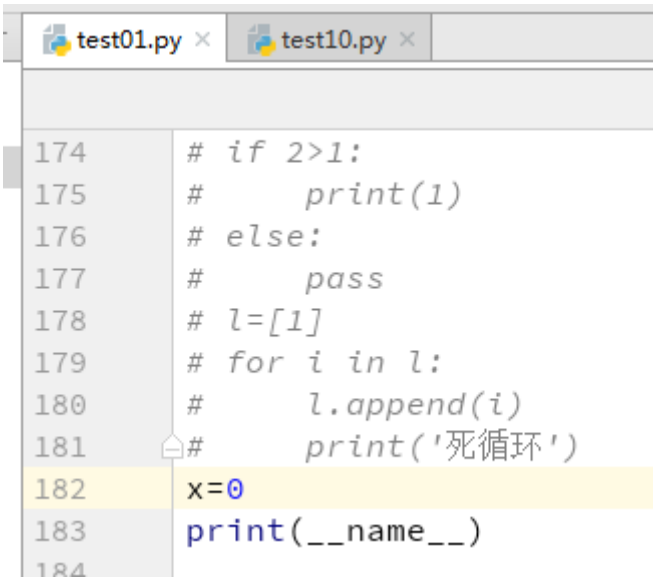
```
test01
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
1

Process finished with exit code 0
```

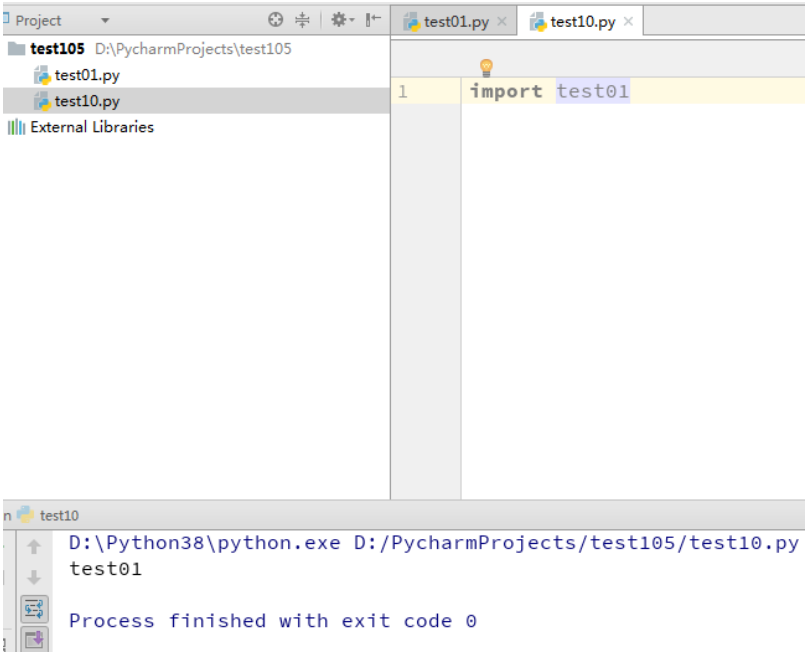
-
- 代码入口与代码调用
- 代码的调用：可以将已有的代码复用，面向对象编程的特色之一
- 代码调用的关键字：from...import
  - 标准库或同层文件下的调用
    - 标准库：python主程序自带的所有模块
    - 引用格式：import XXX
  - 非标准库的调用
    - 非标准库：第三方模块
    - from 库/模块/类 import 模块/类/函数/变量
  - 引用的快捷方式：将需要引用的类/函数/变量写在空白处等待报错，弹出红色小提示后选择需要引入的对象
    - ALT+回车键快速导入
-
- 代码入口：系统隐藏变量 _ _ name _ _
  - 在当前模块下_ _ name _ _的值为 _ _ main _ _

```
183  print(__name__)
```

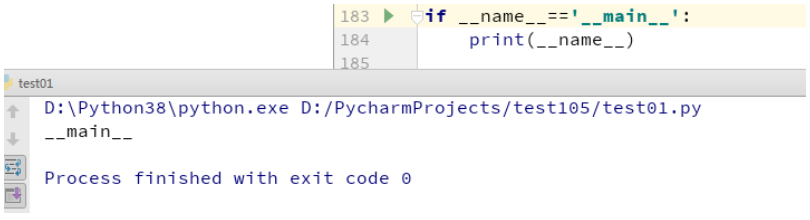
```
test01
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
__main__

Process finished with exit code 0
```

  - 被其他模块引用时，_ _ name _ _的值变为模块名称





- 可以根据_ _ name _ _的特性，进行条件判断
  - 保护当前模块下的调试代码被引用时，不会被执行



  - 提供一个代码执行的入口
-
- 作业:
- 输入一个正整数，判断该数是否为质数，如果是质数，输出该数是质数，如果不是质数，输出该数的因式，例如:输入7，输出7是质数；输入18，输出18不是质数，它的因式有[2,3,6,9]

```
num=10
l=[]
for i in range(2,num):
    if num%i==0:
        l.append(i)
if l:
    print('%s不是质数，它的因式分解有：%s' %(num,l))
else:
    print('%s是质数' %num)

# s='1234'
# x=0
# d={'0':0,'1':1,'2':2,'3':3,'4':4,'5':5,'6':6,'7'
```

```
test01
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
10不是质数，它的因式分解有：[2, 5]

Process finished with exit code 0
```

```
num=17
l=[]
for i in range(2,num):
    if num%i==0:
        l.append(i)
if l:
    print('%s不是质数，它的因式分解有：%s' %(num,l))
else:
    print('%s是质数' %num)

# s='1234'
# x=0
# d={'0':0,'1':1,'2':2,'3':3,'4':4,'5':5,'6':6,'7':
```

```
test01
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
17是质数

Process finished with exit code 0
```

- num=17
- l=[]
- for i in range(2,num):
- if num%i==0:
- l.append(i)
- if l:
- print('%s不是质数，它的因式分解有：%s' %(num,l))
- else:
- print('%s是质数' %num)
-
- 不使用任何数据类型强制转换，将类似于**'1234'**字符串，转换为整数型

```
s='1234'
x=0
d={'0':0,'1':1,'2':2,'3':3,'4':4,'5':5,'6':6,'7':7,'8':8,'9':9}
for i in range(len(s)):
    x=x+d[s[i]]*10**(len(s)-1-i)
print(x,type(x))
```

```
test01
D:\Python38\python.exe D:/PycharmProjects/test105/test01.py
1234 <class 'int'>

Process finished with exit code 0
```

- s='1234'
- x=0
- d={'0':0,'1':1,'2':2,'3':3,'4':4,'5':5,'6':6,'7':7,'8':8,'9':9}
- for i in range(len(s)):
- x=x+d[s[i]]*10**(len(s)-1-i)
- print(x,type(x))