

## 课程介绍:

- 1、字符串的驻留机制
- 2、字符串的常用操作
- 3、字符串的比较
- 4、字符串的切片操作
- 5、格式化字符串
- 6、字符串的编码转换

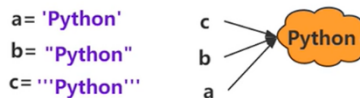
## 字符串的驻留机制

### • 字符串

- 在Python中字符串是基本数据类型，是一个不可变的字符序列

### • 什么叫字符串驻留机制呢？

- 仅保存一份相同且不可变字符串的方法，不同的值被存放在字符串的驻留池中，Python的驻留机制对相同的字符串只保留一份拷贝，后续创建相同字符串时，不会开辟新空间，而是把该字符串的地址赋给新创建的变量



## 字符串的驻留机制

### • 驻留机制的几种情况(交互模式)

- 字符串的长度为0或1时
- 符合标识符的字符串
- 字符串只在编译时进行驻留，而非运行时
- [-5,256]之间的整数数字

- sys中的intern方法强制2个字符串指向同一个对象
- PyCharm对字符串进行了优化处理

## 字符串的驻留机制

- 驻留机制的几种情况(交互模式)
  - 字符串的长度为0或1时
  - 符合标识符的字符串
  - 字符串只在编译时进行驻留，而非运行时
  - [-5,256]之间的整数数字
- sys中的intern方法强制2个字符串指向同一个对象
- PyCharm对字符串进行了优化处理



## 字符串的驻留机制

- 字符串驻留机制的优缺点
  - 当需要值相同的字符串时，可以直接从字符串池里拿来使用，避免频繁的创建和销毁，提升效率和节约内存，因此拼接字符串和修改字符串是会比较影响性能的。
  - 在需要进行字符串拼接时建议使用 str类型的join方法，而非+ ,因为join()方法是先计算出所有字符串中的长度，然后再拷贝，只new一次对象，效率要比"+"效率高



## 字符串的常用操作

- 字符串的查询操作的方法

功能	方法名称	作用
查询方法	index()	查找子串substr第一次出现的位置, 如果查找的子串不存在时, 则抛出ValueError
	rindex()	查找子串substr最后一次出现的位置, 如果查找的子串不存在时, 则抛出ValueError
	find()	查找子串substr第一次出现的位置, 如果查找的子串不存在时, 则返回-1
	rfind()	查找子串substr最后一次出现的位置, 如果查找的子串不存在时, 则返回-1



## 字符串的常用操作

### • 字符串的大小写转换操作的方法

功能	方法名称	作用
大小写转换	<code>upper()</code>	把字符串中所有字符都转成大写字母
	<code>lower()</code>	把字符串中所有字符都转成小写字母
	<code>swapcase()</code>	把字符串中所有大写字母转成小写字母, 把所有小写字母都转成大写字母
	<code>capitalize()</code>	把第一个字符转换为大写, 把其余字符转换为小写
	<code>title()</code>	把每个单词的第一个字符转换为大写, 把每个单词的剩余字符转换为小写

## 字符串的常用操作

### • 字符串内容对齐操作的方法

功能	方法名称	作用
字符串对齐	<code>center()</code>	居中对齐, 第1个参数指定宽度, 第2个参数指定填充符, 第2个参数是可选的, 默认是空格, 如果设置宽度小于实际宽度则返回原字符串
	<code>ljust()</code>	左对齐, 第1个参数指定宽度, 第2个参数指定填充符, 第2个参数是可选的, 默认是空格, 如果设置宽度小于实际宽度则返回原字符串
	<code>rjust()</code>	右对齐, 第1个参数指定宽度, 第2个参数指定填充符, 第2个参数是可选的, 默认是空格, 如果设置宽度小于实际宽度则返回原字符串
	<code>zfill()</code>	右对齐, 左边用0填充, 该方法只接收一个参数, 用于指定字符串的宽度, 如果指定的宽度小于等于字符串的长度, 返回字符串本身

## 字符串的常用操作

### • 字符串劈分操作的方法

功能	方法名称	作用
字符串的劈分	<code>split()</code>	从字符串的左边开始劈分, 默认的劈分字符是空格字符串, 返回的值都是一个列表
		以通过参数 <code>sep</code> 指定劈分字符串时的劈分符
		通过参数 <code>maxsplit</code> 指定劈分字符串时的最大劈分次数, 在经过最大次劈分之后, 剩余的字符串会单独做为一部分
	<code>rsplit()</code>	从字符串的右边开始劈分, 默认的劈分字符是空格字符串, 返回的值都是一个列表
		以通过参数 <code>sep</code> 指定劈分字符串时的劈分符
		通过参数 <code>maxsplit</code> 指定劈分字符串时的最大劈分次数, 在经过最大次劈分之后, 剩余的字符串会单独做为一部分

# 字符串的常用操作

## • 判断字符串操作的方法

功能	方法名称	作用
判断字符串的方法	isidentifier()	判断指定的字符串是不是合法的标识符
	isspace()	判断指定的字符串是否全部由空白字符组成(回车、换行, 水平制表符)
	isalpha()	判断指定的字符串是否全部由字母组成
	isdecimal()	判断指定字符串是否全部由十进制的数字组成
	isnumeric()	判断指定的字符串是否全部由数字组成
	isalnum()	判断指定字符串是否全部由字母和数字组成



# 字符串的常用操作

## • 字符串操作的其它方法

功能	方法名称	作用
字符串替换	replace()	第1个参数指定被替换的子串, 第2个参数指定替换子串的字符串, 该方法返回替换后得到的字符串, 替换前的字符串不发生变化, 调用该方法时可以通过第3个参数指定最大替换次数
字符串的合并	join()	将列表或元组中的字符串合并成一个字符串



# 字符串的比较操作

## • 字符串的比较操作

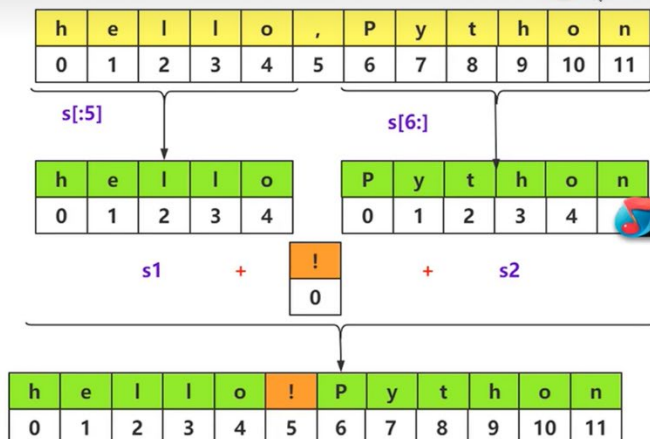
- 运算符: >, >=, <, <=, ==, !=
- 比较规则: 首先比较两个字符串中的第一个字符, 如果相等则继续比较下一个字符, 依次比较下去, 直到两个字符串中的字符不相等时, 其比较结果就是两个字符串的比较结果, 两个字符串中的所有后续字符将不再被比较
- 比较原理: 两上字符进行比较时, 比较的是其ordinal value(原始值), 调用内置函数ord可以得到指定字符的ordinal value。与内置函数ord对应的是内置函数chr, 调用内置函数chr时指定ordinal value可以得到其对应的字符



## 字符串的切片操作

- 字符串是不可变类型

- 不具备增、删、改等操作
- 切片操作将产生新的对象



马士兵教育

## 格式化字符串

- 为什么需要格式化字符串

证明。

兹证明 XXX，身份证号：XXX，现居家庭住址：XXX，为我公司在职员工，工作地址为：XXX，单位负责人：XXX，座机联系电话：XXX，因工作需要，每天需出入该小区，本单元每天早晚有定时消毒、测量体温、出入登记等防控疫情措施。

特此证明。

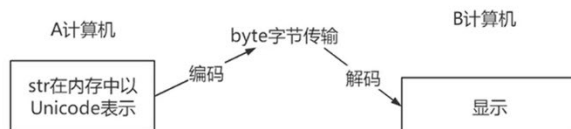
公司名称：XXX

XXX 年 XX 月 XX 日

马士兵教育

## 字符串的编码转换

- 为什么需要字符串的编码转换



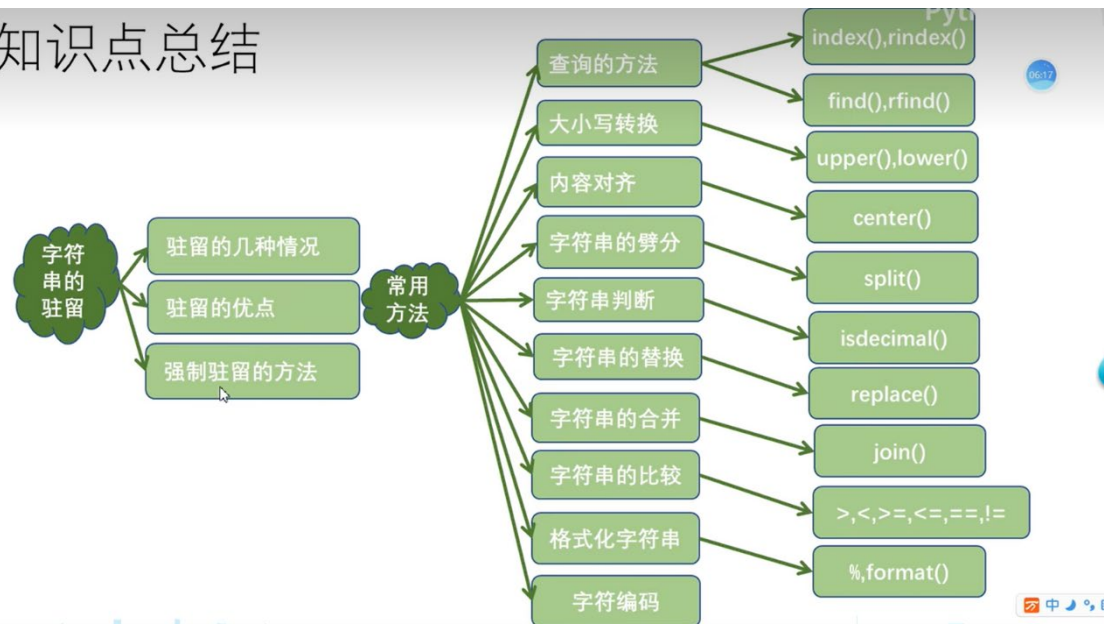
- 编码与解码的方式

- 编码：将字符串转换为二进制数据(bytes)
- 解码：将bytes类型的数据转换成字符串类型

马士兵教育



## 知识点总结



## 课程介绍:

- 1、函数的创建和调用
- 2、函数的参数传递
- 3、函数的返回值
- 4、函数的参数定义
- 5、变量的作用域
- 6、递归函数

## 函数的创建和调用

- 什么是函数
  - 函数就是执行特定任务和以完成特定功能的一段代码
- 为什么需要函数
  - 复用代码
  - 隐藏实现细节
  - 提高可维护性
  - 提高可读性便于调试
- 函数的创建

```
def 函数名 ([输入参数]) :  
    函数体  
    [return xxx]
```



# 函数的参数传递

## • 函数调用的参数传递

- 位置实参
  - 根据形参对应的位置进行实参传递
- 关键字实参
  - 根据形参名称进行实参传递

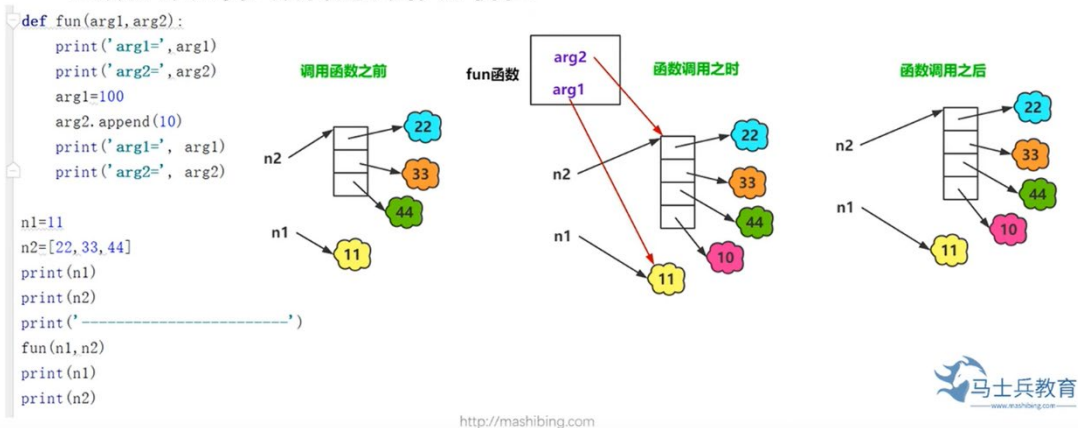
calc( 10, 20 )  
def calc( a , b ):

calc( b=10 , a=20 )  
def calc( a , b ):



# 函数的参数传递

## • 函数调用的参数传递内存分析图



<http://mashibing.com>



# 函数的返回值

## • 函数返回多个值时，结果为元组

```
def fun(num):  
    odd=[] #存奇数  
    even=[] #存偶数  
    for i in num:  
        if i%2:  
            odd.append(i)  
        else:  
            even.append(i)  
    return odd,even  
  
print(fun([10,29,34,23,44,53,55]))
```

```
Run: demo3 x  
"C:\Program Files\Python38\python.exe"  
E:/vippython/chap10/demo3.py  
([29, 23, 53, 55], [10, 34, 44])
```

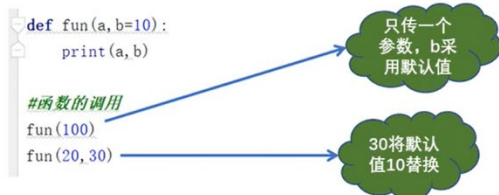


<http://mashibing.com>

## 函数的参数定义

### • 函数定义默认值参数

- 函数定义时，给形参设置默认值，只有与默认值不符的时候才需要传递实参



## 函数的参数定义

### • 个数可变的位置参数

- 定义函数时，可能无法事先确定传递的位置实参的个数时，使用可变的位置参数
- 使用\*定义个数可变的位置形参
- 结果为一个元组

```
def fun(*args):  
    print(args)  
fun(10)  
fun(10, 20, 30)
```

```
Files\python30\python.exe  
E:/dream/chapfile/d.py  
(10,)  
(10, 20, 30)
```

### • 个数可变的键字形参

- 定义函数时，无法事先确定传递的键字实参的个数时，使用可变的键字形参
- 使用\*\*定义个数可变的键字形参
- 结果为一个字典

```
def fun(**args):  
    print(args)  
fun(a=10)  
fun(a=10, b=20, c=30)
```

```
E:/dream/chapfile/d.py  
{ 'a': 10 }  
{ 'a': 10, 'b': 20, 'c': 30 }
```

## 函数的参数总结

序号	参数的类型	函数的定义	函数的调用	备注
1	位置实参		√	
	将序列中的每个元素都转换为位置实参		√	使用*
2	关键字实参		√	
	将字典中的每个键值对都转换为关键字实参		√	使用**
3	默认值形参	√		
4	键字形参	√		使用*
5	个数可变的位置形参	√		使用*
6	个数可变的键字形参	√		使用 **



# 变量的作用域

00:06

- 变量的作用域

- 程序代码能访问该变量的区域

- 根据变量的有效范围可分为

- 局部变量

- 在函数内定义并使用的变量，只在函数内部有效，局部变量使用global声明，这个变量就会就成全局变量

- 全局变量

- 函数体外定义的变量，可作用于函数内外



# 递归函数

00:07

- 什么是递归函数

- 如果在一个函数的函数体内调用了该函数本身，这个函数就称为递归函数

- 递归的组成部分

- 递归调用与递归终止条件

- 递归的调用过程

- 每递归调用一次函数，都会在栈内存分配一个栈帧，
  - 每执行完一次函数，都会释放相应的空间

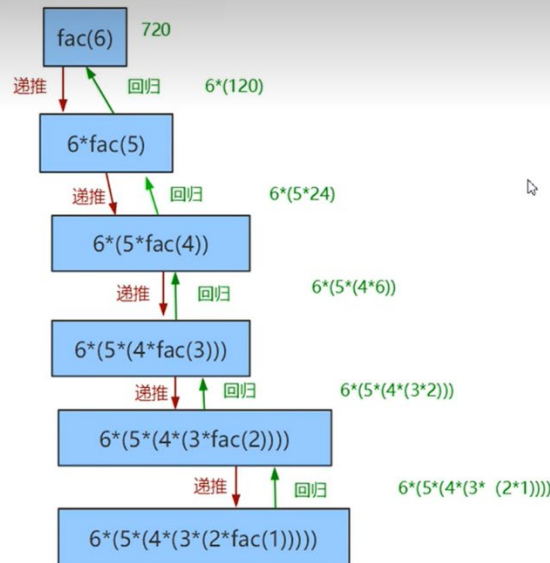
- 递归的优缺点

- 缺点：占用内存多，效率低下
  - 优点：思路 and 代码简单



## 递归函数

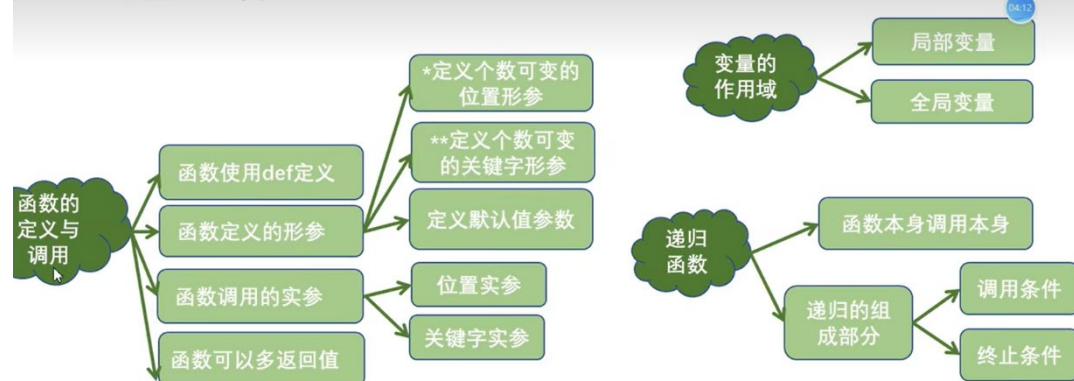
- 使用递归来计算阶乘



<http://mashibing.com>

马士兵教育  
[www.mashibing.com](http://www.mashibing.com)

## 知识点总结



## 课程介绍:

- 1、Bug的由来及分类
- 2、不同异常类型的处理方式
- 3、异常处理机制
- 4、PyCharm的调试模式

马士兵教育  
[www.mashibing.com](http://www.mashibing.com)

# Bug的由来及分类

00:44

## • Bug的由来

- 世界上第一部万用计算机的进化版-马克2号(Mark II)

## • Debug



# Bug的由来及分类

00:05

## • Bug的常见类型

- 粗心导致的语法错误 SyntaxError
- (1)

```
age=input('请输入你的年龄:')
if age>=18
    print('成年人，做事需要负法律责任了')
```

- (2)

```
while i<10:
    print(i)
```

# Bug的由来及分类

00:47

## • Bug的常见类型

- 粗心导致的语法错误 SyntaxError
- (3)

```
for i in range(3):
    uname=input('请输入用户名:')
    pwd=input('请输入密码:')
    if uname='admin' and pwd='admin':
        print('登录成功! ')
        break
    else
        print('输入有误 ')
else
    print('对不起，三次均输入错误')
```

# Bug的由来及分类

## • Bug的常见类型

### • 粗心导致错误的自查宝典

- 1.漏了**末尾的冒号**，如if语句,循环语句,else子句等
- 2.**缩进错误**，该缩进的没缩进，不该缩进的瞎缩进
- 3.把**英文符号**写成中文符号，比如说：引号，冒号，括号
- 4.字符串拼接的时候，把**字符串和数字**拼在一起
- 5.没有**定义变量**，比如说while的循环条件的变量
- 6.“==”比较运算符和“=”赋值运算符的混用

# Bug的由来及分类

00:07

## • Bug的常见类型

- 知识不熟练导致的错误
- (1) 索引越界问题IndexError

```
lst=[11,22,33,44]
print(lst[4])
```

- (2) append()方法的使用掌握不熟练

```
lst=[]
lst.append('A','B','C')
print(lst)
```



# Bug的由来及分类

## • Bug的常见类型

- 思路不清导致的问题解决方案

- (1) 使用print()函数
- (2) 使用“#”暂时注释部分代码

### • 题目要求

- 豆瓣电影Top250排行，使用列表存储电影信息，要求输入名字在屏幕上显示xxx出演了哪部电影。

```
lst=[{'rating':[9.7,50], 'id':'1292052', 'type':['犯罪','剧情'], 'title':'肖申克的救赎',
      'actors':['蒂姆·罗宾斯','摩根·弗里曼']},
      {'rating':[9.6,50], 'id':'1291546', 'type':['剧情','爱情','同性'], 'title':'霸王别姬',
      'actors':['张国荣','张丰毅','巩俐','葛优']},
      {'rating':[9.6,50], 'id':'1296141', 'type':['剧情','犯罪','悬疑'], 'title':'控方证人',
      'actors':['泰隆·鲍华','玛琳·黛德丽']}]
```

# Bug的由来及分类

## • Bug的常见类型

- 思路不清导致的问题

```
name=input('请输入你要查询的演员:')
for item in lst:
    for movie in item:
        actors=movie['actors']
        if name in actors:
            print(name+'出演了:'+movie)
```

```
E:/dream/chap11/demol.py
请输入你要查询的演员:张国荣
Traceback (most recent call last):
  File "E:/dream/chap11/demol.py", line 15, in <module>
    actors=movie['actors']
TypeError: string indices must be integers
```

- 使用print()输出item以及moive的值

```
"C:\Program Files\Python38\python.exe" E:/dream/chap11/demol.py
请输入你要查询的演员:张国强
Traceback (most recent call last):
  File "E:/dream/chap11/demol.py", line 17, in <module>
    rating
    actors=movie['actors']
TypeError: string indices must be integers
```



# Bug的由来及分类

## • Bug的常见类型

- 被动掉坑:程序代码逻辑没有错,只是因为用户错误操作或者一些“例外情况”而导致的程序崩溃
- 题目要求:输入两个整数并进行除法运算

```
E:/dream/chap11/demol.py
请输入一个整数:10
请输入另一个整数:5
结果为: 2.0

E:/dream/chap11/demol.py
请输入一个整数:10
请输入另一个整数:0
Traceback (most recent call last):
  File "E:/dream/chap11/demol.py", line 7, in <module>
    result=n1/n2
ZeroDivisionError: division by zero
```

```
请输入一个整数:a
Traceback (most recent call last):
  File "E:/dream/chap11/demol.py", line 5, in <module>
    n1=int(input('请输入一个整数:'))
ValueError: invalid literal for int() with base 10: 'a'
```

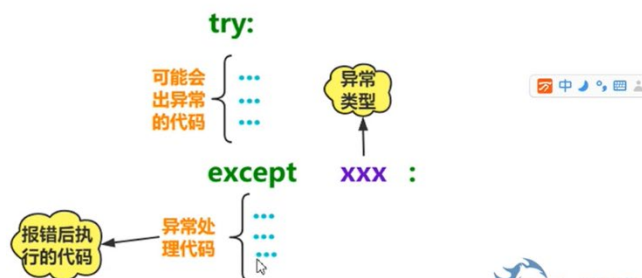


# Python的异常处理机制

## • Bug的常见类型

- 被动掉坑问题的解决方案
- Python提供了异常处理机制,可以在异常出现时即时捕获,然后内部“消化”,让程序继续运行

```
try:
    n1=int(input('请输入一个整数:'))
    n2=int(input('请输入另一个整数:'))
    result=n1/n2
    print('结果为:',result)
except ZeroDivisionError:
    print('除数不能为0哦!!!')
```





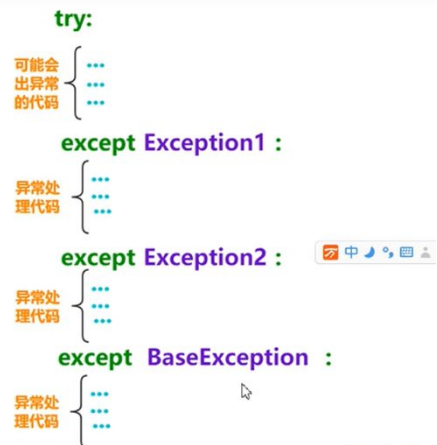
# Python的异常处理机制

## • 多个except结构

- 捕获异常的顺序按照先子类后父亲类的顺序，为了避免遗漏可能出现的异常，可以在最后增加BaseException

```
try:
    n1=int(input('请输入一个整数:'))
    n2=int(input('请输入另一个整数:'))
    result=n1/n2
    print('结果为:',result)
except ZeroDivisionError:
    print('除数不能为0哦!!!')
except ValueError:
    print('不能将字符串转换为数字')
except BaseException as e:
    print(e)
```

<http://mashibing.com>



马士兵教育  
www.mashibing.com

# Python的异常处理机制

## • try...except...else结构

- 如果try块中没有抛出异常，则执行else块，如果try中抛出异常，则执行except块

```
try:
    n1=int(input('请输入一个整数:'))
    n2=int(input('请输入另一个整数:'))
    result=n1/n2
except BaseException as e:
    print('出错了')
    print(e)
else:
    print('结果为:',result)
```

马士兵教育  
www.mashibing.com

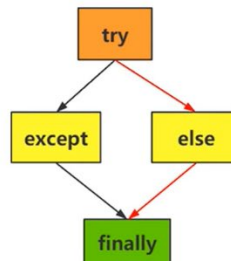
马士兵教育  
www.mashibing.com

# Python的异常处理机制

## • try...except...else...finally结构

- finally块无论是否发生异常都会被执行，能常用来释放try块中申请的资源

```
try:
    n1=int(input('请输入一个整数:'))
    n2=int(input('请输入另一个整数:'))
    result=n1/n2
except BaseException as e:
    print('出错了')
    print(e)
else:
    print('结果为:',result)
finally:
    print('无论是否产生异常，总会被执行的代码')
print('程序结束')
```



马士兵教育  
www.mashibing.com

马士兵教育  
www.mashibing.com

# Python中常见的异常类型

- Python常见的异常类型

序号	异常类型	描述
1	ZeroDivisionError	除(或取模)零 (所有数据类型)
2	IndexError	序列中没有此索引(index)
3	KeyError	映射中没有这个键
4	NameError	未声明/初始化对象 (没有属性)
5	SyntaxError	Python 语法错误
6	ValueError	传入无效的参数

## Python的异常处理机制

- **traceback**模块

- 使用traceback模块打印异常信息

```
import traceback
try:
    print('1. -----')
    num=10/0
except:
    traceback.print_exc()
```

```
1. -----
Traceback (most recent call last):
  File "E:/dream/chap11/demol.py", line 7, in <module>
    num=10/0
ZeroDivisionError: division by zero
```

# PyCharm开发环境的调试


## • 断点

- 程序运行到此处，暂时挂起，停止执行。此时可以详细观察程序的运行情况，方便做出进一步的判断

```
4  
5 num=int(input('请输入年龄:'))  
6 print(num)
```

## • 进入调试视图

- 进入调试视图的三种方式

- (1)单击工具栏上的按钮 
- (2)右键单击编辑区：点击：debug'模块名'
- (3)快捷键:shift+F9

马士兵教育

马士兵教育  
www.mashibing.com

# 知识点总结

