程序有两种形式：编译型，解释型 程序运行方式：交互式，文件式

标志符：字母、\_、数字 (区分大小写) 关键字（保留字）

int：二进制 0B,0b 八进制 0O,0o 十六进制 0x,0X Float

运算符：+ - \* / % \*\* // 位运算符： ＆ | ^ ～

转义字符：\: \t \n \r

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Str | “ ” | 有序数据类型——序列  （存在先后关系）  Len()  Min()  Max()  s.index(x,i,j)  s.count(x) | 不可变 | len(str)、max(str)、min(str)  upper(), lower()  islower(),isupper(),isdigit,isalpha  find(),count(str)  startswith(), endswith()  str.startswith(‘1’)/endswith(‘1’)  str.split(seq,num)  ‘seq’.join(str)  Input(), print(),eval()  print(a, sep=’’, end=’ ’*(不分行)*)  Sort reverse=true 降序 false升序  Type()变量类型  Id()变量id值 |
| List | [ , ] | 可变 | ls.appe  nd(x),ls.pop(i)  ls.remove(x),ls.reverse(x),ls.sort()  del ls[i:j] |
| Tuple  元组 | ( , )  只有一个元素时逗号不可省 | 不可变 | len(),max(),min()  列表和元组相互转换的函数是tuple(lst)和list(tup)  **t=(‘string’)字符串与t=()空元组的区别** |
| Dict  字典 | { : ，：}  内置的映射类型 | 无序类型——不可索引 | 可变 | 由key-value的键值对组成.  dicts字典名，key键，value值  in,dicts[‘key’]查找值  dicts[key]=value添加与修改  dicts.keys()返回所有键  dicts.values()返回所有值,  dicts.items()返回所有键值对,  dicts.get(key, default)找键 |
| Set  集合 | { , }  没有重复元素 | **{}空字典与set()空集合的区别**  S.add(x), S.clear(), S.remove(), len(S)  S & T， S | T， S-T， S^T， S<=T， S>=T  **词频统计，排序输出** |

|  |  |
| --- | --- |
| str.format() | 位置参数、关键字参数 |
| % | %10s |
| f-string |  |
| 格式控制模式 | (‘{:<8},{:>8},{:\*^8}’.format()) |

python具有**函数式编程**的特性

lambda函数：lambda params : expression

eg. F= lambda x,y : x\*x + y\*y ，F(10,20)=10\*10+20\*20=500

闭包：概念、例子，装饰器应用

内部函数引用了外部函数作用域的变量，这个内部函数被称为闭包。

创建条件：闭包函数必须有嵌套函数;嵌套函数需要引用外部函数中的变量;外部函数需要将嵌套函数名作为返回值返回。

递归：函数在其定义或声明中直接或间接调用自身的一种方法。

斐波那契数列、汉诺塔等问题

变量作用域：nolocal，global，全局变量的访问与修改

局部变量；全局变量

g\_var = 20 #全局变量

def fun(item):

**global** g\_var #声明全局变量g\_var

g\_var = 30 #全局变量的修改

item = g\_var

return item

print(fun(10)) #30

内层函数访问外层函数的局部变量：nolocal

def outer():

x=100

global y

y=200

def inner():

nonlocal x

x=1000

y=2000

print('inner函数中的x,y值为：',x,y) #inner函数中的x,y值为： 1000 2000

inner()

print('outer函数中的x,y值为：',x,y) #outer函数中的x,y值为： 1000 200

x=10

y=20

outer()

print("执行outer,inner函数之后x,y值为：",x,y)

#执行outer,inner函数之后x,y值为： 10 200

|  |  |
| --- | --- |
| 内置函数：不用import引入 | |
| 数学函数 | abs(), divmod(),max(),min(),pow(),round()返回浮点数的四舍五入值,sum() |
| 转换函数 | int(),float(),str(),bin()转换成二进制字符串  hex()转换成十六进制字符串  str(),ord(),chr() |
| 序列函数 | all(),any(),range(),map(),filter(),reduce(),zip(),  sorted()默认false升序true降序,  reversed(),eval() |

|  |  |
| --- | --- |
| 创建类：封装性、继承性、多态性 | Class Name():  类的属性（成员变量）  类的方法（成员方法） |
| 创建对象 | 对象名 = 类名()  对象名.属性名 = 值 |
| 构造方法：初始化对象 | \_\_init\_\_() |
| 析构方法：释放对象所占资源 | \_\_del\_\_() |
| self参数 | 成员方法的第1个参数是 self，表示是对象自身，通过self可以访问成员属性 |
| 类定义 | class class\_name():  xxxx属性  def \_\_init\_\_(self, param1, param2,....):  self.param1= param1  self.param2=param2  def func(self,param3):  self.param2.....  param3.....  return xxxx |
| 成员变量：在构造方法\_\_init\_\_()中定义的  定义时以self作为第1个参数。  成员变量属于实例（对象），只能通过对象名访问。 | 类变量：在类中方法之外定义的。  类变量属于类，可以通过类名访问，也可以通过对象名访问，被类的所有对象共享。 |
| 类方法：既可以通过对象名调用类方法，又可以通过类名调用类方法  class 类名:  @classmethod  def 类方法名(cls):  方法体  #通过cls参数可以访问类的属性 | 静态方法：起到类似于函数的作用。  class 类名:  @staticmethod  def 静态方法名():  方法体  没有self参数，所以无法访问类的实例属性；  没有cls参数，所以无法访问类属性。 |

|  |  |
| --- | --- |
| 类的继承  类的继承是指在一个现有类的基础上去构建一个新的类，构建出来的新类称作子类，被继承的类称作父类，子类会自动拥有父类所有可继承的属性和方法。 | class 子类名(父类名):  类的属性  类的方法  以两个下划线“\_\_”开头的属性是**私有属性**，只在类的内部访问。  以两个下划线“\_\_”开头的方法是**私有方法**，只在类的内部访问。 |
| 方法重写  如果父类的方法不能满足子类的需求，子类可以按照自己的方式重新实现从父类中继承的方法，这就是方法的重写。 | 子类中重写的方法要和父类被重写的方法具有相同的方法名和参数列表。  super()方法用于在继承过程中访问父类的成员。 |
| 多继承  在子类名称后的括号中标注出要继承的多个父类，并且多个父类之间使用逗号分隔。 | class 子类(父类1, 父类1):  属性描述  方法描述 |
| 类的多态  在同一个方法中，这种由于参数不同而导致执行效果各异的现象就是多态。 | Python的多态并不考虑对象的类型，而是关注对象具有的行为，根据被引用子类对象特征的不同，得到不同的运行结果。  Python的多态并不要求继承的存在。但继承关系的存在，对多态起到约束作用，可以使程序更为健壮。 |
| 运算符重载 | 创建类、方法、属性，结合异常处理 |

|  |  |
| --- | --- |
| 模块  提高代码的可维护性、可重用性。  有利于避免函数名和变量名冲突。  模块是实现的某一特定功能的函数和类的文件。 | import modulename [as alias]  from modulename import fun1,fun2,… |
| \_\_name\_\_属性 | 如果.py文件作为模块被调用，  \_\_name\_\_的属性值为模块文件的主名，  如果模块独立运行，  \_\_name\_\_属性值为\_\_main\_\_。  语句if \_\_name\_\_ == 'main' 的作用就是控制这两种不同情况执行代码的过程。 |
| 包：模块文件所在的目录  包的外层目录必须在Python的搜索路径中。包的下级子目录中，每个目录需要包含一个\_\_init\_\_.py文件， | \_\_init\_\_.py  \_\_init\_\_.py文件可以为空，也可以在其中定义\_\_all\_\_列表指定包中可以导入的模块。 |
| 标准库（内置库） | Random ， math，  datetime, |
| 第三方库 | pip install  turtle包、jieba包词频统计  Numpy, matplotlib, requests, pandas |

|  |  |
| --- | --- |
| 文件操作 | f = open(“filename.txt”, “w+”):  f.writelines(s)  f.read(size)  f.seek(0)  f.tell() |
| 文件打开与关闭 | myfile = open(filename[,mode])  mode取值为："r"，"r+"，"w"，"w+"，"a"，"rb"，"wb"等  myfile.close()  myfile.flush() |
| 文件读写 | seek(), tell(),移动文件指针  read(size), readline(size),readlines(hint),  write(str), writelines(seq\_of\_str)  定位读写  seek(n)：定位到文件的n位置处  tell()：返回当前的文件指针位置 |
| 文件目录操作：os包 | os, os.path  对文件夹进行各种操作的方法，都需要引入os模块（包） |
| CSV文件读写 | import csv  csv.reader()和csv.writer() |

|  |  |
| --- | --- |
| 异常类  Exception | NameError，ZeroDivisionError, IndexError, KeyError, AttributeError, SyntaxError（语法错误） FileNotFoundError, IOError |
| 异常处理 | try:  语句块 #抛出异常  except exceptionName1: #捕获异常  异常处理代码1  except exceptionName2:  异常处理代码2  else:  无异常发生时的语句块  finally:  必须处理的语句 |
| 抛出异常：raise  raise主动抛出所需要类型的出错信息，显示地抛出异常 | raise异常类 #抛出异常，并隐式地创建类对象  raise异常类对象 #抛出异常，创建异常类的实例对象  raise #重新抛出刚刚发生的异常 |
| 自定义异常 | 例子：定义一个Circle类，其中有求面积的方法，当半径小于0时，抛出一个用户自定义异常。 |

|  |  |
| --- | --- |
| from tkinter import \* #引入包  Win = TK() #窗口  Win.title(“title”)  Win.geometry(“600x400”)  Win.config()  Win.mainloop() | title() #设置标题  geometry() #设置窗口大小 “宽\*高” |
| 布局管理 | Pack(), grid(), place(),  Frame容器 |
| 常用组件 | config() #设置字体等属性  Label #创建标签  Button---command属性 #创建按钮  Entry #输入组件  Listbox #创建列表框  Radiobutton  Checkbutton  Spinbox |
| 事件处理 | button  command参数、bind()方法  btn1= Button(win,text="test",command=fun) |
| 消息窗口 | messagebox  tkinter.messagebox.showinfo()  from tkinter import \*  def fun():  messagebox.showinfo(title="mbox",message="Hello!")    win=Tk()  win.title("配置组件属性") #title()方法  win.geometry("300x200") #geometry()方法  win.mainloop() |