2017-9-26 10:38:19

1. 《PYTHON QT GUI快速编程 PYQT编程指南》
2. 科学运算、人工智能: 典型库NumPy, SciPy, Matplotlib, Enthought librarys,pandas
3. 金融：量化交易，金融分析，在金融工程领域，Python不但在用，且用的最多，而且重要性逐年提高。原因：作为动态语言的Python，语言结构清晰简单，库丰富，成熟稳定，科学计算和统计分析都很牛，生产效率远远高于c,c++,java,尤其擅长策略回测
4. 通常我们所说的动态语言、静态语言是指动态类型语言和静态类型语言。

（1）动态类型语言：动态类型语言是指在运行期间才去做数据类型检查的语言，也就是说，在用动态类型的语言编程时，永远也不用给任何变量指定数据类型，该语言会在你第一次赋值给变量时，在内部将数据类型记录下来。Python和Ruby就是一种典型的动态类型语言。

（2）静态类型语言：静态类型语言与动态类型语言刚好相反，它的数据类型是在编译期间检查的，也就是说在写程序时要声明所有变量的数据类型，C/C++是静态类型语言的典型代表，其他的静态类型语言还有C#、JAVA等。

5. 强类型定义语言和弱类型定义语言

（1）强类型定义语言：强制数据类型定义的语言。也就是说，一旦一个变量被指定了某个数据类型，如果不经过强制转换，那么它就永远是这个数据类型了。强类型定义语言是类型安全的语言。

（2）弱类型定义语言：数据类型可以被忽略的语言。它与强类型定义语言相反, 一个变量可以赋不同数据类型的值。

Python是动态语言，是强类型定义语言（类型安全的语言）; VBScript是动态语言，是弱类型定义语言（类型不安全的语言）; JAVA是静态语言，是强类型定义语言（类型安全的语言）。

6. utf-8是可变长度的，存英文用ASCII，存中文需要3个字节。

Unicode符号范围 | UTF-8编码方式

(十六进制) | （二进制）

------------------------------+---------------------------------------------

0000 0000-0000 007F | 0xxxxxxx

0000 0080-0000 07FF | 110xxxxx 10xxxxxx

0000 0800-0000 FFFF | 1110xxxx 10xxxxxx 10xxxxxx

0001 0000-0010 FFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

7. 当行注视：# 被注释内容

多行注释：三个双引号或单引号，""" 被注释内容 """ 或’’’ 被注释内容 ’’’

8. 三个双引号或单引号可以作为多行字符串赋值，单行赋值用一个单引号或双引号。

9. 幂运算“\*\*”， 取整除“//”， 取模“%”，

10. Python3里input输入的默认为字符串，作数字需要类型转换int()。

11. if 条件1:

print("1")

elif 条件2:

print("2")

else:

12. while true:

Break#contine

else:

13.for i in range(0,10,1):

Break#contine

else:

14.列表list，元组（不可变列表）tuple，字典（无序）dict

15. 三元运算:result = 值1 if 条件 else 值2

16. os.system()只执行命令不保存结果，可以用os.popen().read()

17.创建目录os.mkdir()

18.

>>>'€20'.encode('utf-8')

b'\xe2\x82\xac20'

>>> b'\xe2\x82\xac20'.decode('utf-8')

'€20'

19. 列表list

切片

print(name[起始位置:结束位置+1]) #从起始位置取到结束位置

print(name[:结束位置+1]) #从第一个取到结束位置，等同print(name[0:结束位置+1])

print(name[-3:-1]) #(-1=-2+1)取到最后第二个

print(name[-3:]) #取到最后一个  
print(name[: :step]) #step是步长

追加 name.append(成员)

插入names.insert(位置, 成员)

修改names[位置]=”xx”

删除成员name.remove(成员)

del name[位置]#等同于name.pop(位置)

name.pop() #没有下标默认删除最后一个

删除列表del name

查找name.index(成员)

统计name.count(成员)

清空列表name.clear()

反转列表name.reverse()

排序name.sort()#按ASCII顺序排序，不同类型不能排序会出错

扩展name.extend()  
浅复制name1=name.copy()#复制一个新的列表指针赋给name1，但是只复制第一层，第一层成员如果是指针，指针指向的第二层不会复制；name1=name二个指针指向同一个列表

其它浅复制name1=copy.copy(name)，name1=name[:]，name1=list(name)

深复制需要引入copy库。

Import copy

name=[…]

name1=copy.deepcopy(name)

列表循环

for i in name:

print(i)

20.元祖tuple不能修改，又叫只读列表

查找name.index(成员)

统计name.count(成员)

21. for i in enumerate(表):#输出每个下标和对应的成员

22. 判断字符串是否为数字str.isdigit()

23. 打印颜色字体：

格式：printf("\033[显示方式;前景色;背景色m字符串 \033[0m")

例子：printf("\033[1m\033[45;33m HELLO\_WORLD \033[0m\n");

颜色代码: QUOTE:

前景色 背景色 颜色

---------------------------------------

30 40 黑色

31 41 红色

32 42 绿色

33 43 黃色

34 44 蓝色

35 45 紫红色

36 46 青蓝色

37 47 白色

显示方式 意义

---------------------------------------

0 终端默认设置

1 高亮显示

4 使用下划线

5 闪烁

7 反白显示

8 不可见

30~37 设置前景色

40~47 设置背景色

nA 光标上移n行

nB 光标下移n行

nC 光标右移n行

nD 光标左移n行

24. 退出程序exit()【’ekzit】

25. 字符串操作

name="my name is alex"

name1="My name is {name2} and {year} old."

print(name.capitalize())#首字母大写

print(name.count("a"))#返回2，统计字母出现的次数

print(name.center(50,"-"))#打印50个字符，name字符串放中间不够用'-'补上

print(name.endswith("ex"))#判断结尾是否为"ex"，返回Ture或False

print(name.find("name"))#找第一个出现的字符串返回数值

print(name1.format(name2="alex",year=23))#格式化

print(name1.format\_map({'name2':'alex','year':12}))#字典格式化

print("12a".isalnum())#英文字符或数字

print("Aa".isalpha())#英文字符

print("12".isdecimal())#十进制数

print("123".isdigit())#整数

print("\_a123".isidentifier())#是不是一个合法的标识符命名

print("aaaaa@ %".islower())#是不是全部小写，非字母不判断

print("AAAAA@ %".isupper())#是不是全部大写，非字母不判断

print(" ".isspace())#一个或多个空格

print("My Name Is Alex".istitle())#字符串每个首字母是否大写

print(",".join(["1","2","3","4"]))#把一个字符串列表连接起来

print(name.ljust(50,"-"))#打印50个字符，name字符串放左面不够用'-'补上

print(name.rjust(50,"-"))#打印50个字符，name字符串放右面不够用'-'补上

print(name.lower())#变小写

print(name.upper())#变大写

print(name.lstrip())#去掉左边空格回车

print(name.rstrip())#去掉右边空格回车

print(name.strip())#去掉2边空格回车

print(name.replace("a","A",1))#替换，控制替换个数

print(name.rfind("a"))#最右边相同字符的位置

print(name.split())#字符串按空格分成列表，也可以指定字符作为分隔符

print(name.splitlines())#字符串按换行分成列表

print(name.swapcase())#字符串大小写切换

print(name.title())#字符串每个首字母大写

print(name.zfill(50))#字符串全面补0直到长度为50

26.字典由键和对应值成对组成。字典也被称作关联数组或哈希表。字典是无序的没有下标。

27.字符串为只读，修改是生成新的字符串

28.集合set

29.文件操作

f=open(文件名,’r’,encoding=”utf-8”)#返回文件句柄，’r’只读； ’w’只写覆盖原来的； ’a’添加；

**’r+’**可读可在最后追加；’w+’先创建一个新文件并可读，写只能追加；’a+’追加并可读。’rb’二进制文件读，如网络传输(只能用二进制传输)，视频。’wb’二进制文件创建。’ab’二进制追加。

f=open(文件名,’wb’)

f.write(“hello”.encode())//需要转成二进制类型

for line in f: //高效的循环方法，内存只占用一行

print(line)

f.readline()//读一行

f.readlines()//读所有行转成列表

f.tell()//返回读到第几个字符

f.seek(n)//定位到第n个字符

f.flush()//刷新，强制写缓存中的数据

//在显示进度中的作用

import sys,time

for I in range(50)://进度条效果

sys.stdout.write(“#”)

sys.stdout.flush()//不存缓冲直接显示，没有它会缓冲字符，一次输出

time.sleep(1)

f.truncat()//没有参数清空，有数字n，会从开始向后长度n的字符保留，之后的全删除。

30.文件插入或删除，两种方法：a.将文件全部读入内存，在内存中修改后一次写入文件；b.一行行读取旧文件，并写入新的文件中，找到需要修改的内容修改后写入新文件，关闭文件后可以用新文件替换旧文件。

31.避免打开文件忘记关闭，with open('log','r') as f, open('log1','r') as f1:

32.一行代码不超过80个字符，用\来换行

33.windows默认中文字符编码是GBK。转码先转成UNICODE再转成需要的编码: s.decode(“当前的编码”).encode(“需要转的编码”)。在Python2里默认编码是ascii,在Python3里默认编码是unicode。用encode()转成unicode是bytes类型。decode()显示指定编码的字符。unicode的字符串没有decode()函数，可以先编码在解码s.encode(“utf-8”).decode(“utf-8”)。

s=u”中文” #字符串全加u代表是字符串编码是unicode

34.获取默认的编码

import sys

print(sys.getdefaultencoding())

35.编程方式：

面向对象🡪类🡪class

面向过程🡪过程🡪def，过程是没有返回值的函数。返回 值为None。

函数式编程🡪函数🡪def，函数是逻辑结构化和过程化的一种编程方法。

36.函数定义：定义一段代码的执行过程

def test(x): #def定义函数的关键字;test函数名;()内科定义形参

’’’函数说明’’’ #文档描述

x+=1 #函数体

return x #定义返回值

函数优点：代码重用；保持一致性；可扩展

返回值：=0：返回None；=1：返回object；>1：返回元祖（tuple）

函数定义了两个形参def test(x,y):

在函数调用中采用位置参数调用实参要按照形参x,y的位置顺序，也可以用关键字参数调用实参可以不按顺序，位置参数和关键字参数混用时关键字参数要在位置参数后面，先保证位置参数的顺序。

test(1,2); # <==> test(y=2,x=1); # <==> test(1,y=2);

默认参数，提前给形参赋（缺省）值def test(x,y=2): 可以在函数调用时省略实参赋值test(x)。

参数组定义，def test(\*args):，传进去多个位置参数组成一个元祖

test(1,2,3,4,5); # test(\*[1,2,3,4,5]);

字典参数组定义，def test(\*\*kwargs):，

test(name='you', age=20) # {'name': 'you', 'age': 20}以关键字参数的形式传进去字典

test(\*\*{'name': 'you', 'age': 20}) # {'name': 'you', 'age': 20}以字典的形式传进参数

参数组放在所有形参的最后。

37.高阶函数，一个函数接收另一个函数作为参数。

def add(x,y,fn):

return fn(x) + fn(y)

res = add(3,-6,abs)

print(res)

38.字符串转字典 eval(“字符串”)。

eval函数，官方解释为：将字符串str当成有效的表达式来求值并返回计算结果。

1、计算字符串中有效的表达式，并返回结果

>>> eval('pow(2,2)')

4

>>> eval("n + 4")

85

2、将字符串转成相应的对象（如list、tuple、dict和string之间的转换）

>>> a = "[[1,2], [3,4], [5,6], [7,8], [9,0]]"

>>> eval(a)

[[1, 2], [3, 4], [5, 6], [7, 8], [9, 0]]

>>> a = "{1:'xx',2:'yy'}"

>>> eval(a)

{1: 'xx', 2: 'yy'}

>>> a = "(1,2,3,4)"

>>> eval(a)

(1, 2, 3, 4)

3、将利用repr()转换的字符串再反转回对象

>>> list1 = [1,2,3,4,5]

>>> repr(list1)

'[1, 2, 3, 4, 5]'

>>> type(repr(list1))

<type 'str'>

>>> type(eval(repr(list1)))

<type 'list'>

>>> eval(repr(list1))

[1, 2, 3, 4, 5]

39. Python 有办法将任意值转为字符串：将它传入repr() 或str() 函数。

函数str() 用于将值转化为适于人阅读的形式，而repr() 转化为供解释器读取的形式。

40.装饰器，本质是函数，功能是装饰其它函数，就是为其它函数添加附加功能。新增功能不能修改被装饰函数的源代码。不能修改被装饰函数的调用方式。对于被装饰函数不会感觉到被修改了（透明）。高阶函数+嵌套函数=>实现装饰器

41.函数即变量，函数和变量一样申请到一块内存存放函数体，函数体只是一段字符串，函数名是函数体在内存中的地址。

匿名函数没有def显示的定义函数名

calc=lambda x:x\*3

print(calc(3)) -> 9

42.高阶函数：a.把一个函数名当做实参传给另一个函数（在不修改被调用函数源码的情况下为其添加功能）；b.返回值中包含函数名（不修改函数的调用方式）。满足任一条件就是高阶函数。

43.函数嵌套：在一个函数的函数体内用def去声明另外一个函数。

44.装饰器，在需要装饰的函数前加”@装饰函数”。在装饰函数内嵌套新的函数增加新的功能。

45.列表生成式：[I for I in range(10)], [func(I) for I in range(10)]

46.生成器，一边循环一边计算的机制，只有在调用时才生成，只记录当前位置，可以节省内存。(I for I in range(10))。不支持类似列表的切片取值，只能向后一个一个取出数据\_\_next\_\_()。函数生成器用yield x返回数据。yield类似断点，通过外部函数唤醒，\_\_next\_\_()只是唤醒，send()唤醒并传一个值。可以产生并行效果，叫协程。

47. 可以直接作用于for循环的对象统称为可迭代对象：Iterable。isinstance()可以判断一个对象是否是可迭代对象Iterable。

**可以被next()函数调用并不断返回下一个值的对象称为迭代器：Iterator**。isinstance()可以判断一个对象是否是迭代器对象Iterator。

48.把列表，字典，字符串等可迭代对象变成迭代器，使用iter()函数。

49.迭代器可以表示一个无限大的数据。Range()在Python3中是一个迭代器。

50.python内置方法

bytearray()把字符串转成可修改的字节数组

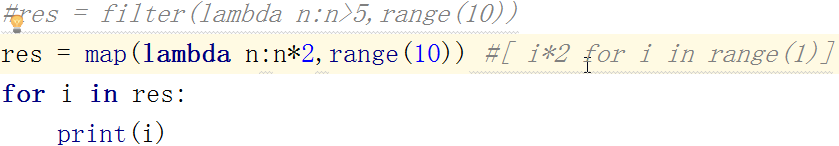
chr(97) ->’a’

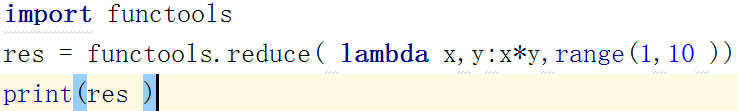
ord(’a’) ->97

exec()执行字符串形式的代码

dir()可以查看对象所有的方法。

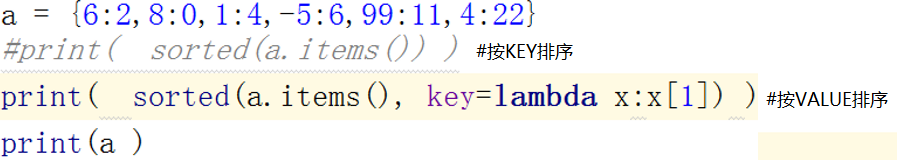
Filter() map()





globals()以字典形式返回当前程序里所有的变量和值

hash()产生唯一映射值



51. json不同语言之间进行数据交互

import json

json序列化， json.dumps()

Json反序列化， json.loads()

Pickle用法和json一样





52. 假设你的项目名为foo, 我比较建议的最方便快捷目录结构这样就足够了:

Foo/

|-- bin/

| |-- \_\_init\_\_.py

| |-- foo.py

|

|-- conf

| |-- \_\_init\_\_.py

| |-- settings.py

|

|-- core/

| |-- tests/

| | |-- \_\_init\_\_.py

| | |-- main.py

| |

| |-- \_\_init\_\_.py

| |-- main.py

|

|-- logs

|

|-- docs/

| |-- conf.py

| |-- abc.rst

|

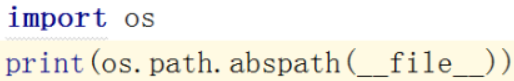
|-- setup.py

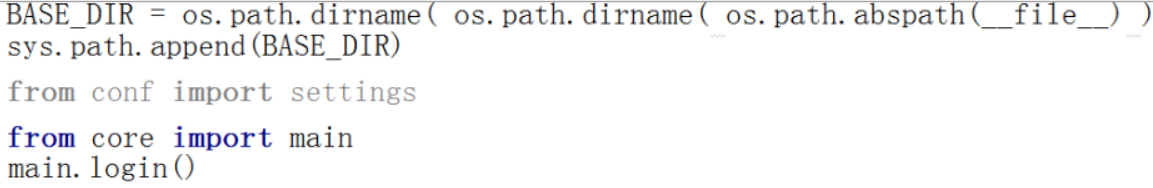
|-- requirements.txt

|-- README

1. bin/: 存放项目的一些可执行文件，当然你可以起名script/之类的也行。
2. foo/: 存放项目的所有源代码。(1) 源代码中的所有模块、包都应该放在此目录。不要置于顶层目录。(2) 其子目录tests/存放单元测试代码； (3) 程序的入口最好命名为main.py。
3. docs/: 存放一些文档。
4. setup.py: 安装、部署、打包的脚本。
5. requirements.txt: 存放软件依赖的外部Python包列表。
6. README: 项目说明文件。

53.返回文件的绝对路径





54. 实现ATM常用功能

https://github.com/triaquae/py3\_training/tree/master/atm

55.