# Python基础概念

1. 到底什么是Python（大蟒蛇）？（其他技术进行对比）

1)开源。 跨平台能力强（可移植性）啥都能干,号称胶水语言.

2)解释型语言。运行之前不需要编译。运行速度比编译语言要慢。

3)动态类型语言。不需要说明变量的类型。

4)适合面向对象的编程。每个元素都是变量。

5)丰富的库。

6)用途广泛

1.系统编程

2数据库编程 （自带gadfly DB-api => sql server, oracle, mysql）

3.网络编程 （全面支持sockets xml）

4.文本处理 （xml）

5.数学处理（标准数学库）

6.图形处理 （PIL TKinter）

7.多媒体应用 （pyOpenGL）

8.自动化

1. Python是如何进行内存管理的？

三个方面。

一、对象的引用计数机制：对象都有引用计数。

引用计数增加：

1，一个对象分配一个新名称

2，将其放入一个容器中（如列表、元组或字典）

引用计数减少：

1，使用del语句对对象别名销毁

二、垃圾回收

当一个对象的引用计数归零时，它将被垃圾收集机制处理掉。

三、内存池机制

Python提供了对内存的垃圾收集机制，但是它将不用的内存放到内存池而不是返回给 操作系统。（小于256个字节的对象直接使用内存池pymalloc，不频繁申请释放内存， 提高效率）

1. Python 测试框架

1. py.test ：

- python风格编写测试用例和套件

2. nose ：

- 比py.test更易用和维护！

3. zope ：

- zope框架大型代码

- -j 可以指定测试进程的个数，不是在单一进程中测试，多个cpu测试

并行测试 py.test最好，支持多操作系统，如果项目是在多个平台上，py.test可以胜任。

1. py.test单元测试框架

最大特点：测试驱动开发。自定义测试顺序，甚至跳跃。可以传递数据。

可以自动发现测试用例，凡是test\_\*.Py都会被检测到自动运行。

1.test fixture

用于初始化、清理等动作。

2.testcase

测试用例，unittest的最小单元。用以对指定输入的返回结果进行检测。在unittest中提供 了TestCase基类， 用来创建新的测试用例类。

3.test suite

测试套件，一系列测试用例或测试套件的集合。在unittest中由TestSuite类实现。

4.test runner

测试执行器，负责用例执行并生成测试报告，在unittest中提供了命令行模式和GUI模式来执行。

unittest的流程：写好TestCase，然后由TestLoader加载TestCase到TestSuite，然后由TextTestRunner来运行TestSuite，运行的结果保存在TextTestResult中，我们通过命令行或者unittest.main()执行时，main会调用TextTestRunner中的run来执行

1. 其他应用

数据库：pymysql

网页测试：selenium

代码覆盖测试：coverage.Py

1. Python都有哪些自带的数据结构？

可变：列表list[]、集合set{}、字典dict{} ；

不可变：字符串 str、元组 tuple() 、数 。

# 基础操作

1. hello world.

str = "Hello"

str += " world!"

print str

1. 列表（第一个list[0],最后一个list[-1]或者list[len(list)-1]）

基础：

mylist = [1, ["another", "list"], ("a", "tuple")]; (可以多层)

mylist = ["List item 1", 2, 3.14];

mylist[0] = "List item 1 again";

mylist[-1] = 3.21 ;

创建

Mylist = range(10)

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

A2 = [i for i in Mylist if i !=2]

Mylist = [i for i in range(10)]

Mylist.append(10)

Mylist.extend([11,12,13])

del Mylist[-1] (删除最后一个)

读取：

mylist = ["List item 1", 2, 3.14]

print mylist[:] 输出 ['List item 1', 2, 3.14]

print mylist[0:2] 输出 ['List item 1', 2] （左包含右不包含）

print mylist[-3:-1] 输出 ['List item 1', 2] （倒数第三个到倒数第一个）

print mylist[1:] 输出 [2, 3.14] （第二个到最后）

相关操作

mylist.sort() （方法，不返回list）

mylist.reverse() （方法，不返回list）

mylist[][] (二维列表数组)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

mylist = []

for i in range(0,10):

mylist.append([])

for j in range(0,10):

mylist[i].append(j)

print mylist[2][3]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 集合（没有顺序）

创建的2个方法

set1 = set([1,2,3,4])

set2 = {2,3,4,5}

set1.add(5)

set1.remove(1)

a = set1 | set2 并集 set([1, 2, 3, 4, 5])

b = set1 & set2 交集 set([2, 3, 4])

c = set1 - set2 差集 set([1])

d = set1 ^ set2 对称差集 set([1, 5])

1. 字典（没有顺序）

创建的3个方法

dict1 = {'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4}

dict1 = dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))

dict1 = dict(a=1, b=2, c=3)

dict1["f"] = 10 （添加）

del dict["f"] (删除)

1. 元组

mytuple = (1, 2, 3); #元组则是不可变的一维数组

1. 字符串

str.upper() #字母大写 这个是返回一个新的 str不可变

str.lower()

str.replace()

str.capitalize() #首字母大写 这个是返回的

1. 通过变量填充

name = 'zhangsan'

age = 25

print('my name is %s'%(name))

print('i am %d'%(age))

\是换行 \*号是几个

print "Name: %s\

Number: %d\

String: %s" % ("dd", 3, 3 \* "-")

输出 Name: dd Number: 3 String: ---

1. for \* in\* ,if \* in \*

for number in range(9):

if number in (7, 9):

print "number in (7, 9)";

break

else:

print " number is %d"%number;

Continue

1. lambda 匿名函数（适用于短小函数）

myfunc = lambda x,y: x+y

print myfunc(2,3)

1. 导入外部库。

import random

import time

from time import clock

1. 文件I / O

myfile = open(r"D:\\hello.txt", "w")

myfile.write("This is a sample string")

myfile.close()

myfile = open(r"D:\\hello.txt")

print myfile.read()

myfile.close()

1. 其他杂项

1)数值判断可以链接使用，例如 1<a<3 能够判断变量 a 是否在1和3之间。

2) lst1 = [1, 2, 3]

lst2 = [3, 4, 5]

List1 = [x \* y for x in lst1 for y in lst2] for in for in

List1 = [x for x in lst1 if 4 > x > 1] for in if

3)函数里用到全局变量 要用global申明

4)程序新自定义名字myfunction = len; #len是内置程序的名字，自己写的函数也可以重命名

5)当对字典进行in判断是，判断的是key(前面那个)

1. 装饰器: 让函数在不需要做任何代码变动的前提下增加额外功能。

def dec(func):

def wrapper(\*arg,\*\*kw):

print "Do something before func!!"

result = func(\*arg,\*\*kw)

print "Do something after func!!"

return result\*10

return wrapper

@dec

def plus(x,y):

return x+y

print plus(2,3) 输出 50

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

注意如果要调用函数中的参数可以用

from inspect import signature

sig = signature(func)

# 获得装饰器传来的参数， 函数签名与之绑定，字典类型

bind\_types = sig.bind\_partial(\*args, \*\*kwargs).arguments

If elif else

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 面试题

1. 【装饰器】 foo(2) = ？？

def dec(f):

n = 3

def wrapper(\*args,\*\*kw):

return f(\*args,\*\*kw) \* n

return wrapper

@dec

def foo(n):

return n \* 2

分析：前置：局部n=3 后置：\*n（这里是局部的3），执行： 先里面的2\*2 ，再外层的\*3 ， 2\*2\*3=12

1. 合法性

不合法 dict1 = {[1,2,3]: “uestc”}（字典的键必须是唯一，list不唯一）

合法 dict2 = {(1,2,3): “uestc”}（元祖唯一不能改变）

合法 x = y = z = 1

不合法 x = (y = z + 1) 这个 x = (y = z + 1) 中右半部分括起来后，相当于把一个赋值表达式赋值给变量 x

不能用数字作为变量开头

1. range

for i in range(2):

print i #打印 0，1 （左包含 右不包含）

for i in range(4,6):

print i #打印 4，5

for i in range(6,1,-1) 倒序

for i in range(6,1,-2) 倒序以2为间隔

1. bar(2)(3) ？

def bar(multiple):

def foo(n):

return multiple \*\* n

return foo

分析：（2是外层输入数据）（3是里层数据） 2\*\*3 ，2的3次方=8

1. print ‘a’> 'b' or 'c'

分析：‘a’> 'b' 是false ， false or ‘c’ 打印 c

1. CLASS类打印什么？

class Person:

def \_\_init\_\_(self):

pass

def getAge(self):

print \_\_name\_\_

p = Person()

p.getAge()

打印 \_\_main\_\_

1. 元组（tuple） a = (1, 2, 3) ，哪些操作合法？

b = a[1:-1] 合法, b = (2,) 第一个包含，到最后一个不包含。

b = a\*3 合法 b = (1,2,3,1,2,3,1,2,3)

a[2] = 4 不合法，元组不可改变

b = list(a) 合法， 变成list类型

1. python中主要存在四种命名方式：

1、object #公用方法

2、\_object #半保护，只有类对象和子类对象自己能访问，不能用import导入。

3、\_\_object #全私有，全保护 #私有成员“private”，意思是只有类对象自己能访问，连子类对象也不能访

问到这个数据，不能用import导入。

4、\_\_object\_\_#内建方法，用户不要这样定义

1. python解释型语言的特性有什么？

非独立（依赖执行环境），效率低（边运行边编译）

1. args与位置参数

def foo(x,y=1,\*args):

pass

foo (1,2,3,4,5) // 其中的x为1，y=1的值被2替换，3,4,5都给args，即args=(3,4,5)

1. try except esle finally

def exceptTest():

try:

#第1步

except KeyError, e:

#有错误 第2步

except IndexError, e:

#有错误 第2步

else:

#没错误 第2步

finally:

#第3步

1. a=range(10)

print a[-3] #7 倒数第三个

print a[2:6] #[2,3,4,5]（左包含右不包含）

print a[::3] #[0,3,6,9] start:end:间隔

print a[2-3] #a[-1] 9

1. copy

a = { '1' : 1, '2' : 2 }

b = a.copy()

a['1'] = 5

sum = a['1'] + b['1']

print sum

分析：a.copy()浅复制（copy父对象，子对象不copy） 5+1=6

import copy

a = [1, 2, 3, 4, ['a', 'b']]

b = a #只是地址，增加引用

c = copy.copy(a) #浅copy 只copy父对象

d = copy.deepcopy(a) # 深copy 完全独立

a.append(5)

a[4].append('c')

a= [1, 2, 3, 4, ['a', 'b', 'c'], 5]

b= [1, 2, 3, 4, ['a', 'b', 'c'], 5]

c= [1, 2, 3, 4, ['a', 'b', 'c']]

d= [1, 2, 3, 4, ['a', 'b']]

1. a= set([1,1,2,2,3,4])

print a

分析 ： 打印 set([1, 2, 3, 4])， set 会去重复的，因为set是没有顺序的

list1 = [1,1,2,2,3]

list2 = []

for i in list1:

if i not in list2:

list2.append(i)

1. append expand

numbers = [1, 2, 3]

numbers.append(4)

numbers.append([7,8,9])

numbers.extend([10,11,12])

print numbers

分析：[1, 2, 3, 4, [7, 8, 9], 10, 11, 12] ， expend只能加一个， extend可以加多个

1. 全局变量

x = 1

def change(a):

global x #要使用x 必须要先全局申明

x += 1

print x

change(x)

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

x = 1

def change(a):

a += 1 #或者老实的换成a

print a

change(x)

1. python严格区分大小写。 无法精确表示浮点数。

print type(1/2) =》 <type 'int'> 除法向下取整

print type(0.5) =》 <type 'float'>

1. map

>>> map(square, [1,2,3,4,5]) # 计算列表各个元素的平方

[1, 4, 9, 16, 25]

>>> map(lambda x: x \*\* 2, [1, 2, 3, 4, 5]) # 使用 lambda 匿名函数

[1, 4, 9, 16, 25]

1. 转义符

print "\nwoow" 打印 换行woow

print r"\nwoow" 打印 \nwoow （r 代表原始字符串标识符）

print "\\nwoow" 打印 \nwoow (\\为转义符)

1. 排序后 倒序打印

a=[1,2,10,4,6,7,5,3,8,9]

a.sort()

for i in range(len(a)-1,-1,-1): #注意i：9->0 , -1不包含（左包含右不包含）， 最后那个-1是倒序

print i," ",(a[i])

1. match()和search()的区别？

print(re.match(‘super’, ‘superstition’).span()) =》 (0, 5) match 从头开始匹配

print(re.match(‘super’, ‘insuperable’)) =》None match 不能从中间开始

print(re.search(‘super’, ‘superstition’).span()) =》 (0, 5) search 可以从中间或者前面开始

print(re.search(‘super’, ‘insuperable’).span()) =》(2, 7)

1. 随机数

random.randint(1,10) #1-10之间的随机int

random.random() #随机float浮点数

1. 单引号，双引号，三引号的区别

单引号和双引号是等效的，如果要换行，需要符号(\),三引号则可以直接换行。

1. 数字，字符串转换？

str(99) int（“99”）

1. 输出（看这个参数输入）

def f(x,l=[]):

for i in range(x):

l.append(i\*i)

print l

f(2) # [0,1]

f(3,[3,2,1]) #[3,2,1,0,1,4]

1. 继承 （注意这里要加object）

class A(object):

def go(self):

print "go A go!"

class B(A):

def go(self):

super(B, self).go()

print "go B go!"

b = B()

b.go()

先打印A 再打印B

1. xrange与range的区别

xrange 用法与 range 完全相同，range生成完整list对象，xrange 是一个生成器（效率高，不会一下子全部生 成，用一个生成一个）。

1. \*args,\*\*kwargs的作用是什么？如何使用？

\*args允许函数传入不定量个数的非关键字参数，\*\*kwargs允许函数传入不定量个数的关键字参数

1. 用装饰器实现一个首字母大写， 就是把函数作为一个参数传入upper

def start\_word\_upper(f):

def wrapper(\*args,\*\*kwargs):

w = f(\*args,\*\*kwargs)

w = w.capitalize() #capitalize首字母大写

return w

return wrapper

@start\_word\_upper

def greetins(word='hi there'):

return word

print greetins()

1. PyChecker是一个python代码的静态分析工具。
2. 字符串替换

str1 = "北京市麦达技术数字有限公司"

str2 = str1.replace("北京市","").replace("技术","").replace("有限","").replace("公司","")

输出 str1 = "麦达数字"

1. 把字符串“HELLO PYTHON”从大写字母全部转换成小写字母并换行显示，然后输出到计算机c盘的hello.txt文件中保存。

str = 'HELLO PYTHON'

new\_s = ""

for letter in str:

letter = letter.lower()

new\_s = new\_s+letter+"\n"

print new\_s

with open(r'D:/hello.txt','a') as f: # myfile = open(r’\*\*’,’a’)

f.write(new\_s)

1. 设计一个程序，求出1+3！+5！+7！+9！+50！的和。

# 求1+3！+5！+7！+9！

m = 0

for i in range(1,6):

n = 1

for j in range(1,2\*i):

n = n\*j

m += n

# 求50！

l = 1

for x in range(1,51):

l = l\*x

# 和

sum = m+l

print(sum)

1. 给定一个值为整数的数组int\_array,找出int\_array中第二大的整数。

int\_array = [2,1,3,5,4]

int\_array.sort() #倒序a.reverse()

result = int\_array[len(int\_array)-2]

print result

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int\_array = [2,1,3,5,4]

temp= int\_array[0]

pre= int\_array[0]

for i in range(1,len(int\_array)):

if (temp< int\_array[i]):

pre = temp

temp = int\_array[i]

else:

if (pre<int\_array[i]):

pre = int\_array[i]

print pre

1. cmp，map，filter的作用？

# cmp:compare比较

cmp(a,b) # a<b 返回-1 a>b 返回1 a=b返回0

# map：映射

map(function, iterable) # 将对应输出结果保存为一个list

# 例：

def plus\_one(x):

return x+1

map(plus\_one,[1,2,3,4]) ===>[2,3,4,5]

# filter:过滤器

filter(function, iterable) # 将 function依次作用于iterable的每个元素，如果返回值为true, 保留元素，否则从 iterable里面删除

# 例：

def bigger\_than\_three(x):

return (x>3)

filter(bigger\_than\_three,[1,2,3,4,5]) ===>[4,5]

1. 请阐述\_\_new\_\_和\_\_init\_\_的区别？

\_\_new\_\_:创建对象时调用，会返回当前对象的一个实例

\_\_init\_\_:创建完对象后调用，对当前对象的一些实例初始化，无返回值

调用顺序：先调用\_\_new\_\_生成一个实例再调用\_\_init\_\_方法对实例进行初始化，比如添加属性。

1. @classmethod、@staticmethod、@property都是啥意思？

class test():

a=0

def \_\_init\_\_(self,x):

self.a=x

print "\_\_init\_\_ a = ", self.a

@classmethod

def do\_it\_before\_create\_class(cls,x):

x=x\*10;

print "do\_it\_before\_create\_class"

return cls(x)

@staticmethod

def just\_do\_something\_static(x, y):

print "djust\_do\_something\_static x+y ="

return x + y

@property

def data(self):

print "get data a"

return self.a

print test.just\_do\_something\_static(1,2) #纯静态方法

print "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

A=test.do\_it\_before\_create\_class(1) # A是一个int ，类似于c++里面的类的静态方法一般用于预处理x， 然后创建实例

print "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

print A.data #直接得到数据

结果：

djust\_do\_something\_static x+y =

3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

do\_it\_before\_create\_class

\_\_init\_\_ a = 10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

get data a

10

1. 文章倒置 list= str.split() , list.reverse, str = ‘ ’.join(list),

字符串倒序str = str[::-1] ，注意没有reverse， str不可变。

def func(mystr):

result = ""

count = 0

for i in range(len(mystr)-1,-1,-1):

if i == 0:

count += 1

for j in range(0,count):

result += mystr[i+j];

else:

if str[i] == " " :

for j in range(1,count+1):

result += mystr[i+j];

result += " "

count = 0

else:

count += 1

return result

str = "this is a text reverse"

print func(str)

打印：reverse text a is this

def func2(str):

rev=str.split("")

rev.reverse()

result = " ".join(rev) #空格的方式join 列表

return result

1. 有如下代码，请写出输出

def test(x,l=[]):

for o in range(x):

l.append(o)

print l

test(3)

test(1,[3,2,1])

test(3)

输出如下：

[0,1,2]

[3,2,1,0]

[0,1,2,0,1,2]

可变类型作为参数在不显示传递参数时，每次函数调用时候都是共用的；如果传递了参数(比如第二种)则不会互相影响。

1. Python2与Python3的区别

1.性能

Py3.0运行速度比Py2.5慢30%。

2.编码

Py3.X源码文件默认使用utf-8编码，这就使得以下代码是合法的：

>>> 中国 = 'china'

>>>print(中国)

china

1、

py2: （1）源码重复量多，臃肿。

（2）源码语法不清晰，掺杂着c,php,java,的一些陋习

py3:几乎是重构后的源码，规范，清晰，优美

2、

py2:默认ASKLL编码不能识别中文。文件头需要添加：#-- encoding:utf-8 --

py3:默认编码方式就是utf-8

3、3. 语法

1）Py3.X去除了long类型，现在只有一种整型——int，但它的行为就像2.X版本的long

1. 深入了解装饰器 \*args,\*\*kwargs 参数的使用

def dec(func):

def warpper(\*args,\*\*kwargs):

print args

print kwargs

for k , v in kwargs.items(): #字典变成list元组去除

print k, v

result = func(\*args,\*\*kwargs)

return result

return warpper

@dec

def func\_1(a,ls):

return a\*ls[0]

@dec

def func\_2(a,\*\*kw):

return a

if \_\_name\_\_ =="\_\_main\_\_":

a = 1

ls = [2,3]

func\_1(a,ls) #打印 (1, [2, 3]) 、 {}

func\_2(a,x=1,y=2)#打印 (1,)、{'y': 2, 'x': 1}、y 2、x 1

1. Json

def createData(transStr):

data = {

'i': transStr ,

'doctype': 'json',

}

return data

r = requests.request("post", url, params=createData("handsome boy"))

d = json.loads(r.text)

str = d['translateResult'][0][0]['tgt']

print(str)