

# Part 1 : Introduction

运行 python 程序

—— 编辑器 (大多数)

—— 终端 (大型程序):

python \*.py

python 定义变量不用声明变量类型

(char, int, double, struct ...)

字符串处理:

x.	<u>lower()</u>	常用
x.	<u>upper()</u>	
x.	<u>title()</u>	

拼接:  $y = x_1 + x_2 + "Hello".$

空白处理

x.	<u>rstrip()</u>	←
x.	<u>(strip())</u>	+ ←
x.	<u>strip()</u>	—

(temporary)  
not forever)

print

python 2:	print "A"
python 3:	<u>print("A")</u>

函数

Other type → string: str()

Perl 语言与 Python 语言

灵活, 多变.

松弛

简约, 优美.

统一

## Part 2: List · tuple · dictionary · Set.

列表	元组	字典	集合
①	②	③	④

Comparison:

① List: 元素可重复，类型可不同，有序。

不同于 C 中 array 不同于 ③, ④

e.g. `listA = ['a', 1, [2]]`

② tuple: 与 list 一致，但不可修改。

e.g. `tupleA = ('a', 1)`

无法进行索引。

分片等

③ Dictionary: key : Value, 二者一一对应。无序。

e.g. `dictA = { 'name': 'AA', 'age': 21, 'B': { '3': 4 } }`

④ Set: 无序。不重复，为一组 key 的集合。有并、交、差、对称差的操作。创建 set 不同于 ①, ②, ③,

无直接插入式，可通过先 create ①/②，再调用: `set()`

得知。

e.g. `listA = ['a', 1, [2]]`

`setA = set(listA)`

list:

元素访问  $\Rightarrow$  list(index)

list[0] / list[-1]

第一个

最后一个，在

不知 length 时，也可取  
最后一个元素。

元素操作

revise.

add

delete.

· append(x)

· insert(index, x)

del

list[index]

x = list.pop() / pop(index)

· remove(value)

此前均为 index。若

不知其 index 时，用 remove.

组织 list

method (方法)

· sort()

· sort(reverse=True)  $\Rightarrow$  反向

sort

sorted(list)

$\Rightarrow$  暂时。

· reverse()

$\Rightarrow$  永久

function (函数)

len(list)

遍历 list:

for A-element in listA:

python 中用 indentation  
来表示 blocks

sentance A  
Sentance B

sentance C  
Sentance D

创建 list ————— 直接定义各个元素.

list() 函数:

listA = list(range(a, b))

[a, b)

逐个增加:

listA = [ ]

for x in range(a, b):

listA.append(kx+c)

任意操作

列表解析:

listA = [(kx+c) for x in range(a, b) ]

①

元素表达式

②

元素范围，个数

`slice`( 切片 ) :

`sublist A = listA [ a : b ] / [ a : ] / [ : b ] / [ a , b )`

可为元素数

e.g. `listAA = listA[ : ]` # real copy.

`listAA = listA` # pseudo copy.

类似于 C++ 中 array

性质：头指针指向同一地址，并无进行 real copy.

简单统计：

$$\left. \begin{array}{l} a = \max(\text{listA}) \\ b = \min(\text{listA}) \\ c = \mid \text{listA} \mid \end{array} \right\} \text{function}$$

# If 语句：

多条件单代码块：

① if condition 1:  
  --  
  elif condition 2:  
    --  
  elif condition 3:  
    --  
  else condition 4:  
    --

与 C/C++ 不同之处：

' else 不跟最后一个 elif,  
而是表如果以上均不满足. 类似于 switch ... case 中 default.

## 逻辑表达式

② if

condition 1

and  
or

condition 2 :

⇒ 不同于 C/C++ 中 &&, ||.

3种判断：

③ if

element\_A /  
  sublist A

in  
not in

list A : 元素是否在/不在其中.

if list A = 非 empty

Dictionary:

元素访问  $\Rightarrow$  提取 (key)。

$v = \text{dict}[\text{key}]$

e.g.  $x = \text{dict}['\text{name}']$

元素操作

- revise.  $\text{dict}[\text{key}] = \text{value}$
- add  $\text{dict}[\text{key}] = \text{value}$ . # 加入 key:value
- delete  $\boxed{\text{del}}$   $\text{dict}[\text{key}]$

遍历 Dictionary

遍历所有 item  $\Rightarrow \text{dict.items()}$   
遍历所有 key  $\Rightarrow \text{dict.keys()}$   
遍历所有 value  $\Rightarrow \text{dict.values()}$

△ 此三者遍回均为 list, 因此可用 list 的遍历操作。

△ values 重复的值可能很多, 列表中包含大量  
重复项, 可用 set 去除:

for element in set( dict.values() ):

=====

- - -

此时独一无二。

△ 可能需要按顺序 遍历 key 值, 用

for element in sorted( dict.keys() ):  
- - -

e.g. for k, v in dict.items():

for k in dict.keys():

for v in dict.values():

嵌套

① Dist [ List ]  $\Rightarrow$  每个元素均有不同特征，  
每个元素的特征类似。

② List [ Dist ]  $\Rightarrow$  每个元素均类似，但特  
征不同

③ Dist [ Dist ]  $\Rightarrow$  每个元素不同，特征  
不同

e.g. ①  $\Rightarrow$  Dist: 汽车 每个list: 一种部件

List 中元素: 部件中零件

②  $\Rightarrow$  Dist: 人 每个list: 户口本

Dist 中元素: 人的信息

③  $\Rightarrow$  Dist: 车铺 Dist 中元素: 不同车

子Dist 中元素: 车的不同零件

# Part 3 : function , class

导入模块 / 中的函数:

import module  $\Rightarrow$  用函数时，需 module.function

直接用 { from module import funct, func | ...  
func() { from module import \*

指定别名 { import module as md  
名 { from module import funct as f

function :

△ 传递 list 作实参。为防止 list 在 func 中

被修改，用 func (listA[:], ...)   
传副本

△ 传递任意数量实参：

def func(\*tuple-P)

... - - -

实参均被放入该 tuple 中

△ 传递任意数量关键字实参。

def func(\*\*dict-P)

... - - -

关键字实参均被放入该 dict 中

△ 函数参数过多  $\Rightarrow$  def funct(  
双 indentation, < [ ] P-O, P-I- ):  
与函数体区分开

其余性质均

与 C/C++

相同。

Class:

创建类

直接创建: class A():

继承: class A(parent-class):

△ Member variable 定义: 于 \_\_init\_\_(self, ...) 中

构造函数

△ 无析构函数

△ 所有的 member function 均是 public, 无 private, protected, 且第一个形参均为 self.

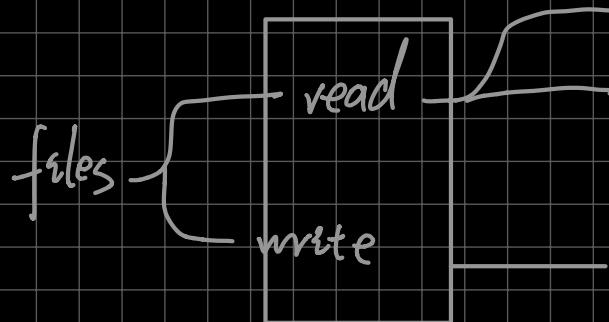
△ 子类通过 super().\_\_init\_\_(parameters...) 调用  
父类的构造函数进行初始化. 不包含 self.

△ 类的派生  $\Rightarrow$  于 \_\_init\_\_(self, ...) 中定义  
工具类对象.

△ 类的命名规范  $\Rightarrow$  马它峰命名法 class MyClass():

# Part 4 : files, exceptions

string = file-object.read()



listA = file-object.readlines()

返回 list，中 1 个元素为 1 (line。  
(此行可调 .split() 得到 1 个 list  
表 1 个 line，list 中元素为 line 中各列)

file-object.write("AAB\n")

先用 file object

表示一个文件。

with open(file-path, 'r') as file-object:

---

write 与打开文件方式有关。  
'w' 则覆盖，'a' 则追加。

'r'  
'w'  
'a'  
'rt'

不能使用 sentencesA

sentencesB

再使用 file-object，  
因为 with 之后自动调用 close()

exceptions :

常见: ZeroDivisionError, FileNotFoundError

try :

可能抛出 exception

代码块

except exception-name:

处理 exception / pass  
—— 什么都不能

e.g JSON file.

import json

FileName = ' \* .json '

write { with open ( FileName , ' w ') as file-object :  
        [ json . dump ( contents , file-object ) ] }

read { with open ( FileName , ' r ') as file-object :  
        [ load-content = json . load ( file-object ) ] }