# Python基础语法整理

## 1.注释

**单行注释** # 注释内容 （快捷键： ctrl + /）

**多行注释** """注释"""  '''注释1'''

（编码规范：一般习惯代码后面添加两个空格再书写注释文字）

**定义函数的说明文档**

def 函数名(参数):

""" 说明文档的位置 """

代码

......

**查看**

help(函数名)

## ****2．输入****

变量=input("提示信息")

## 3.输出

**格式化符号**

%s：格式化输出字符串

%d：格式化输出整数

%f：格式化输出浮点数

**f-字符串**

print(f'我的名字是{name}，今年{age}岁了')

**转义字符**

\n：换行

\t：制表符

**print结束符**

print（'内容',end=''）

## ****4.转换数据类型****

变量名=类型（变量名）

### 5.运算符

### 算数运算符（只列出了一部分）

|  |  |  |
| --- | --- | --- |
| // | 整除 | 9 // 4 输出结果为2 |
| \*\* | 指数 | 2 \*\* 4 输出结果为 16，即 2 \*2\* 2\* 2 |

（混合运算优先级顺序：()高于 \*\* 高于 \* / // % 高于 + -）

### 逻辑运算符

### And or not

## ****6.取随机数****

random.randint(a, b)   #返回 [a, b] 之间的整数，包含 a 和 b

## ****7.语句****

**7.1 条件语句**

**If 语句**

**（1）1个判定**

if 要判断的条件:

条件成立时，要做的事情

**（2）2个判定**

if 条件:

else:

**（3）n个判定**

if xxx1:

事情1

    elif xxx2:

事情2

    Elif/else xxx3:

事情3

**（4）if嵌套**

if 条件1:

满足条件1 做的事情1

         if 条件2:

满足条件2 做的事情1

**（5）if实现三目运算**

条件成立执行的表达式 if 条件 else 条件不成立执行的表达式

**7.2 循环语句**

**While循环**

**（1）基础**

while 条件:

条件满足时，做的事情1

i += 1  #循环

sum = sum + 1  #求和

**（2）while循环嵌套**

while 条件1:

条件1满足时，做的事情1

         while 条件2:

条件2满足时，做的事情1

**7.3 For循环**

**（1）基础**

for 临时变量 in 列表或者字符串等可迭代对象:

            循环满足条件时执行的代码

**7.4 循环结合else**

**While+else**

while 条件:

条件成立重复执行的代码

else:

循环正常结束之后要执行的代码

**For+else**

for 临时变量 in 序列:

重复执行的代码

...

else:

循环正常结束之后要执行的代码

**7.5 循环结束方式**

Break：整体跳出

Continue：跳出本次

**8.字符串**

**（1）表述方式：**

变量名= '字符串'    变量名 = "字符串"

例：name3 = ''' Tom '''  name4 = """ Rose """

（三引号字符串，支持换行）

**（2）下标切片**

print(变量名[下标])

 print(变量名[0:n]) # 取 下标0~n-1的字符

**8.2常见操作：**  
**（1）找子串：**

字符串序列.find/index(子串) #区别：index会报错

**（2）统计次数：**

字符串序列.count(子串) #返回 str在start和end之间 在 mystr里面出现的次数

**（3）替换：**

字符串序列.replace(旧子串, 新子串, 替换次数) #如果 count 指定，则替换不超过 count 次.

**（4）切片**：

字符串序列.split(分割字符, num)

**（5）插入**：

字符或子串.join(多字符串组成的序列)# mystr 中每个元素后面插入str,构造出一个新的字符串

**（6）改变书写方式**

1.第一个字符大写：

mystr.capitalize()

2.字符串的每个单词首字母大写/小写：

字符串. Title/lower（）

3.小写转大写：

字符串.upper()

4.删除字符串 左边/右边/两边的空白字符：

字符串.lstrip/rstrip/strip()

**（7）返回一个原字符串左/右对齐/居中并使用空格填充至长度 width 的新字符串：**

字符串.ljust/rjust/center

**(8) 检查字符串是否是以子串开头/结尾：**

字符串序列.startswith/endswith(子串, 开始位置下标, 结束位置下标)

**(9) 检测是否全为字母/数字/字母或数字：**

字符串.isalpha/isdigit()

**(10) 检测是否只含空格：**

字符串.isspace()

**9.列表**

**（1）格式：**

namesList = [   ，，，]

**（2）打印：**

print(namesList[0])#一个引号一个位置

**（3）查找：**

列表序列.index(数据, 开始位置下标, 结束位置下标)

**（4）统计**：

列表.count(数据)#指定数据在当前列表中出现的次数

**（5）访问列表⻓度**

len(列表) #即列表中数据的个数。

**（6）判断是否存在**：

in/not in#判断指定数据是否在某个列表序列

**（7）添加**

添加整个数据：

列表.append(数据)

逐一添加：

列表.extend（数据)

指定位置添加：

列表.insert(index, object) #在指定位置index前插入元素object

**（8）删除元素：**

del 目标 #目标为列表或数据#根据**位置**删除

列表序列.pop(下标)#删除**最后一个**

列表序列.remove(数据)#根据**数值**删除

列表.clear（） #**清空**列表

**（9） 修改数据：**

1.数据逆置：

列表.reverse()

2.排序：

列表序列.sort(reverse=False/true)# reverses属性表示排序规则，reverse = True 降序， reverse = False 升序（默认）

**(10) 复制**

列表. copy()

**(11) 列表遍历**

1.while 循环

name\_list = ['Tom', 'Lily', 'Rose']

i = 0

while i < len(name\_list):

    print(name\_list[i])

    i += 1

  2.for 循环

name\_list = ['Tom', 'Lily', 'Rose']

for i in name\_list:

print(i)

**10.元组**

一个元组可以存储多个数据，但元组内的数据是不能修改的。

**(1) 定义**

t1 = (10, 20, 30)  # 多个数据元组

t2 = (10,)  # 单个数据元组

**（2）打印**

print(type(元组))

元组数据不支持修改，只支持查找，#但是如果元组里面有列表，修改列表里面的数据则是⽀支持的。

**（3） 查找**

元组[位置]

index()  #查找某个数据，如果数据存在返回对应的下标，否则报错，语法和列表、字符串的index方法相同。

元组.count(数据)  #统计某个数据在当前元组出现的次数。

len(元组)  #统计元组中数据的个数。

**(4)应用**

**1.自动组包**

tt = (10,20,30)

a,b,c = tt# a= 10 b= 20 c= 30

print(a,b,c)

**2.自动解包**

t= 10, 202, 30  # t是元组类型

print(t)

print(type(t))

**3.格式化输出**

t = ('bill',30)

print('姓名是:%s,年纪是:%d'%t) #元组元素1占用姓名  元素2占用年纪

**4.让列表不可被修改,保护数据安全**

l = [10,20,30]

# 将列表变成元组 元素就不可变了

t = tuple(l)

**11.字典**

字典数据和数据顺序没有关系，即字典不支持下标，

**(1) 定义**

dict1 = { ： ：：}

**(2) 增加**

字典序列[key] = 值  #如果key存在则修改这个key对应的值；如果key不存在则新增此键值对

**(3) 删除**

del 字典[数据]

清空字典：字典.clear()

**（4）修改**

字典序列[key] = 值

#如果key存在则修改这个key对应的值 ；如果key不存在则新增此键值对，修改的是key后面的

**（5）查找**

 字典[key]

字典序列.get(key, 默认值)

字典.keys()#获取字典所有键

 字典.values()#前键后值，3.4互补

 字典.items()#获取键+值

**（6）字典的循环遍历**

字典 = {'name': 'Tom', 'age': 20, 'gender': '男'}#key：键，value：数值，item：元素

for key/value/item  in dict1.keys/values/items():    #注意复数

    print(key/value/item)

遍历字典的键值对

dict1 = {'name': 'Tom', 'age': 20, 'gender': '男'}

for key, value in dict1.items():

    print(f'{key} = {value}')

**12.集合**

特点

集合可以去掉重复数据；

集合数据是无序的，故不支持下标

{}代表的是空字典，不是空集合，创建空集合，需要set()

**（1）定义：**

集合=set（）

**（2）打印**

print(type(集合))

**（3）增加数据**

集合.add(数据)

集合.update([数据])  #追加列表

**（4）删除数据**

集合.remove(数据)   #删除集合中的指定数据，如果数据不存在则报错。

集合.discard(数据)  #删除集合中的指定数据，如果数据不存在也不会报错。

集合.pop(数据)  #随机删除集合中的某个数据，并返回这个数据。

**（5）查找数据**

数据 in/not in 集合

### 13 总结

**（1）公共方法**

|  |  |
| --- | --- |
| **函数** | **描述** |
| len() | 计算容器中元素个数 |
| del 或 del() | 删除 |
| max() | 返回容器中元素最大值 |
| min() | 返回容器中元素最小值 |
| range(start, end, step) | 生成从start到end的数字，步长为 step，供for循环使用 |
| enumerate() | 函数用于将一个可遍历的数据对象(如列表、元组或字符串)组合为一个索引序列，同时列出数据和数据下标，一般用在 for 循环当中。 |

强制类型转换

1.tuple(被转换的)  #作用：将某个序列转换成元组

2.list()  #作用：将某个序列转换成列表

3.set()  #作用：将某个序列转换成集合

**（2）字符串、列表、元组、字典、集合语法对比**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 增 | 删 | 改 | 查 |
| 字符串 | 字符或子串.join(多字符串组成的序列) |  |  | 字符串序列.index(子串) |
| 列表 | 列表.append(数据) | del 目标 | 列表序列.sort(reverse=False/true) | 列表序列.index(数据, 开始位置下标, 结束位置下标) |
| 元组 | index()： |  |  |  |
| 字典 | 字典序列[key] = 值 | del 字典[数据] | 字典序列[key] = 值 | 字典序列.get(key, 默认值) |
| 集合 | 集合.add(数据) | 集合.remove(数据) # | | 数据 in/not in 集合 |
|  |  |  |  |  |

### 14 函数

**14.0 变量**

**（1）概念**

全局变量：定义在函数外的变量

局部变量：定义在函数里的变量

**（2）修改**

global 变量1, 变量2

**14.1 无参函数**

**（1）定义函数**

def 函数名():

代码

**（2） 调用函数**

函数名()

**14.2 含参函数**

**（1）定义函数**

def 函数名(参数1, 参数2):

**（2）调用函数**

函数名(参数1, 参数2)

函数名（参数1，参数2）  #位置参数：调用时参数位置必须和定义时相同

函数名（参数2=，参数1=） #关键字参数：忽略了顺序，不填写则取默认值

**补充：不定长参数**

def functionname([formal\_args,] \*args, \*\*kwargs):

"""函数\_文档字符串"""

function\_suite

return [expression]

**14.3 带有返回值的函数**

**（1）把结果返回给调用者**

return 变量

**（2）保存返回值**

变量=函数名（参数1，参数2）

**14.4 lambda 表达式**

如果一个函数有一个返回值，并且只有一句代码，可以使用 lambda简化。

lambda 参数列表 ： 表达式

**分类**

lambda的参数形式

lambda: 表达式

一个参数

lambda 参数: 表达式

默认参数

lambda key=value: 表达式

不定长位置参数

lambda \*args: 表达式

不定长关键字参数

lambda \*\*kwargs: 表达式

**14.5 高阶函数**

**map(function, list)**   #根据提供的函数对指定序列做映射.

**reduce(function, list)  #** 对参数序列中元素进行累加.

**filter()  #**用于过滤序列, 过滤掉不符合条件的元素, 返回一个 filter 对象, 如果要转换为列表, 可以使用 list() 来转换.

**sort(） #**用来对容器中元素进行排序

## ****15 文件****

**15.1 文件的主要操作**

**(1) 打开文件**

f.open(目标文件名的字符串,访问模式：只读、写入、追加等)  #打开/创建文件

**（2）关闭文件**

f.close()

**(3) 写文件**

f.write#写数据

**（4）读数据**

f.read(数据长度)  #读数据

f.readlines（） # 一次性读取

**（5）获取当前读写位置**

f.tell()  #获取当前读写位置

**（6）定位**

seek(o偏移量，方向)  #0:文件开头 1：当前位置  2：文件末尾

**15.2 文件的相关操作**

**（1）重命名**

import os

os.rename( )

**（2）删除文件**

import os

os.remove( )

**（3）创建文件夹**

import os

os.mkdir( )

**(4) 获取当前目录**

import os

os.getcwd()

**（5）改变默认目录**

import os

os.chdir("../")

**（6）获取目录列表**

import os

os.listdir("./")

**（7）删除文件夹**

import os

os.rmdir( )

**说明**

|  |  |
| --- | --- |
| **模式** | **描述** |
| **r** | **以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。** |
| **rb** | **以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。这是默认模式。** |
| **r+** | **打开一个文件用于读写。文件指针将会放在文件的开头。** |

**功能**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **功能** | **r** | **r+** | **w** | **w+** | **a** | **a+** |
| 读 | + | + |  | + |  | + |
| 写 |  | + | + | + | + | + |
| 创建 |  |  | + | + | + | + |
| 覆盖 |  |  | + | + |  |  |
| 指针在开始 | + | + | + | + |  |  |
| 指针在结尾 |  |  |  |  | + | + |

## 15.面向对象

**15.1 三大特征：封装、继承、多态**

**15.2 类**

理解：类别，范围大（玫瑰花）

class 类名:

方法列表

**15.2 对象**

理解：具体的，（小王子的玫瑰花）

对象名1 = 类名()

对象名2 = 类名()

对象名3 = 类名()

**15.3 self 和 info**

**info：**实例方法，第一个参数一定是self

**self：** 指的是调用该函数的对象。

**15.4 \_\_init\_\_()方法**

作用：初始化对象。

有参数在（）里直接加参数

**15.5 \_\_str\_\_()方法**

如果类定义了**str**方法，那么就会打印从在这个方法中 return 的数据。