# Python关键整理

1. **头文件**
   1. 标准库
      1. Sys —— (system)python解析器相关（命令行参数等）
      2. os —— (operation system)操作系统相关
      3. math —— C标准数学函数（除复数）
      4. random —— 随机数生成
      5. pikle —— 数据持久化（仅python程序可读）
      6. subprocess —— 系统进程操作
      7. queue —— 队列操作
      8. StringIO —— 字符串储存操作
      9. logging —— 日志相关
      10. json —— JSON格式数据操作
      11. time(datatime) —— 系统时间（超过2038年用datatime）
      12. re —— (regular expression)正则表达式处理
      13. csv —— CSV格式文件操作
      14. threading —— 多线程操作
      15. asyncio —— 异步I/O
      16. abc —— 抽象类实现
      17. ……
   2. other
      1. ?
2. **标识符**
   1. 第一个字符必须是字母表中字母或下划线 \_
   2. 标识符的其他的部分由字母、数字和下划线组成。
   3. 标识符对大小写敏感。
3. **关键字（保留字）**

不能把它们用作任何标识符名称

* 1. Python 的标准库提供了一个 keyword 模块，可以输出当前版本的所有关键字

{

>>> import keyword

>>> keyword.kwlist

}

* 1. 总结记录

end 将结果输出到同一行，或者在输出的末尾添加不同的字符

1. **宏定义**
2. **转义字符**

\(在行尾时) 续行符

\\ 反斜杠符号

\' 单引号

\" 双引号

\a 响铃

\b 退格(Backspace)

\000 空

\n 换行

\v 纵向制表符

\t 横向制表符

\r 回车，将 \r 后面的内容移到字符串开头并替换

\f 换页

\yyy 八进制数，y代表 0~7 的字符，例如：\012 代表换行。

\xyy 十六进制数，以\x 开头，y 代表的字符

\other 其它的字符以普通格式输出

1. **格式规定**
   1. python使用缩进来表示代码块
      1. 同一个代码块的语句必须包含相同的缩进空格数
   2. 多行语句
      1. 句末反斜杠连接’\’
      2. 括号内语句可跨行 —— () , [] , {}
   3. 注释
      1. 单行注释 —— ‘#’
      2. 多行注释 —— ’’’ 或者 “””
   4. with-as
   5. \\
2. **数据类型**
   1. 标准数据类型
      1. 不可变数据
         1. 数字 —— Number
            1. 整型数 —— int
            2. 布尔 —— bool
            3. 浮点数 —— float
            4. 复数 —— complex
         2. 字符串 —— String
         3. 元组 —— Tuple
   2. 可变数据
      1. 列表 —— List
      2. 字典 —— Dictionary
      3. 集合 —— Set
   3. 高级数据类型
      1. 字节数组 —— bytes

PS: list , tuple , string 都属于序列(sequence)

1. **数据操作应用**
   1. 浮点数定义 —— x.y 或者 x.ye(E)z
   2. 复数定义 —— x + yj 或者 complex( x , y )
   3. 字符串（字符）
      1. Python 中的字符串不能改变（类比C语言常量）
      2. Python 没有单独的字符类型，一个字符就是长度为 1 的字符串
      3. 定义：string\_name = “…”
         1. ‘ 和 “ 完全相同
         2. 三引号指定多行字符串，符号位于段落首尾

（ ‘‘‘ 或者 “““ ）

* + - 1. 转义符’\’

—— 正常情况下转义，如果字符串首添加字母’r’则不发生转义

* + - 1. 字面意义连接字符串（类比C语言）

—— “Hello ” “world !” == “Hello world !”

* + 1. 运算符操作字符串（类比C++字符串类）
       1. —— ‘+’ 连接字符串
       2. —— ‘\*’ 重复字符串
       3. —— [] 索引、截取字符串
       4. —— r/R原始字符串（转义字符等无效）
    2. 字符串引用（截取）格式 —— 变量[头下标:尾下标:步长]
       1. 头下标

引用（截取）起始处，不可省略

该处下标处字符会被读取操作

* + - 1. 尾下标

引用（截取）结尾处，可省略（默认到字符串尾）

该处下标处字符不会被读取操作

* + - 1. 步长

两次引用之间的移动步数，可省略（间隔=步长-1）

负数则自右向左移动

* + - 1. 字符串索引

从左往右 ‘0’

从右往左 ‘-1’

* + 1. 字符串格式化（类比C语言）
       1. 格式指令
          1. %d
          2. %u
          3. %o
          4. %x(%X)
          5. %f
          6. %e(%E) 用科学计数法格式化浮点数
          7. %g(%G) %e,%f(%E,%f)简写
          8. %c
          9. %s
          10. %p 用十六进制数格式化变量的地址
       2. 辅助指令
          1. \* 定义宽度或者小数点精度
          2. - 用做左对齐
          3. + 在正数前面显示加号( + )
          4. <sp> 在正数前面显示空格
          5. # 在八进制数前面显示零('0')

在十六进制前面显示'0x'或者'0X'

(取决于用的是'x'还是'X')

* + - * 1. 0 显示的数字前面填充'0'而不是默认的空格
        2. % '%%'输出一个单一的'%'
        3. (var) 映射变量(字典参数)
        4. m.n. m 是显示的最小总宽度

n 是小数点后的位数(如果可用的话)

* 1. 列表
     1. 数据类型阐述：[类比python字符串、C语言（结构体）数组]
        1. 有序对象集合
        2. 列表（集合）中元素类型可修改（字符串不可改）
        3. 列表（集合）中元素类型可不同
        4. 以偏移来存取元素
     2. 定义：List\_name = [ elem1 , elem2 , … ]
     3. 引用截取：变量[头下标:尾下标:步长]

（类比字符串）

* + - 1. 运算符操作（类比字符串）
      2. 内置方法？
  1. 元组
     1. 数据类型阐释：
        1. 与列表类似
        2. 元组元素不可修改（字符串可视为特殊元组）
        3. 元组中元素可包含可变对象
     2. 定义：Tuple\_name = ( elem1 , elem2 , … )

—— 特殊元组格式：

# 空元素元组 Tuple\_name = ()

# 单元素元组 Tuple\_name = ( elem , )

* + 1. 引用截取：变量[头下标:尾下标:步长]

（类比字符串）

* + 1. 运算符操作（类比字符串）
  1. 集合
     1. 数据类型阐释：
        1. 无序性（类比数学）
        2. 唯一性（类比数学）
        3. 可变性
     2. 定义格式：（可为空）
        1. set\_name = { elem1 , elem2 , … }
        2. set\_name = set( elem1 , elem2 , … )
     3. 操作应用
        1. ?
  2. 字典
     1. 数据类型阐释（对比list）
        1. 无序对象（键：值）集合
        2. 以键来存取
        3. 映射（数据）类型
        4. 键必须使用不可变类型
        5. （元素）无序性、唯一性
     2. 定义格式：
        1. dict\_name = { key1 : value1 , … }
     3. 操作应用?
  3. 字符数组
     1. 数据类型阐释
        1. 不可变的二进制序列
        2. 整数值元素（0~255）
        3. 常用于处理传输二进制数据
        4. 元素不可变
     2. 定义格式：
     3. 操作应用?

1. **数据类型转换**
   1. 转换方式
      1. 隐式转换 —— 自动完成
      2. 显式转换 —— 类型函数
   2. 类型函数
   3. int(x , base)

将x转换为一个整数

* 1. float(x)

将x转换到一个浮点数

* 1. complex(real,imag) / complex(“x+yj”)

创建一个复数

* 1. chr(x)

将一个整数转换为一个字符

* 1. ord(x)

将一个字符转换为它的整数值

* 1. hex(x)

将一个整数转换为一个十六进制字符串

* 1. oct(x)

将一个整数转换为一个八进制字符串

* 1. str(x)

将对象 x 转换为字符串

* 1. repr(x)

将对象 x 转换为表达式字符串

* 1. eval(str)

用来计算在字符串中的有效Python表达式,并返回一个对象

* 1. tuple(s)

将序列 s 转换为一个元组

* 1. list(s)

将序列 s 转换为一个列表

* 1. set(s)

转换为可变集合

* 1. frozenset(s)

转换为不可变集合

* 1. dict(d)

创建一个字典。d 必须是一个 (key, value)元组序列。

**运算符（类比**C**语言）**

1. 算术运算符
   1. +
   2. -
   3. \*
   4. /
   5. % —— 取模
   6. \*\* —— 幂（x^y）
   7. // —— 整除（舍去小数）

交互模式中，单次运算结果会赋给变量’\_’

1. 比较（关系）运算符
   1. ==
   2. !=
   3. > (>=)
   4. < (<=)
2. 赋值（复合）运算符
   1. = , += , -= , \*= , /= , %= , \*\*= , //=
   2. := —— 海象运算符

在表达式中同时进行赋值和返回赋值

* 1. ‘=’ 赋值格式：

var1 , var2 , … = val1 , val2 , …

1. 逻辑运算符
   1. and —— && (C language)
   2. or —— || (C language)
   3. not —— ! (C language)
2. 位运算符
   1. & , | , ^ , ~ , << , >>
3. 成员运算符（返回bool值 true/false ）
   1. in
   2. not in
4. 身份运算符（比较两个对象的存储单元是否为同一对象）
   1. is
   2. is not

“””

is 用于判断两个变量引用对象是否为同一个

== 用于判断引用变量的值是否相等。

“””

1. 运算符优先级
   1. 分级说明
      1. 相同单元格内的运算符具有相同优先级
      2. 相同单元格内的运算符从左至右分组（除了幂运算反向）
      3. 运算符均指二元运算，除非特别指出。
   2. 分级（查阅链接）

<https://www.runoob.com/python3/python3-basic-operators.html#ysf8>

I/O**输入输出**

1. 标准I/O
2. 文件I/O

**程序基本结构**

1. 顺序
   1. 占位语句 —— pass
2. 分支
3. 循环

**交互式编程**

?

**脚本式编程**

?

**推导式（数据处理）**

1. 概念阐述
   1. 数据处理方式
   2. 从一个数据序列构建另一个新的数据序列
2. python3.11支持的推导式
   1. 列表

new\_list = [expression for var in org\_list (if … )]

* 1. 字典

new\_dict = [key\_exp: val\_exp for var in collection (if … )]

var —— key, value, ……

collection —— 集合类数据容器（包括range()区间等）

* 1. 集合

new\_set = {expression for var in org\_set (if … )}

* 1. 元组

new\_generator = (expression for var in collection (if … ))

new\_tuple = tuple(new\_generator)

collection —— 集合类数据容器（包括range()区间等）

**迭代器&生成器**

1. 迭代器
   1. 概念阐述
      1. 记忆遍历的位置的对象
   2. 操作使用
      1. 生成对象

iter()

* + 1. 遍历对象（只可向前不可向后）

next()

* + 1. 中止遍历

StopIteration

1. 生成器
   1. 概念阐述
      1. 只能产生迭代器对象
   2. 操作使用
      1. yield

**函数**

1. 概念阐述
2. 组成要素
   1. 函数名
   2. 函数体（内容）
   3. 返回值
   4. 参数（对象）
      1. 必须参数

需要按顺序

* + 1. 关键字参数

不需要按顺序（python参数匹配机制）

def function\_name(arg1, arg2 ):

…

function\_name(arg1 = val1, arg2 = val2)

* + 1. 默认参数
    2. 不定长参数

带\* 参数（未命名）会以元组(tuple)的形式导入存放

带\*\*参数（未命名）会以字典{dict}的形式导入存放

* + 1. 其他

单独出现’\*’，则’\*’后的参数必须用关键字导入

1. 定义格式（注意缩进）

def function\_name(arg1, … , \*var\_args ):

“””

文档字符串

“””

body of the function

return [expression]

1. 操作应用
   1. 函数调用
   2. 参数传递
      1. 可变对象
         1. 改变对象值（内容）
      2. 不可变对象
         1. 创造新的对象（舍弃原来的对象）
   3. 匿名函数(lambda)
      1. 概念
      2. 定义格式：

return\_var = lambda arg\_list : expression

**文件操作**

1. 、、

**正则表达式**

……

见“正则表达式”

**面向对象编程**(OOP)

1. 这里默认指python3.x定义的新式类
2. 概念阐述
   1. 特征
      1. 封装
      2. 继承
      3. 多态
   2. 类
   3. 对象
3. 组成要素
   1. 类 —— 实例的抽象
      1. 内部类
         1. 直接使用外部类调用内部类；
         2. 先对外部类进行实例化，然后再实例化内部类。
      2. 抽象类
      3. 类间关系
         1. 依赖关系
         2. 关联关系
         3. 继承关系
   2. 字段
      1. 普通字段
      2. 静态字段
   3. 属性 —— 数据成员/数据封装（名字）
      1. 属性类别（等级区分）
         1. 类级
         2. 实例级（默认共享类级别属性）
      2. 属性类别（使用范围区分）
         1. 公有属性 —— 默认（无特别声明）
         2. 私有属性 —— “\_\_”（双下划线）开头成员
         3. 内置属性 —— 系统默认添加（一般前后有双下划线）
      3. 内置属性
         1. \_\_name\_\_ —— 类名字
         2. \_\_doc\_\_ —— 类文档字符串
         3. \_\_bases\_\_ —— 所有父类构成的元组
         4. \_\_dict\_\_ —— 类属性
         5. \_\_module\_\_ —— 类定义所在的模块
         6. \_\_class\_\_ —— 实例所对应的类
      4. 操作原则
         1. 属性的获取是按照从下到上的顺序来查找属性；
         2. 类和实例是两个完全独立的对象；
         3. 属性设置是针对实例本身进行的。
   4. 方法 —— 函数（行为）封装（对属性的操作）
      1. 方法类别（使用范围区分）
         1. 公有方法 —— 仅实例化对象调用
         2. 私有方法 —— “\_\_”（双下划线）开头成员，外部类&对象不可调用
         3. 类方法 —— 类&对象均可调用
         4. 静态方法 —— 类似“全局方法”（类和所有实例化对象共享）
         5. 魔术方法 —— 前后有双下划线
      2. 参数self

一个类可以产生多个对象，当某对象调用（类）方法时，该对象就会把自身的引用作为第一个参数自动传给该方法。

* 1. 实例（对象）

1. 定义格式

class class\_name(base\_class):

“””

# class documentation string #类文档字符串说明解释

“””

data\_name(= value) #公有属性

\_\_data\_name(= value) #私有属性

def public\_method(self, arg1, … ): #公有方法

body of the method

def \_\_private\_method(self, … ): #私有方法

body of the method

@classmethod #类方法

def class\_method(self, … ):

body of the method

@staticmethod #静态方法

def static\_method(arg1, … ):

body of the method

def \_\_method\_\_(self, … ): #魔术方法

body of the method

class innerclass\_name(): #内部类

……

object\_name = class\_name(arg1, arg2, … )

#类实例化（创建对象）

1. 操作应用
   1. 类定义
      1. class为关键字
      2. base\_class为所继承类，可省略（将默认或等效为继承object类）
      3. object为一切类的父类（原始类）
   2. 对象创建（类实例化）
      1. 依赖
      2. 关联
      3. 继承
   3. 访问调用