

python_basic

[官方文档很详细](#)

python简介

- 1 1. 程序员：
2 程序设计人员。
3
- 4 2. 程序：
5 一组计算机能识别和执行的指令，是实现某种需求的软件。
6
- 7 3. 操作系统：
8 管理和控制计算机软件与硬件资源的程序；
9 隔离不同硬件的差异，使开发程序简单化。
10 例如，Windows, Linux, Unix。
11
- 12 4. 硬件：
13 主板--计算机的主要电路系统。
14 CPU --主要负责执行程序指令，处理数据。
15 硬盘--持久化存储数据的记忆设备，容量大，速度慢。
16 内存--临时存储数据的记忆设备，容量小，速度快。
17 IO设备--键盘、鼠标、显示器。

python定义



- 1 是一个免费、开源、跨平台、动态、面向对象的编程语言。
- 2 Python是一种跨平台的计算机编程语言，只要该平台下有Python解释器，那么Python编写的程序就能在该平台运行，目前Python支持的平台主要有：Linux, Windows, Mac os, Android, iOS等。
Python完全可以创建大型软件，例如豆瓣、知乎服务端。Python语言底层是基于C语言，开发效率高，但是运行效率低。

python程序的执行方式

交互式

- 1 在命令行输入指令，回车即可得到结果。
- 2 1. 打开终端
- 3 2. 进入交互式: python3, ipython3
- 4 3. 编写代码: print("hello world")
- 5 4. 离开交互式: exit() or exit

文件式

- 1 将指令编写到.py文件，可以重复运行程序。
- 2 1. 编写文件。
- 3 2. 打开终端
- 4 3. 进入程序所在目录：cd 目录
- 5 4. 执行程序：python3 文件名

linux 常用命令

- 1 1. pwd: 查看当前工作目录的路径
- 2 2. ls: 查看指定目录的内容或文件信息
- 3 3. cd: 改变工作目录（进入到某个目录）
- 4 练习:
- 5 1. 在指定目录创建python文件.
- 6 --目录: /home/tarena/1905/month01
- 7 --文件名: exercise01.py
- 8 2. 在文件中写入: print("你好，世界!")
- 9 3. 运行python程序

执行过程



- 1 计算机只能识别机器码(1010)，不能识别源代码(python)。
- 2 1. 由源代码转变成机器码的过程分成两类：编译和解释。
- 3 2. 编译：在程序运行之前，通过编译器将源代码变成机器码，例如：C语言。
- 4 -- 优点：运行速度快
- 5 -- 缺点：开发效率低，不能跨平台。
- 6 3. 解释：在程序运行之时，通过解释器对程序逐行翻译，然后执行。例如Javascript
- 7 -- 优点：开发效率高，可以跨平台；
- 8 -- 缺点：运行速度慢。
- 9 4. python是解释型语言，但为了提高运行速度，使用了一种编译的方法。编译之后得到pyc文件，存储了字节码（特定于Python的表现形式，不是机器码）。导模块才会编译。
- 10 源代码 -- 编译 --> 字节码 -- 解释 --> 机器码
- 11 |-----1次-----|

解释器类型

- 1 1. CPython (C语言开发)
- 2 2. Jython (java开发)
- 3 3. IronPython (.net开发)

数据基本运算

pycharm常用快捷键

1. 移动到本行开头: **home**键
2. 移动到本行末尾: **end**键
3. 注释代码: **ctrl + /**
4. 复制行: **ctrl + d**
5. 选择列: 鼠标左键 + **alt**
6. 移动行: **shift + alt + 上下箭头**
7. 智能提示: **Ctrl + Space**
8. **Alt + 回车**

注释

- 1 给人看的, 通常是对代码的描述信息。
- 2 1. 单行注释: 以**#号**开头。
- 3 2. 多行注释: 三引号开头, 三引号结尾。

函数

- 1 表示一个功能, 函数定义者是提供功能的人, 函数调用者是使用功能的人。
- 2 例如:
- 3 1. **print**(数据) 作用: 将括号中的内容显示在控制台中
- 4 2. 变量 = **input**("需要显示的内容") 作用: 将用户输入的内容赋值给变量

变量

- 1 1. 定义: 关联一个对象的标识符。
- 2 2. 命名: 必须是字母或下划线开头, 后跟字母、数字、下划线。
- 3 • 不能使用关键字(蓝色), 否则发生语法错误: **SyntaxError: invalid syntax**。
- 4 3. 建议命名: 字母小写, 多个单词以下划线隔开。
- 5 • **class_name**
- 6 4. 赋值: 创建一个变量或改变一个变量关联的数据。
- 7 5. 语法: 变量名 = 对象
- 8 • 变量名1 = 变量名2 = 对象
- 9 • 变量名1, 变量名2, = 数据1, 数据2 优雅
- 10 或者 变量名1, 变量名2, = 对象1, 对象2
- 11
- 12 价值: 在内存中操作数据的技术
- 13 定义: 关联一个对象的标识符。
- 14 作用: 在内存中存储数据。
- 15 适用性: 一切对象。
- 16 语法: 变量名 = 对象
- 17 • 语义: 内存图
- 18 • 变量名: 真实内存地址的别名
- 19 • 见名知意
- 20 • 赋值号: 将右边对象的地址复制给左边内存空间。
- 21
- 22 """
- 23 变量
- 24 程序运行在哪里? -- 内存
- 25 程序在处理什么? -- 数据

```

26         价值：在内存中操作数据的技术
27         变量就是内存中的一块地
28     语法：
29         变量名称 = 数据
30         变量名称1,变量名称2 = 数据1,数据2    优雅
31         变量名称1 = 变量名称2 = 数据
32     赋值号=: 将右边的结果复制一份给左边
33
34     删除变量：
35         del 变量名称
36         del 变量名称1,变量名称2
37     """
38
39     # 创建单个变量
40     person_name01 = "金海"
41     # 用多个数据，创建多个变量
42     person_name02, person_name03 = "铁林", "徐天"
43     # 用单个数据，创建多个变量
44     person_name04 = person_name05 = "柳如丝"
45
46     del person_name04
47     del person_name01, person_name02
48
49     person_name06 = person_name05 + person_name05

```

del 语句

- 1 1. 语法：
- 2 del 变量名1, 变量名2
- 3 2. 作用：
- 4 用于删除变量,同时解除与对象的关联.如果可能则释放对象。
- 5 3. 自动化内存管理的引用计数：
- 6 每个对象记录被变量绑定(引用)的数量,当为0时被销毁。

核心数据类型

- 1 1. 在python中变量没有类型，但关联的对象有类型。
- 2 2. 通过type()函数可查看。

空值对象None

- 1 1. 表示不存在的特殊对象。
- 2 2. 作用：占位和解除与对象的关联。

整形int

- 1 1. 表示整数，包含正数、负数、0。
- 2 如： -5, 100, 0
- 3 2. 字面值：
- 4 十进制： 5
- 5 二进制： 0b开头，后跟1或者0 binary
- 6 八进制： 0o开头，后跟0~7 octal
- 7 十六进制： 0x开头，后跟0~9,A~F,a~f hex
- 8 3. 小整数对象池：CPython 中整数 -5 至 256,永远存在小整数对象池中,不会被释放并可重复使用。

浮点型float

1. 表示小数，包含正数、负数，`0.0`。
2. 字面值：
小数：`1.0` `2.5`
科学计数法：`e`/`E`（正负号）指数
- `1.23e-2`（等同于`0.0123`）
- `1.23456e5`（等同于`123456.0`）

字符串str

- 1 是用来记录文本信息（文字信息）。
- 2 字面值：双引号

复数complex

- 1 由实部和虚部组成的数字。
- 2 虚部是以`j`或`J`结尾。
- 3 字面值：`1j` `1+1j` `1-1j`

布尔bool

- 1 用来表示真和假的类型
- 2 `True` 表示真（条件满足或成立），本质是`1`
- 3 `False` 表示假（条件不满足或不成立），本质是`0`

数据类型转换

1. 转换为整形：`int`(数据)
2. 转换为浮点型：`float`(数据)
3. 转换为字符串：`str`(数据)
4. 转换为布尔：`bool`(数据)
- 5 结果为`False`: `bool(0)` `bool(0.0)` `bool(None)`
5. 混合类型自动升级：
7 `1 + 2.14` 返回的结果是 `3.14`
8 `1 + 3.0` 返回结果是： `4.0`

运算符

算术运算符

- 1 `+` 加法
- 2 `-` 减法
- 3 `*` 乘法
- 4 `/` 除法：结果为浮点数
- 5 `//` 地板除：除的结果去掉小数部分
- 6 `%` 求余
- 7 `**` 幂运算
- 8 优先级从高到低： `()`
- 9 • `**`
- 10 • `*` `/` `%` `//`
- 11 • `\+` `-`

增强运算符

1	<code>y += x</code>	等同于 <code>y = y + x</code>
2	<code>y -= x</code>	等同于 <code>y = y - x</code>
3	<code>y *= x</code>	等同于 <code>y = y * x</code>
4	<code>y /= x</code>	等同于 <code>y = y / x</code>
5	<code>y //= x</code>	等同于 <code>y = y // x</code>
6	<code>y %= x</code>	等同于 <code>y = y % x</code>
7	<code>y **= x</code>	等同于 <code>y = y ** x</code>

比较运算符

1	<code><</code>	小于
2	<code><=</code>	小于等于
3	<code>></code>	大于
4	<code>>=</code>	大于等于
5	<code>==</code>	等于
6	<code>!=</code>	不等于
7	返回布尔类型的值	
8	比较运算的数学表示方式: <code>0 <= x <= 100</code>	

逻辑运算符

```
1 ##### 与and
2 表示并且的关系，一假俱假。
3 示例：
4     True and True # True
5     True and False # False
6     False and True # False
7     False and False # False
8 ##### 或or
9 表示或者的关系，一真俱真
10 示例：
11    True or True      # True
12    True or False     # True
13    False or True     # True
14    False or False    # False
15
16 ##### 非not
17 表示取反
18 例如：
19    not True # 返回False
20    not False # 返回True
21
22 ##### 短路运算
23 一但结果确定，后面的语句将不再执行。
```

身份运算符

```
1  语法：
2      x is y
3      x is not y
4  作用：
5  is 用于判断两个对象是否是同一个对象，是时返回True，否则返回False。
6  is not 的作用与is相反
7  ### 优先级
8  • 高到低：
9  算数运算符
10 比较运算符
11 快捷运算符
12 身份运算符
13 逻辑运算符
```

语句

行

```
1  1. 物理行：程序员编写代码的行。
2  2. 逻辑行：python解释器需要执行的指令。
3  3. 建议一个逻辑行在一个物理行上。
4  4. 如果一个物理行中使用多个逻辑行，需要使用分号；隔开。
5  5. 如果逻辑行过长，可以使用隐式换行或显式换行。
6  隐式换行：所有括号的内容换行，称为隐式换行
7      括号包括：（） [] {} 三种
8  显式换行：通过折行符 \（反斜杠）换行，必须放在一行的末尾，目的是告诉解释器，下一行也是本行的语句。
```

pass 语句

```
1 通常用来填充语法空白。
```

选择语句

```
1  ### If  elif  else  语句
2  1. 作用：
3      让程序根据条件选择性的执行语句。
4  2. 语法：
5      if 条件1:
6      • 语句块1
7      elif 条件2:
8      • 语句块2
9      else:
10     • 语句块3
11  3. 说明：
12     elif 子句可以有0个或多个。
13     else 子句可以有0个或1个，且只能放在if语句的最后。
14  ### if 语句的真值表达式
15  if 100:
16  • print("真值")
17  等同于
18  if bool(100):
19      print("真值")
20  ### 条件表达式
```

```
21 | 语法: 变量 = 结果1 if 条件 else 结果2
22 | 作用: 根据条件(True/False) 来决定返回结果1还是结果2。
```

循环语句

```
1 | ### while语句
2 | 1. 作用:
3 | 可以让一段代码满足条件, 重复执行。
4 | 2. 语法:
5 | while 条件:
6 |     满足条件执行的语句
7 | else:
8 |     不满足条件执行的语句
9 | 3. 说明:
10 | else子句可以省略。
11 | 在循环体内用break终止循环时, else子句不执行。
```

for 语句

```
1 | 1. 作用:
2 | 用来遍历可迭代对象的数据元素。
3 | 可迭代对象是指能依次获取数据元素的对象, 例如: 容器类型。
4 | 2. 语法:
5 | for 变量列表 in 可迭代对象:
6 |     语句块1
7 | else:
8 |     语句块2
9 | 3. 说明:
10 | else子句可以省略。
11 | 在循环体内用break终止循环时, else子句不执行。
12 | ### range 函数
13 | 1. 作用:
14 | 用来创建一个生成一系列整数的可迭代对象(也叫整数序列生成器)。
15 | 2. 语法:
16 | range(开始点, 结束点, 间隔)
17 | 3. 说明:
18 | 函数返回的可迭代对象可以用for取出其中的元素
19 | 返回的数字不包含结束点
20 | 开始点默认为0
21 | 间隔默认值为1
```

跳转语句

```
1 | ### break 语句
2 | 1. 跳出循环体, 后面的代码不再执行。
3 | 2. 可以让while语句的else部分不执行。
4 | ### continue 语句
5 | 跳过本次, 继续下次循环。
```

容器类型

通用操作

```
1  ### 数学运算符
2  1. +: 用于拼接两个容器
3  2. +=: 用原容器与右侧容器拼接,并重新绑定变量
4  3. *: 重复生成容器元素
5  4. *=: 用原容器生成重复元素, 并重新绑定变量
6  5. < <= > >= == !=: 依次比较两个容器中元素,一但不同则返回比较结果。
7  ### 成员运算符
8  1. 语法:
9  • 数据 in 序列
10 数据 not in 序列
11 2. 作用:
12 • 如果在指定的序列中找到值, 返回bool类型。
13 ### 索引index
14 1. 作用: 访问容器元素
15 2. 语法: 容器[整数]
16 3. 说明:
17 正向索引从0开始, 第二个索引为1, 最后一个为len(s)-1。
18 反向索引从-1开始, -1代表最后一个, -2代表倒数第二个, 以此类推, 第一个是-len(s)。
19 ### 切片slice
20 1. 作用:
21 从容器中取出相应的元素重新组成一个容器。
22 2. 语法:
23 容器[(开始索引):(结束索引):(步长)]
24 3. 说明:
25 小括号()括起的部分代表可省略
26 结束索引不包含该位置元素
27 步长是切片每次获取完当前元素后移动的偏移量
28 ### 内建函数
29 1. len(x) 返回序列的长度
30 2. max(x) 返回序列的最大值元素
31 内置函数max对字典进行操作时, 使用key做参数, 字典的key类型要一致, 且能比较, 否则将报错。
32 3. min(x) 返回序列的最小值元素
33 4. sum(x) 返回序列中所有元素的和(元素必须是数值类型)
```

字符串 str

```
1  ### 定义
2  由一系列字符组成的不可变序列容器, 存储的是字符的编码值。
3  ### 编码
4  1. 字节byte: 计算机最小存储单位, 等于8 位bit.
5  2. 字符: 单个的数字, 文字与符号。
6  3. 字符集(码表): 存储字符与二进制序列的对应关系。
7  4. 编码: 将字符转换为对应的二进制序列的过程。
8  5. 解码: 将二进制序列转换为对应的字符的过程。
9  6. 编码方式:
10 --ASCII编码: 包含英文、数字等字符, 每个字符1个字节。
11 --GBK编码: 兼容ASCII编码, 包含21003个中文; 英文1个字节, 汉字2个字节。
12 --Unicode字符集: 国际统一编码, 旧字符集每个字符2字节, 新字符集4字节。
13 -- UTF-8编码: Unicode的存储与传输方式, 英文1字节, 中文3字节。
14 ##### 相关函数
15 1. ord(字符串): 返回该字符串的Unicode码。
16 2. chr(整数): 返回该整数对应的字符串。
17 ### 字面值
18 ##### 单引和双引号的区别
```

```

19 1. 单引号内的双引号不算结束符
20 2. 双引号内的单引号不算结束符
21 ##### 三引号作用
22 1. 换行会自动转换为换行符\n
23 2. 三引号内可以包含单引号和双引号
24 3. 作为文档字符串
25 ##### 转义字符
26 1. 改变字符的原始含义。
27 \ ' \ " \ \n \ \t \0 空字符
28 2. 原始字符串: 取消转义。
29 • a = r"C:\newfile\test.py"
30 ##### 字符串格式化
31 1. 定义:
32 生成一定格式的字符串。
33 2. 语法:
34 字符串%(变量)
35 "我的名字是%s,年龄是%s" % (name, age)
36 3. 类型码:
37 %s 字符串 %d整数 %f 浮点数

```

列表 list

```

1  ### 定义
2  由一系列变量组成的可变序列容器。
3  ### 基础操作
4  1. 创建列表:
5  列表名 = []
6  列表名 = list(可迭代对象)
7  2. 添加元素:
8  列表名.append(元素)
9  列表名.insert(索引, 元素)
10 3. 定位元素:
11 索引、切片
12 4. 遍历列表:
13 • 正向:
14 • for 变量名 in 列表名:
15 •     变量名就是元素
16 • 反向:
17 • for 索引名 in range(len(列表名)-1, -1, -1):
18 •     列表名[索引名]就是元素
19 5. 删除元素:
20 列表名.remove(元素)
21 del 列表名[索引或切片]
22 ### 深拷贝和浅拷贝
23 浅拷贝: 复制过程中, 只复制一层变量, 不会复制深层变量绑定的对象的复制过程。
24 深拷贝: 复制整个依赖的变量。
25
26 ### 列表VS字符串
27 1. 列表和字符串都是序列, 元素之间有先后顺序关系。
28 2. 字符串是不可变的序列, 列表是可变的序列。
29 3. 字符串中每个元素只能存储字符, 而列表可以存储任意类型。
30 4. 列表和字符串都是可迭代对象。
31 5. 函数:
32 将多个字符串拼接为一个。
33 result = "连接符".join(列表)
34 将一个字符串拆分为多个。
35 列表 = "a-b-c-d".split("分隔符")

```

```

36  ### 列表推导式
37  1. 定义：
38  使用简易方法，将可迭代对象转换为列表。
39  2. 语法：
40  变量 = [表达式 for 变量 in 可迭代对象]
41  变量 = [表达式 for 变量 in 可迭代对象 if 条件]
42  3. 说明：
43  如果if真值表达式的布尔值为False,则可迭代对象生成的数据将被丢弃。
44  ### 列表推导式嵌套
45  1. 语法：
46  变量 = [表达式 for 变量1 in 可迭代对象1 for 变量2 in 可迭代对象2]
47  2. 传统写法：
48  result = []
49  for r in ["a", "b", "c"]:
50      for c in ["A", "B", "C"]:
51          result.append(r + c)
52  3. 推导式写法：
53  result = [r + c for r in list01 for c in list02]

```

元组tuple

```

1  ### 定义
2  1. 由一系列变量组成的不可变序列容器。
3  2. 不可变是指一旦创建，不可以再添加/删除/修改元素。
4  ### 基础操作
5  1. 创建空元组：
6  元组名 = ()
7  元组名 = tuple()
8  2. 创建非空元组：
9  元组名 = (20,)
10  元组名 = (1, 2, 3)
11  元组名 = 100,200,300
12  元组名 = tuple(可迭代对象)
13  3. 获取元素：
14  索引、切片
15  4. 遍历元组：
16  • 正向：
17  • for 变量名 in 列表名：
18  •     变量名就是元素
19  • 反向：
20  • for 索引名 in range(len(列表名)-1,-1,-1):
21  •     元组名[索引名]就是元素
22  ### 作用
23  1. 元组与列表都可以存储一系列变量，由于列表会预留内存空间，所以可以增加元素。
24  2. 元组会按需分配内存，所以如果变量数量固定，建议使用元组，因为占用空间更小。
25  3. 应用：
26  变量交换的本质就是创建元组：x, y = y, x
27  格式化字符串的本质就是创建元组："姓名:%s, 年龄:%d" % ("tarena", 15)

```

字典dict

```

1  ### 定义
2  1. 由一系列键值对组成的可变映射容器。
3  2. 映射：一对一的对应关系，且每条记录无序。
4  3. 键必须唯一且不可变(字符串/数字/元组)，值没有限制。
5  ### 基础操作

```

```

6  1. 创建字典：
7  字典名 = {键1: 值1, 键2: 值2}
8  字典名 = dict (可迭代对象)
9  2. 添加/修改元素：
10 语法：
11     字典名[键] = 数据
12 说明：
13     键不存在，创建记录。
14     键存在，修改映射关系。
15 3. 获取元素：
16 变量 = 字典名[键] # 没有键则错误
17 4. 遍历字典：
18 • for 键名 in 字典名：
19 •     字典名[键名]
20 for 键名,值名 in 字典名.items():
21 语句
22 5. 删除元素：
23 del 字典名[键]
24 ### 字典推导式
25 1. 定义：
26 使用简易方法，将可迭代对象转换为字典。
27 2. 语法：
28 {键:值 for 变量 in 可迭代对象}
29 {键:值 for 变量 in 可迭代对象 if 条件}
30 ### 字典VS 列表
31 1. 都是可变容器。
32 2. 获取元素方式不同,列表用索引,字典用键。
33 3. 字典的插入,删除,修改的速度快于列表。
34 4. 列表的存储是有序的,字典的存储是无序的。

```

集合set

```

1  ### *****定义\*****
2  1. 由一系列不重复的不可变类型变量组成的可变映射容器。
3  2. 相当于只有键没有值的字典(键则是集合的数据)。
4  ### 基础操作
5  1. 创建空集合：
6  集合名 = set()
7  集合名 = set(可迭代对象)
8  2. 创建具有默认值集合：
9  集合名 = {1, 2, 3}
10 集合名 = set(可迭代对象)
11 3. 添加元素：
12 集合名.add(元素)
13 4. 删除元素：
14 集合名.discard(元素)
15 ### 运算
16 1. 交集&: 返回共同元素。
17 s1 = {1, 2, 3}
18 s2 = {2, 3, 4}
19 s3 = s1 & s2 # {2, 3}
20 2. 并集: 返回不重复元素
21 s1 = {1, 2, 3}
22 s2 = {2, 3, 4}
23 s3 = s1 | s2 # {1, 2, 3, 4}
24 3. 补集-: 返回只属于其中之一的元素
25 s1 = {1, 2, 3}

```

```

26 s2 = {2, 3, 4}
27 s1 - s2 # {1} 属于s1但不属于s2
28 补集^: 返回不同的元素
29     s1 = {1, 2, 3}
30     s2 = {2, 3, 4}
31     s3 = s1 ^ s2 # {1, 4} 等同于(s1-s2 | s2-s1)
32 4. 子集<: 判断一个集合的所有元素是否完全在另一个集合中
33 5. 超集>: 判断一个集合是否具有另一个集合的所有元素
34     s1 = {1, 2, 3}
35     s2 = {2, 3}
36 • s2 < s1 # True
37   s1 > s2 # True
38 6. 相同或不同== !=: 判断集合中的所有元素是否和另一个集合相同。
39     s1 = {1, 2, 3}
40     s2 = {3, 2, 1}
41     s1 == s2 # True
42     s1 != s2 # False
43 子集或相同,超集或相同 <= >=
44 ### 集合推导式
45 1. 定义:
46 使用简易方法,将可迭代对象转换为集合。
47 2. 语法:
48 {表达式 for 变量 in 可迭代对象}
49 {表达式 for 变量 in 可迭代对象 if 条件}

```

固定集合 frozenset

```

1 ### 定义
2 不可变的集合。
3 ##### 作用
4 固定集合可以作为字典的键,还可以作为集合的值。
5 ##### 基础操作
6 创建固定集合: frozenset(可迭代对象)
7 ##### 运算
8 等同于set

```

函数 function

```

1 # pycharm相关设置
2 1. “代码自动完成p”时间延时设置
3 File -> Settings -> Editor -> General -> Code Completion -> Autopopup in
  (ms):0
4 2. 快捷键:
5 Ctrl + P          参数信息 (在方法中调用参数)
6 Ctrl + Q          快速查看文档
7 Ctrl + Alt + M    提取方法

```

```

1 ## 定义
2 1. 用于封装一个特定的功能,表示一个功能或者行为。
3 2. 函数是可以重复执行的语句块,可以重复调用。
4 功能、参数、返回值
5 ## 作用
6 提高代码的可重用性和可维护性(代码层次结构更清晰)。
7 ## 定义函数
8 1. 语法:

```

```
9 def 函数名(形式参数):
10     函数体
11 2. 说明:
12 def 关键字: 全称是define, 意为"定义"。
13 函数名: 对函数体中语句的描述, 规则与变量名相同。
14 形式参数: 方法定义者要求调用者提供的信息。
15 函数体: 完成该功能的语句。
16 3. 函数的第一行语句建议使用文档字符串描述函数的功能与参数。
17 ## 调用函数
18 1. 语法: 函数名(实际参数)
19 2. 说明: 根据形参传递内容。
20 ## 返回值
21 1. 定义:
22 方法定义者告诉调用者的结果。
23 2. 语法:
24 return 数据
25 3. 说明:
26 return后没有语句, 相当于返回 None。
27 函数体没有return, 相当于返回None。
28 ## 可变 / 不可变类型在传参时的区别
29 1. 不可变类型参数有:
30 数值型(整数, 浮点数, 复数)
31 布尔值bool
32 None 空值
33 字符串str
34 元组tuple
35 固定集合frozenset
36 2. 可变类型参数有:
37 列表 list
38 字典 dict
39 集合 set
40 3. 传参说明:
41 不可变类型的数据传参时, 函数内部不会改变原数据的值。
42 可变类型的数据传参时, 函数内部可以改变原数据。
```

函数参数

```
1 ### 实参传递方式argument
2 ##### 位置传参
3 定义: 实参与形参的位置依次对应。
4 ##### 序列传参
5 定义: 实参用*将序列拆解后与形参的位置依次对应。
6 ##### 关键字传参
7 定义: 实参根据形参的名字进行对应。
8 ##### 字典关键字传参
9 1. 定义: 实参用**将字典拆解后与形参的名字进行对应。
10 2. 作用: 配合形参的缺省参数, 可以使调用者随意传参。
11 ### 形参定义方式parameter
12 ##### 缺省参数
13 1. 语法:
14 def 函数名(形参名1=默认实参1, 形参名2=默认实参2, ...):
15     函数体
16 2. 说明:
17 缺省参数必须自右至左依次存在, 如果一个参数有缺省参数, 则其右侧的所有参数都必须有缺省参数。
18 缺省参数可以有0个或多个, 甚至全部都有缺省参数。
19 ##### 位置形参
20 语法:
```

```

21 def 函数名(形参名1, 形参名2, ...):
22     函数体
23     ##### 星号元组形参
24     1. 语法:
25     def 函数名(*元组形参名):
26         函数体
27     2. 作用:
28     收集多余的位置传参。
29     3. 说明:
30     一般命名为'args'
31     形参列表中最多只能有一个
32     ##### 命名关键字形参
33     1. 语法:
34     def 函数名(*, 命名关键字形参1, 命名关键字形参2, ...):
35         函数体
36     def 函数名(*args, 命名关键字形参1, 命名关键字形参2, ...):
37         函数体
38     2. 作用:
39     强制实参使用关键字传参
40     ##### 双星号字典形参
41     1. 语法:
42     def 函数名(**字典形参名):
43         函数体
44     2. 作用:
45     收集多余的字典传参
46     3. 说明:
47     一般命名为'kwargs'
48     形参列表中最多只能有一个
49     ##### 参数自左至右的顺序
50     位置形参 --> 星号元组形参 --> 命名关键字形参 --> 双星号字典形参

```

作用域LEGB

```

1  1. 作用域: 变量起作用的范围。
2  2. Local局部作用域: 函数内部。
3  3. Enclosing 外部嵌套作用域 : 函数嵌套。
4  4. Global全局作用域: 模块(.py文件)内部。
5  5. Builtin内置模块作用域: builtins.py文件。
6  ## 变量名的查找规则
7  1. 由内到外: L -> E -> G -> B
8  2. 在访问变量时, 先查找本地变量, 然后是包裹此函数外部的函数内部的变量, 之后是全局变量, 最后是内置变量。
9  ## 局部变量
10 1. 定义在函数内部的变量(形参也是局部变量)
11 2. 只能在函数内部使用
12 3. 调用函数时才被创建, 函数结束后自动销毁
13 ## 全局变量
14 1. 定义在函数外部, 模块内部的变量。
15 2. 在整个模块(py文件)范围内访问(但函数内不能将其直接赋值)。
16 ## global 语句
17 1. 作用:
18 在函数内部修改全局变量。
19 在函数内部定义全局变量(全局声明)。
20 2. 语法:
21 global 变量1, 变量2, ...
22 3. 说明
23 在函数内直接为全局变量赋值, 视为创建新的局部变量。

```

```
24 不能先声明局部的变量，再用global声明为全局变量。
25 ## nonlocal 语句
26 1. 作用：
27 在内层函数修改外层嵌套函数内的变量
28 2. 语法
29 nonlocal 变量名1,变量名2, ...
30 3. 说明
31 在被嵌套的内函数中进行使用
```