**1.**

**# 常用的三种输出格式**

# 1. 带格式输出,{0}是指输出的第0个元素，同理{1}为第1个元素，{2}为第2个... 可以不按顺序排列

**print( "Hello {0}， I'm {2}, I,m {1} year old".format("world", age, name) )**

# 2. 使用类型输出，指定输出类型

**print( "I am %s, today is %d year"%(name, year) )**

# f字符串，{}中为元素，是.format的简化形式

**print( f"Today is {year}")**

Python print format() 格式化内置函数

<https://www.cnblogs.com/itfat/p/7455253.html>

**2.**

\_\_name\_\_ 是当前模块名，当模块被直接运行时模块名为 \_\_main\_\_ 。这句话的意思就是，当模块被直接运行时，以下代码块将被运行，当模块是被导入时，代码块不被运行。

<https://www.zhihu.com/question/49136398>

**3.**

Python编程中raise可以实现报出错误的功能，而报错的条件可以由程序员自己去定制。在面向对象编程中，可以先预留一个方法接口不实现，在其子类中实现。如果要求其子类一定要实现，不实现的时候会导致问题，那么采用raise的方式就很好。而此时产生的问题分类是**NotImplementedError**。

<https://www.runoob.com/python/python-exceptions.html>

我们可以使用raise语句自己触发异常

**raise语法**格式如下：

raise [Exception [, args [, traceback]]]

语句中 Exception 是异常的类型（例如，NameError）参数标准异常中任一种，args 是自已提供的异常参数。

最后一个参数是可选的（在实践中很少使用），如果存在，是跟踪异常对象。

**try-finally 语句**

try:

<语句>

finally:

<语句> #退出try时总会执行

raise

**4.**

虽然Python语法允许代码块**随意缩进几个空格**，但是位于同一个代码块中的所有语句必须**保持相同的缩进**，不能一下缩进3个空格，一下缩进4个空格。

**5.**

**\*args和\*\*kwargs**



args，kwargs仅仅是一个约定俗成的形参的写法

**args类型是一个元组tuple，而kwargs则是一个字典dict，并且args只能位于kwargs的前面。**

∗ 把函数接受到的多个参数打包成了元组，赋值给了形参。

∗的作用：函数接受实参时，按顺序分配给函数形参，如果遇到带∗的形参，那么就把还未分配出去的实参以元组形式打包（pack）,分配给那个带∗的形参。

∗这次没有用在函数定义中，而是用在了函数**调用**中。作用是

把实参（元组或列表），拆分（unpack）成单个的，依次赋值给函数的形参。

打包（pack）：\*args是把多个位置参数打包成元组，\*\*kwargs是把多个关键字参数打包成字典。

拆分（unpack）：\*args是把打包了的参数拆成单个的，依次赋值给函数的形参，\*\*kwargs是把字典的键值拆成单个的，依次赋值给函数的形参。

**6.**

TODO 在 python 中作为一种 助记符 (Mnemonics)，用来解释 将要做什么 。

**7.**

实例调用\_\_class\_\_属性时会指向该实例对应的类，然后可以再去调用其它类属性，类属性由类调用

**8.**

**python的@property**是python的一种装饰器，是用来修饰方法的。

<https://zhuanlan.zhihu.com/p/64487092>

作用：

我们可以使用@property装饰器来创建只读属性，@property装饰器会将方法转换为相同名称的只读属性,可以与所定义的属性配合使用，这样可以防止属性被修改。

**9.**

Python3 assert（断言）

Python assert（断言）用于判断一个表达式，在表达式条件为 false 的时候触发异常。

断言可以在条件不满足程序运行的情况下直接返回错误，而不必等待程序运行后出现崩溃的情况，例如我们的代码只能在 Linux 系统下运行，可以先判断当前系统是否符合条件。

**10.**

**python 类中变量方法命名时有关下划线的规则**

**1.单下划线：**按照Python中的习惯单下划线一般不应该被外界访问，尽管事实上是可以访问的。

**2.名字前面有双下划线：**Python类中以这种命名的变量或方法不能被外界访问。事实上在Python会把双下划线命名的变量改成另外的名字，因此在外部访问时会报找不到名称的错误

**3.名字前后都有双下划线**：这类名称是Python内置的变量名，本身和我们定义的名称没有什么区别，只是python为了加以区分。如构造函数\_\_init\_\_ 。这类变量或方法是可以访问的。

<https://blog.csdn.net/weixin_44339148/article/details/95216762>

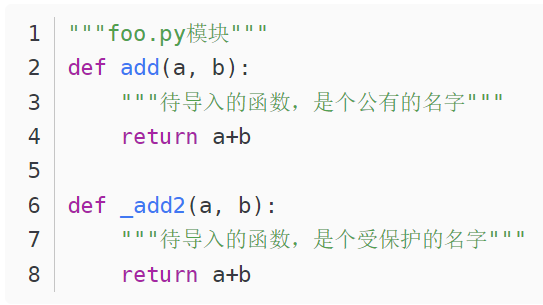
**Python带\_的变量或函数命名，带下划线的方法**

单前置下划线的名字:

《流畅的python》作者称之为“受保护的”名字，有两种主要的用法：

1、作为类名或函数名时，

会阻止其他python脚本通过【from module import \*】语句导入该名字，即该名字不会被星号匹配；



<https://blog.csdn.net/qq_35290785/article/details/93476722>

**11.**

**Python 实例方法、类方法、静态方法的区别与作用**

<https://www.cnblogs.com/wcwnina/p/8644892.html>

Python中至少有三种比较常见的方法类型，即实例方法，类方法、静态方法。它们是如何定义的呢？如何调用的呢？它们又有何区别和作用呢？且看下文。

首先，这三种方法都定义在类中。下面我先简单说一下怎么定义和调用的。（PS：实例对象的权限最大。）

实例方法

定义：第一个参数必须是实例对象，该参数名一般约定为“self”，通过它来传递实例的属性和方法（也可以传类的属性和方法）；

调用：只能由实例对象调用。

类方法

定义：使用装饰器@classmethod。第一个参数必须是当前类对象，该参数名一般约定为“cls”，通过它来传递类的属性和方法（不能传实例的属性和方法）；

调用：类和实例对象都可以调用。

静态方法

定义：使用装饰器@staticmethod。参数随意，没有“self”和“cls”参数，但是方法体中不能使用类或实例的任何属性和方法；

调用：类和实例对象都可以调用。

12.

**Python装饰器**

<https://www.zhihu.com/question/26930016>

**13.**

**占位符**

placeholder，占位符，在tensorflow中类似于函数参数，运行时必须传入值。函数说明：tf.placeholder(dtype, shape=None, name=None)

**14.**

**Python lambda介绍**

[**https://www.cnblogs.com/evening/archive/2012/03/29/2423554.html**](https://www.cnblogs.com/evening/archive/2012/03/29/2423554.html)

**15.**

**global语句是适用于当前整个代码块的声明。它是全局变量的标识符。如果某名字在局部名字空间中没有定义, 就自动使用相应的全局名字. 没有global是不可能手动指定一个名字是全局的.在 global 中出现的名字不能在global 之前的代码中使用.在 global 中出现的名字不能作为形参, 不能作为循环的控制对象, 不能在类定义, 函数定义, import语句中出现.**

**与nonlocal关键字的区别：**

**global语句用以知名某个特定的变量为全局作用域，并重新绑定它。nonlocal语句用以指明某个特定的变量为封闭作用域，并重新绑定它。**

**Python中定义函数时，若想在函数内部对函数外的变量进行操作，就需要在函数内部声明其为global**

**基本格式是：**

**关键字global，后跟一个或多个变量名**

**16.**

**python路径拼接os.path.join()函数的用法**

os.path.join()函数：连接两个或更多的路径名组件

1.如果各组件名首字母不包含’/’，则函数会自动加上

2.如果有一个组件是一个绝对路径，则在它之前的所有组件均会被舍弃

3.如果最后一个组件为空，则生成的路径以一个’/’分隔符结尾

<https://www.cnblogs.com/an-ning0920/p/10037790.html>

**os之遍历文件夹**

[**https://www.cnblogs.com/jim-fan/articles/13053252.html**](https://www.cnblogs.com/jim-fan/articles/13053252.html)

[**https://www.jianshu.com/p/582910d13501**](https://www.jianshu.com/p/582910d13501)

**17.**

**列表解析 List Comprehensions**

<http://blog.sciencenet.cn/blog-3031432-1060628.html>

<https://www.cnblogs.com/RayMin/p/5781725.html>

<https://www.runoob.com/note/42490>

**18.**

**parse\_args（argsparse）：python和命令行之间的交互**

<https://www.cnblogs.com/still-smile/p/11636958.html>

19.

**python--获取当前文件绝对路径的几种方式对比**

<https://www.cnblogs.com/xiaohuboke/p/13611234.html>