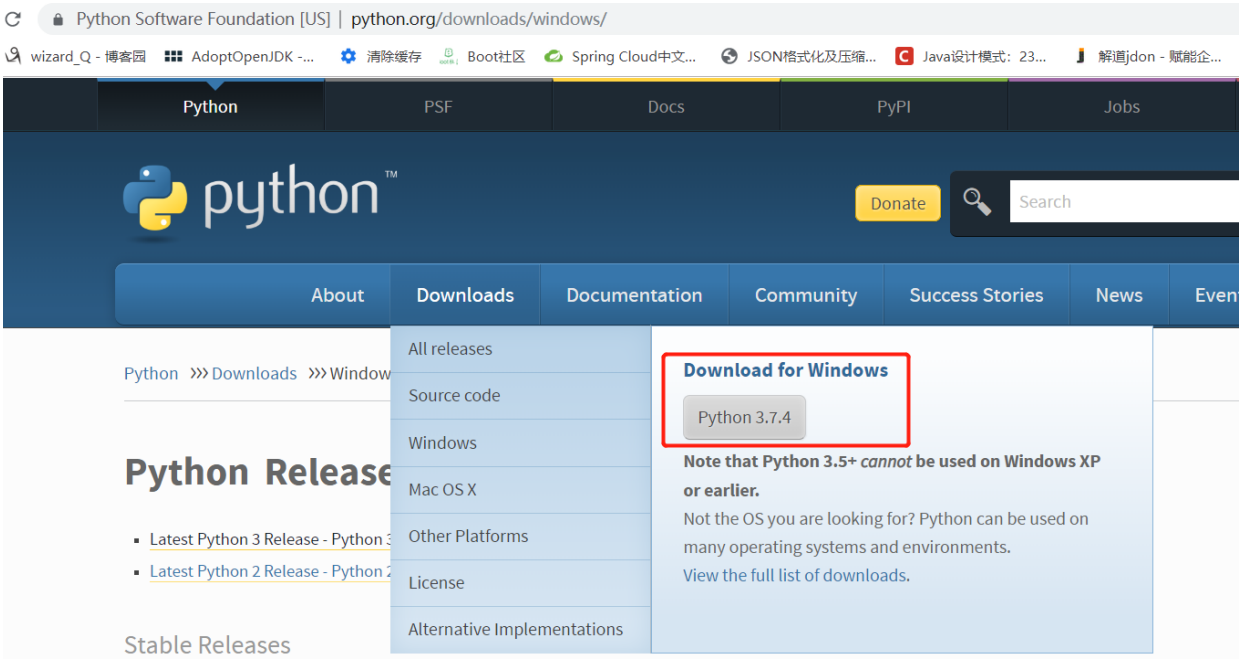
python入门及进阶学习记

官网 https://www.python.org/

下载:



cmd中查看版本及使用

```
C:\Users\ukyo>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
```

Python 教程

Python是一种易于学习又功能强大的编程语言,它提供了高效的高级数据结构,还有简单有效的面向对象编程.

Python优雅的语法和动态类型, 以及解释型语言的本质, 使它成为多数平台上写脚本和快速开发应用的理想语言.

Python解释器及丰富的标准库以源码或机器码的形式提供, 可以到Python官网免费获取适用于各个主要系统平台的版本, 并可自由地分发. 这个网站还包含许多免费第三方Python模块,程序和工具以及附加文档的发布页面或链接.

Python解释器易于扩展, 可以使用C或C++ (或其它可以通过C调用的语言) 扩展新的功能和数据类型. Python也可用于可定制化软件中的扩展程序语言.

这个教程非正式地介绍Python语言和系统的基本概念和功能, 最好在阅读的时候备一个Python解释器进行练习, 不过所有的例子都是相互独立的, suo

有关标准的对象和模块，参阅 Python 标准库。Python 语言参考 提供了更正式的语言参考。想要编写 C 或者 C++ 扩展可以参考 扩展和嵌入 Python 解释器 和 Python/C API 参考手册。也有不少书籍深入讲解Python 。

> 对python的了解程度

Python是一个简洁和强大的面向对象的编程语言.可以与Perl,Ruby,Scheme,Java比较.

Python的一些特点:

- 使用优雅的语法格式,让你的Python程序代码更加易读.
- 是一种易于使用的语言，使您的程序工作起来很简单。这使得Python在不损害可维护性的情况下，成为原型开发和其他特殊编程任务的理想选择。
- 附带一个大型标准库，支持许多常见的编程任务，如连接到web服务器、使用正则表达式搜索文本、读取和修改文件。
- Python的交互模式使测试短代码段变得很容易。还有一个捆绑的开发环境称为IDLE。
- 通过添加用编译语言(如C或c++)实现的新模块，可以很容易地进行扩展。
- 也可以嵌入到应用程序中提供可编程接口。
- 可以在任何地方运行，包括Mac OS X、Windows、Linux和Unix，Android和iOS也有非官方版本。
- 从两个意义上说，它是自由软件。下载或使用Python，或将其包含在应用程序中，都不需要任何成本。Python还可以自由修改和重新分发，因为虽然该语言是受版权保护的，但它是在开放源码许可下可用的。

Python的一些编程语言特性包括:

- 可以使用多种基本数据类型:数字(浮点数、复杂的和无限制长度的长整数)、字符串(ASCII和Unicode)、列表和字典。
- Python支持使用类和多重继承进行面向对象编程。
- 代码可以分为模块和包。
- 该语言支持引发和捕获异常，从而实现更清晰的错误处理。
- 数据类型是强类型和动态类型。混合不兼容的类型(例如试图添加字符串和数字)会引发异常，因此错误会更快被捕获。
- Python包含高级编程特性，如生成器和列表理解。
- Python的自动内存管理使您不必在代码中手动分配和释放内存。

查看Python程序示例代码:
请注意，这些示例是用python2编写的，可能需要一些调整才能在python3下运行。

1 line : Output 输出

```
print 'Hello, world!'
```

2 lines : Input 输入, assignment 赋值

```
name = raw_input('What is your name?\n')
print 'Hi, %s.' % name
```

↑详见2to3

> 术语对照表

交互式终端中默认的Python提示符. 往往会显示于能以交互方式在解释器里执行的样式代码之前.

交互式终端中输入特殊代码行时默认的Python提示符, 包括: 缩进的代码块, 成对的分隔符之内 (圆括号,方括号, 花括号或三重引号) , 或是指定一个装饰器之后.

2to3
一个将 Python 2.x 代码转换为 Python 3.x 代码的工具,能够处理大部分通过解析源码并遍历解析树可检测到的不兼容问题.

2to3 包含在标准库中, 模块名为 lib2to3; 并提供一个独立入口点 Tools/scripts/2to3 .
2to3 - 自动将 Python 2 代码转为 Python 3 代码

> 2to3 的使用

2to3 通常会作为脚本和Python解释器一起安装,你可以在Python根目录的Tools/scripts文件夹下找到它.
2to3 的基本调用参数是一个需要转换的文件或目录列表. 对于目录, 会递归地寻找其中的Python源码.
使用命令:

```
2to3 example.py
```

这个命令会打印出和源文件的区别, 通过传入-w参数, 2to3 也可以把需要的改写写回到原文件中 (除非传入了 -n 参数, 否则会为原始文件创建一个副本):

```
2to3 -w example.py
```

就拿示例代码中的来进行转换测试: https://wiki.python.org/moin/SimplePrograms
一般这个示例代码是Python2中的

```
name = raw_input('What is your name?\n')
print 'Hi, %s.' % name
```

保存文件后执行2to3命令(需要将2to3的工具放入系统中才能快速使用)
例如 : F:\python3.7\Tools\scripts
执行时添加 -w 参数

```
2to3 -w inputName.py
```

会产生一个原件副本,执行后原本名称的已经更改了:

```
name = input('What is your name?\n')
print('Hi, %s.' % name)
```

执行输出:

```
C:\Users\ukyo\Desktop\python-test\lesson1>python inputName.py
What is your name?
ukzq
Hi, ukzq.
```

3 lines : For loop (循环) , built-in enumerate function (内置枚举函数), new style formatting (新样式格式化)

```
friends = ['join','pat','gary','michael']
for i,name in enumerate(friends):
    print "iteration {iteration} is {name}".format(iteration=i,name=name)
```

2to3后:

```
friends = ['jack','kyo','ukyo','lucy']
for i,name in enumerate(friends):
    print("iteration {iteration} is {name}".format(iteration=i,name=nam
e))
```

之后的一直使用Python 3的示例:
4 lines : Fibonacci (斐波那契数列), tuple assignment (元组赋值)

```
parents, babies = (1,1)
while babies < 100:
    print('This generation has {0} babies'.format(babies))
    parents, babies = (babies, parents + babies)
```

5 lines : Functions (方法)

```
def greet(name):
    print('Hello',name)
greet('Jack')
greet('Lucy')
greet('alibaba')
```

6 lines : Import(导入依赖), regular expressions(正则表达式)

```
import re
```

```
for test_string in ['555-1212','ILL-EGAL']:
    if re.match(r'^\d{3}-\d{4}$',test_string):
        print(test_string, 'is a valid US local phone number')
    else:
        print(test_string, 'rejected')
```

7 lines: Dictionaries(字典), generator expressions (生成表达式)

```
prices = {'apple':0.40, 'banana':0.50}
my_purchase = {
    'apple':1,
    'banana':6}
grocery_bill = sum(prices[fruit] * my_purchase[fruit]
                   for fruit in my_purchase)
print('I owe the grocer $%.2f' % grocery_bill)
```

8 lines : Command line arguments(命令行参数) , exception handling(异常处理)

```
# This program adds up integers in the command line
import sys
try:
    total = sum(int(arg) for arg in sys.argv[1:])
    print('sum =', total)
except ValueError:
    print('Please supply integer arguments')
```

9 lines : Opening files (打开文件)

```
# indent your Python code to put into an email
import glob
# glob supports Unix style pathname extensions
python_files = glob.glob('*.py')
for file_name in sorted(python_files):
    print(' -------' + file_name)

    with open(file_name) as f:
        for line in f:
            print('     ' + line.rstrip())

    print()
```

10 lines : Time, conditionals, from…import, for…else

```

```

abstract base class – 抽象基类
抽象基类简称 ABC，是对 duck-typing 的补充，它提供了一种定义接口的新方式，相比之下其他技巧例如 hasattr() 显得过于笨拙或有微妙错误（例如使用 魔术方法）。ABC 引入了虚拟子类，这种类并非继承自其他类，但却仍能被 isinstance() 和 issubclass() 所认可；详见 abc 模块文档。
Python 自带许多内置的 ABC 用于实现数据结构（在 collections.abc 模块中）、数字（在 numbers 模块中）、流（在 io 模块中）、导入查找器和加载器（在 importlib.abc 模块中）。你可以使用 abc 模块来创建自己的 ABC。

annotation – 标注
关联到某个变量、类属性、函数形参或返回值的标签，被约定作为 type hint 来使用。
局部变量的标注在运行时不可访问，但全局变量、类属性和函数的标注会分别存放模块、类和函数的 **annotations** 特殊属性中。
参见 variable annotation、function annotation、PEP 484 和 PEP 526，对此功能均有介绍。

argument – 参数
在调用函数时传给 function (或method) 的值, 参数分为两种:

- 关键字参数: 在函数调用中前面带有标识符 (例如 name=) 或者作为包含在前面带有 ** 的字典里的值传入. 举例来说, 3和5在以下对 complex() 的调用中均属于关键字参数:

```
complex(real=3, imag=5)
complex(**{'real':3,'imag':5})
```

- 位置参数: 不属于关键字参数的参数. 位置参数可出现于参数列表的开头以及/或者作为前面带有 * 的iterable里的元素被传入. 举例来说, 3和5在以下调用中均属于位置参数:

```
complex(3,5)
complex(*(3,5))
```

参数会被赋值给函数体中对应的局部变量。有关赋值规则参见 调用 一节。根据语法，任何表达式都可用来表示一个参数；最终算出的值会被赋给对应的局部变量。
另参见 parameter 术语表条目，常见问题中 参数与形参的区别 以及 PEP 362。