# Python Cheat Sheet-7. 模块与包（module and package），及PyPI发布（distribution）

?> PCS-7（    ）：模块与包

PCS7之前必读参考：[29. Packages（python-course.eu）](https://python-course.eu/python-tutorial/packages.php)①

知识点

描述

代码段

运算结果

备注

一个单独的.py文件即为一个模块（module），包含定义的函数、类或变量；当模块的数量不断增加，最好将其分类于不同的文件夹中，这种有效组织模块的方式就是创建一个包（package），包通常为一个含有\_\_init\_\_.py文件的文件夹，可以包含子模块或子包(子目录)。通常把一个项目组织为一个包，然后可以使用[setuptools](https://pypi.org/project/setuptools/)②工具创建和分发包，从而像标准库或扩展库一样安装包和调用包，使用包内的工具（即定义的函数或类等）。

下述为一个包的文件夹结构。整个项目位于PCS\_7\_package\_example文件夹下，其中包放置于src文件夹下，其中toolkit4beginner文件夹是创建的包，包括两个子包graph和utility文件夹，其中graph子包组织图表绘制类的工具，例如自定义样式的打印箱型图和四象限图（4 quadrant diagram），包含在graph.py文件中；utility子包组织通用的工具，例如展平列表，计算时间长度等都位于general\_tools.py文件中，而描述性统计和获取最近值的函数都位于stats\_tools.py文件中。test文件夹包含用于测试包的模块，确保运行正确，如果出现错误，则需要进行调试。

在包中，都有一个\_\_init\_\_.py文件，确保包能够正确导入及用setuptools打包时，可以找到包的位置，该文件通常为空，也可以像常规的模块写入代码，代码会在python第1次导入一个目录时自动运行，因此主要作为执行（软件）包所需初始化步骤的钩子（hook），控制定义包级别的变量。因为python文件都是按目录（子包）组织模块，python会通过搜索该目录下的文件导入相关模块。而不是搜索所有目录，只搜索目录下包含有\_\_init\_\_.py文件的目录，此时这个目录当作一个包目录，进而搜索包内的模块。如果未含有\_\_init\_\_.py文件，调用该目录下的模块时会引发未找到等错误提示。

如果在python 3.3及以上的版本，如果目录中不含有\_\_init\_\_.py文件，也可以作为单目录命名空间包实现（single-directory namespace packages），正确导入模块。但是也意味着，没有自定义初始化运行时加载控制变量的代码。

PCS\_7\_package\_example/  
├── docs/  
├── pyproject.toml  
├── README.md  
├── setup.py  
├── src/  
│ └── toolkit4beginner/  
│ ├── \_\_init\_\_.py  
│ ├── graph/  
│ │ ├── \_\_init\_\_.py  
│ │ └── graph.py  
│ └── utility/  
│ ├── \_\_init\_\_.py  
│ ├── general\_tools.py  
│ └── stats\_tools.py  
└── tests/  
 ├── \_\_init\_\_.py  
 ├── displayablePath.py  
 ├── pypi\_published\_test.py  
 └── test.py

上述文件夹目录结构中，setup.py文件是setuptools创建和分发包的配置文件；README.md文件是在setup.py文件中读取，用于配置long\_description参数，方便编辑；pyproject.toml文件是从[PEP 621](https://peps.python.org/pep-0621/)③开始，Python 社区选择 pyproject.toml 作为指定项目元数据的标准方式。 Setuptools 采用了此标准，使用此文件中包含的信息作为构建过程的输入（与setup.py作用同，但格式不同，此包中可以移除该文件）。创建分发包有多种打包方式，例如在anaconda下所建立环境的终端（terminal）执行python setup.py sdist，为打包成源码包（即压缩包，.zip，.tar，.gz等，如果不指定参数--formats=gztar，按当前平台默认格式; 执行python setup.py bdist\_wheel， 为打包成二进制包（需要先安装[wheel](https://pypi.org/project/wheel/)④）。打包完成后，会在文件夹下生成相应的文件，目录结构变化为：

PCS\_7\_package\_example/  
├── build/  
│ ├── bdist.win-amd64/  
│ └── lib/  
│ └── toolkit4beginner/  
│ ├── \_\_init\_\_.py  
│ ├── graph/  
│ │ ├── \_\_init\_\_.py  
│ │ └── graph.py  
│ └── utility/  
│ ├── \_\_init\_\_.py  
│ ├── general\_tools.py  
│ └── stats\_tools.py  
├── dist/  
│ ├── toolkit4beginner-0.3.3-py3-none-any.whl  
│ └── toolkit4beginner-0.3.3.tar.gz  
├── docs/  
├── pyproject.toml  
├── README.md  
├── setup.py  
├── src/  
│ ├── toolkit4beginner/  
│ │ ├── \_\_init\_\_.py  
│ │ ├── graph/  
│ │ │ ├── \_\_init\_\_.py  
│ │ │ └── graph.py  
│ │ └── utility/  
│ │ ├── \_\_init\_\_.py  
│ │ ├── general\_tools.py  
│ │ └── stats\_tools.py  
│ └── toolkit4beginner.egg-info/  
│ ├── dependency\_links.txt  
│ ├── PKG-INFO  
│ ├── requires.txt  
│ ├── SOURCES.txt  
│ └── top\_level.txt  
└── tests/  
 ├── \_\_init\_\_.py  
 ├── displayablePath.py  
 ├── pypi\_published\_test.py  
 └── test.py

打包的二进制文件或源码文件为新生成的dist文件夹下的toolkit4beginner-0.3.3-py3-none-any.whl和toolkit4beginner-0.3.3.tar.gz。安装分发包同一般库的安装，可以在终端定位到该文件夹后，直接执行pip install XXX.whl或源码包执行pip install xxx.tar.gz，完成安装。如果要在[PyPI](https://pypi.org/)⑤上发布，可以提供给更多人使用，则可以注册该网站，将生成的源码或者二进制文件通过twine upload dist/...方式推送至自己的注册地址（需要安装[twine](https://pypi.org/project/twine/)⑥工具，推送时提示输入pypi的注册用户名和密码）。上传成成功后（pypi可能不支持代理上传），则可以在pypi网站注册地址的your projects下找到发布的包（因为推送的包不能有重名，如果名字已存在，需要修改后，重新打包推送），在[view](https://pypi.org/project/toolkit4beginner/0.3.3/)⑦下查看安装代码为pip install toolkit4beginner==0.3.3（或存在之前版本，则可以用pip install toolkit4beginner -U方法更新）。 此时该包同任何第三方库一样，可以直接在终端根目录安装，并且不需要源文件，可以为任何人安装使用。

建立一个项目，如果希望最终发布为一个分发包提供给更多人使用，则在建立项目的时候最好就按照包的文件目录结构布局，根据项目要求初步构思好基本的分类，然后在书写代码时，将函数、类和变量按类置于不同的模块下，并将模块置于对应的分类目录下，如果开始粗略的分类不满足要求，则实时调整目录结构，只要保持一个包合理的目录结构就可。最终配置setup.py文件，打包推送发布。

本节的案例是将之前PCS1-6中部分函数分类置于不同的模块和目录下，并增加了一个新的函数boxplot\_custom(data\_dict,\*\*args)位于graph/graph.py模块下。

每一模块都使用if \_\_name\_\_=="\_\_main\_\_":语句进行模块内的测试。如果构建包，在直接使用模块做数据分析时，通常也将函数、类和全局性的变量置于该语句之前，而在该语句之下（缩进），执行调用计算的代码，该语句的目的是避免其它模块调用该模块内的工具时，执行了测试或者用于直接计算部分的代码。

* toolkit/setup.py

该配置文件不在交互式解释器中运行，需要在终端中定位到该文件目录后，执行python setup.py sdist或python setup.py bdist\_wheel，完成打包。通常确定包名name之后，每次模块发生更新时，修改version参数后，重新打包上传至pypi。pypi会将包置于同一个项目下，其下包含每次包更新的版本。参数配置时主要包括两大部分，一部分是项目(版权、管理、版本)信息，例如name, version,license, author,author\_email,description,long\_description,long\_description\_content\_type和url等；另一部分是打包要求、依赖和包位置等胚子信息，定位包所在目录的packages参数，对python版本有要求的python\_requires参数，安装依赖库的 install\_requires参数，是否包含额外文件的include\_package\_data参数等。更多参数可以查看[setuptools documentation](https://setuptools.pypa.io/en/latest/)⑧，其中包括[pyproject.toml](https://setuptools.pypa.io/en/latest/userguide/pyproject_config.html)⑨配置说明。

# -\*- coding: utf-8 -\*-  
"""  
Created on Sun Aug 21 14:57:02 2022  
  
@author: Richie Bao-caDesign设计(cadesign.cn)  
"""  
from setuptools import setup,find\_packages,find\_namespace\_packages  
  
with open("README.md", "r", encoding="utf-8") as fh:  
 long\_description = fh.read()  
  
setup(name='toolkit4beginner', #应用名，即包名  
 version='0.3.3', #版本号  
 license="MIT", #版权声明，BSD,MIT  
 author='Richie Bao-caDesign设计(cadesign.cn)', #作者名  
 author\_email='richiebao@outlook.com', #作者邮箱  
 description='模块、包和分发文件目录组织结构说明', #描述  
 long\_description=long\_description,  
 long\_description\_content\_type="text/markdown",  
 url='https://richiebao.github.io/USDA\_CH\_final', #项目主页   
 #----------------------------------------------------------------   
 package\_dir={"": "src"},  
 packages=find\_packages(where='src'),#包括安装包内的python包；find\_namespace\_packages()，和find\_packages() ['toolkit4beginner']  
 python\_requires='>=3.6', #pyton版本控制  
 platforms='any',  
 install\_requires=['matplotlib','statistics','numpy'] #自动安装依赖包（库）  
 # include\_package\_data=True, #如果配置有MANIFEST.in，包含数据文件或额外其它文，该参数配置为True，则一同打包  
 )

* toolkit4beginner/graph/graph.py

自定义样式图表打印模块。

# -\*- coding: utf-8 -\*-  
"""  
Created on Sun Aug 21 15:08:21 2022  
  
@author: Richie Bao-caDesign设计(cadesign.cn)  
"""  
def boxplot\_custom(data\_dict,\*\*args):  
 '''  
 根据matplotlib库的箱型图打印方法，自定义箱型图可调整的打印样式。   
  
 Parameters  
 ----------  
 data\_dict : dict(list,numerical)  
 字典结构形式的数据，键为横坐分类数据，值为数值列表.  
 \*\*args : keyword arguments  
 可调整的箱型图样式参数包括['figsize', 'fontsize', 'frameOn', 'xlabel', 'ylabel', 'labelsize', 'tick\_length', 'tick\_width', 'tick\_color', 'tick\_direction', 'notch', 'sym', 'whisker\_linestyle', 'whisker\_linewidth', 'median\_linewidth', 'median\_capstyle'].  
  
 Returns  
 -------  
 paras : dict  
 样式更新后的参数值.  
  
 '''  
 import matplotlib.pyplot as plt  
   
 #计算值提取  
 data\_keys=list(data\_dict.keys())  
 data\_values=list(data\_dict.values())   
   
 #配置与更新参数  
 paras={'figsize':(10,10),  
 'fontsize':15,  
 'frameOn':['top','right','bottom','left'],  
 'xlabel':None,  
 'ylabel':None,  
 'labelsize':15,  
 'tick\_length':7,  
 'tick\_width':3,  
 'tick\_color':'b',  
 'tick\_direction':'in',  
 'notch':0,  
 'sym':'b+',  
 'whisker\_linestyle':None,  
 'whisker\_linewidth':None,  
 'median\_linewidth':None,  
 'median\_capstyle':'butt'}  
   
 # print(paras)  
 paras.update(args)  
 # print(paras)  
   
 #根据参数调整打印图表样式  
 plt.rcParams.update({'font.size': paras['fontsize']})  
 frameOff=set(['top','right','bottom','left'])-set(paras['frameOn'])  
   
   
 #图表打印  
 fig, ax=plt.subplots(figsize=paras['figsize'])  
 ax.boxplot(data\_values,  
 notch=paras['notch'],  
 sym=paras['sym'],  
 whiskerprops=dict(linestyle=paras['whisker\_linestyle'],linewidth=paras['whisker\_linewidth']),  
 medianprops={"linewidth": paras['median\_linewidth'],"solid\_capstyle": paras['median\_capstyle']})  
   
 ax.set\_xticklabels(data\_keys) #配置X轴刻度标签  
 for f in frameOff:  
 ax.spines[f].set\_visible(False) #配置边框是否显示  
   
 #配置X和Y轴标签  
 ax.set\_xlabel(paras['xlabel'])  
 ax.set\_ylabel(paras['ylabel'])  
   
 #配置X和Y轴标签字体大小  
 ax.xaxis.label.set\_size(paras['labelsize'])  
 ax.yaxis.label.set\_size(paras['labelsize'])  
   
 #配置轴刻度样式  
 ax.tick\_params(length=paras['tick\_length'],  
 width=paras['tick\_width'],  
 color=paras['tick\_color'],  
 direction=paras['tick\_direction'])  
  
 plt.show()   
 return paras  
  
#%%  
def four\_quadrant\_diagram(data\_dict,method='mean',\*\*args):  
 '''  
 绘制四象限图。  
  
 Parameters  
 ----------  
 data\_dict : dict(list)  
 字典格式数据，包括两组键值.其中键名将用作轴标签  
 method : string, optional  
 按照均值（mean）或中位数(median)方式划分四象限. The default is 'mean'.  
 \*\*args : keyword arguments  
 可调整的图表样式参数包括：['figsize', 'fontsize', 'frameOn', 'crosshair\_color', 'crosshair\_linstyle', 'crosshair\_linewidth', 'labelsize', 'tick\_length', 'tick\_width', 'tick\_color', 'tick\_direction', 'dot\_color', 'dot\_size', 'annotation\_position\_finetune', 'annotation\_fontsize', 'annotation\_lable', 'annotation\_color'].  
  
 Returns  
 -------  
 None.  
  
 '''  
 import matplotlib as mpl   
 import matplotlib.pyplot as plt  
 from statistics import mean,median  
   
 #解决中文乱字符的问题  
 mpl.rcParams['font.sans-serif']=['SimHei']  
 mpl.rcParams['font.serif']=['SimHei']   
   
 #计算值提取  
 (key\_x,x),(key\_y,y)=data\_dict.items()   
  
 #配置与更新参数  
 paras={'figsize':(10,10),  
 'fontsize':15,  
 'frameOn':['top','right','bottom','left'],  
 'crosshair\_color':'red',  
 'crosshair\_linstyle':'--',  
 'crosshair\_linewidth':3,  
 'labelsize':15,  
 'tick\_length':7,  
 'tick\_width':3,  
 'tick\_color':'b',  
 'tick\_direction':'in',  
 'dot\_color':'k',  
 'dot\_size':50,  
 'annotation\_position\_finetune':0.5,  
 'annotation\_fontsize':10,  
 'annotation\_lable':None,  
 'annotation\_color':'gray',  
 }   
 paras.update(args)  
   
 #根据参数调整打印图表样式  
 plt.rcParams.update({'font.size': paras['fontsize']})  
 frameOff=set(['top','right','bottom','left'])-set(paras['frameOn'])   
   
 #图表打印  
 fig, ax=plt.subplots(figsize=paras['figsize'])   
 ax.scatter(x,y,c=paras['dot\_color'],s=paras['dot\_size'])  
   
 crosshair\_color=paras['crosshair\_color']  
 crosshair\_linestyle=paras['crosshair\_linstyle']  
 crosshair\_linewidth=paras['crosshair\_linewidth']  
 if method=='mean':  
 ax.axhline(y=mean(y), color=crosshair\_color, linestyle=crosshair\_linestyle, linewidth=crosshair\_linewidth)   
 ax.axvline(x=mean(x), color=crosshair\_color, linestyle=crosshair\_linestyle, linewidth=crosshair\_linewidth)   
 elif method=='median':  
 plt.axhline(y=median(y), color=crosshair\_color, linestyle=crosshair\_linestyle, linewidth=crosshair\_linewidth)   
 plt.axvline(x=median(x), color=crosshair\_color, linestyle=crosshair\_linestyle, linewidth=crosshair\_linewidth)   
  
 for f in frameOff:  
 ax.spines[f].set\_visible(False) #配置边框是否显示   
   
 #标注点   
 if paras['annotation\_lable']:  
 annotations=paras['annotation\_lable']  
 else:  
 annotations=list(range(len(x)))  
   
 annotation\_position\_finetune=paras['annotation\_position\_finetune']  
 for label,x,y in zip(annotations,x,y):  
 ax.annotate(label,(x+annotation\_position\_finetune,y+annotation\_position\_finetune),  
 fontsize=paras['annotation\_fontsize'],  
 color=paras['annotation\_color'])  
   
 #配置X和Y轴标签  
 ax.set\_xlabel(key\_x)  
 ax.set\_ylabel(key\_y)   
  
 #配置X和Y轴标签字体大小  
 ax.xaxis.label.set\_size(paras['labelsize'])  
 ax.yaxis.label.set\_size(paras['labelsize'])  
   
 #配置轴刻度样式  
 ax.tick\_params(length=paras['tick\_length'],  
 width=paras['tick\_width'],  
 color=paras['tick\_color'],  
 direction=paras['tick\_direction'])   
   
 plt.tight\_layout()  
 plt.show()   
  
# four\_quadrant\_diagram(test\_EC\_scores,  
# # method='median',  
# figsize=(15,15),  
# fontsize=23,  
# frameOn=['bottom','left'],  
# labelsize='30',  
# dot\_size=200,  
# annotation\_fontsize=30,  
# annotation\_lable=test\_names,  
# )   
#%%   
  
if \_\_name\_\_=="\_\_main\_\_":  
 #模块内测试  
 #A\_测试-boxplot\_custom(data\_dict,\*\*args)  
 #%%  
 test\_score\_lst\_dic={'English': [90, 81, 73, 97, 85, 60, 74, 64, 72, 67, 87, 78, 85, 96, 77, 100, 92, 86],   
 'Chinese': [71, 90, 79, 70, 67, 66, 60, 83, 57, 85, 93, 89, 78, 74, 65, 78, 53, 80],   
 'history': [73, 61, 74, 47, 49, 87, 69, 65, 36, 7, 53, 100, 57, 45, 56, 34, 37, 70],   
 'biology': [59, 73, 47, 38, 63, 56, 75, 53, 80, 50, 41, 62, 44, 26, 91, 35, 53, 68]}  
 \_=boxplot\_custom(test\_score\_lst\_dic,  
 figsize=(15,10),  
 fontsize=23,  
 frameOn=['bottom','left'],  
 xlabel='subject',  
 ylabel='score',  
 labelsize='30',  
 tick\_color='r',  
 notch=1,  
 sym='rs',  
 whisker\_linestyle='--',  
 whisker\_linewidth=5,  
 median\_linewidth=5  
 )  
 #%%  
 #B\_测试-four\_quadrant\_diagram(data\_dict,method='mean',\*\*args)  
 test\_EC\_scores={'English': [90, 81, 73, 97, 85, 60, 74, 64, 72, 67, 87, 78, 85, 96, 77, 100, 92, 86],   
 'Chinese': [71, 90, 79, 70, 67, 66, 60, 83, 57, 85, 93, 89, 78, 74, 65, 78, 53, 80], }  
 test\_names=['Mason', 'Reece', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P']  
 four\_quadrant\_diagram(test\_EC\_scores,  
 # method='median',  
 figsize=(15,15),  
 fontsize=23,  
 frameOn=['bottom','left'],  
 labelsize='30',  
 dot\_size=200,  
 annotation\_fontsize=30,  
 annotation\_lable=test\_names,  
 )   
 #%%

* toolkit4beginner/utility/general\_tools.py

一般常用工具模块。

# -\*- coding: utf-8 -\*-  
"""  
Created on Sun Aug 21 18:35:24 2022  
  
@author: Richie Bao-caDesign设计(cadesign.cn)  
"""  
#展平嵌套列表，返回列表  
flatten\_lst=lambda lst: [m for n\_lst in lst for m in flatten\_lst(n\_lst)] if type(lst) is list else [lst]  
  
def flatten\_lst\_generator(lst):  
 '''  
 展平嵌套列表，返回迭代对象。  
  
 Parameters  
 ----------  
 lst : list  
 嵌套列表.  
  
 Yields  
 ------  
 iterable object  
 嵌套列表展平后的迭代对象.  
  
 '''  
 try: #使用语句try/except捕捉异常  
 for n\_lst in lst:   
 for m in flatten\_lst\_generator(n\_lst):  
 yield m  
   
 except:   
 yield lst  
  
def infinite\_generator():  
 '''  
 无穷整数生成器。  
  
 Yields  
 ------  
 i : int  
 整数值.  
  
 '''  
 i=0  
 while True:  
 i+=1  
 yield i  
   
def recursive\_factorial(n):  
 '''  
 阶乘计算。  
  
 Parameters  
 ----------  
 n : int  
 阶乘计算的末尾值.  
  
 Returns  
 -------  
 int  
 阶乘计算结果.  
  
 '''  
 if n==1:  
 return 1  
 else:  
 return n\*recursive\_factorial(n-1)  
  
def start\_time():  
 '''  
 获取当前时间  
  
 Returns  
 -------  
 start\_time : datetime  
 返回当前时间.  
  
 '''  
 import datetime  
   
 start\_time=datetime.datetime.now()  
 print("start time:",start\_time)  
 return start\_time  
  
def duration(start\_time):  
 '''  
 配合start\_time()使用。计算时间长度。  
  
 Parameters  
 ----------  
 start\_time : datetime  
 用于计算时间长度的开会时间.  
  
 Returns  
 -------  
 None.  
  
 '''  
 import datetime  
   
 end\_time=datetime.datetime.now()  
 print("end time:",end\_time)  
 duration=(end\_time-start\_time).seconds/60  
 print("Total time spend:%.2f minutes"%duration)   
  
if \_\_name\_\_=="\_\_main\_\_":  
 #模块内测试  
 #A\_测试-flatten\_lst  
 #%%  
 nested\_lst=[['A','B',['C','D'],'E'],[6,[7,8,[9]]]]   
 print(flatten\_lst(nested\_lst))   
 #%%  
 #B\_测试-flatten\_lst\_generator(lst)  
 fl=flatten\_lst\_generator(nested\_lst)  
 print(list(fl))  
 #%%  
 #C\_测试-infinite\_generator()  
 inf=infinite\_generator()  
 print(next(inf))   
 print(next(inf))   
 #%%  
 #D\_测试-recursive\_factorial(n)  
 print(recursive\_factorial(6))  
 #%%  
 #E\_测试-start\_time();duration(start\_time)  
 s\_t=start\_time()  
 for i in range(10\*\*8):value=i  
 duration(s\_t)   
 #%%

['A', 'B', 'C', 'D', 'E', 6, 7, 8, 9]  
['A', 'B', 'C', 'D', 'E', 6, 7, 8, 9]  
1  
2  
720  
start time: 2022-08-21 23:16:51.631636  
end time: 2022-08-21 23:17:04.243942  
Total time spend:0.20 minutes

* toolkit4beginner/utility/stats\_tools.py

统计类工具模块。

# -\*- coding: utf-8 -\*-  
"""  
Created on Sun Aug 21 18:56:45 2022  
  
@author: Richie Bao-caDesign设计(cadesign.cn)  
"""  
  
def descriptive\_statistics(data,measure=None,decimals=2):  
 '''  
 计算给定数值列表的描述性统计值，包括数量、均值、标准差、方差、中位数、众数、最小值和最大值。  
   
   
 Parameters  
 ----------  
 data : list(numerical)  
 待统计的数值列表.  
 measure : str, optional  
 包括：'count', 'mean', 'std', 'variance', 'median', 'mode', 'min', 'max'. The default is None.  
 decimals : int, optional  
 小数位数. The default is 2.  
  
 Returns  
 -------  
 dict  
 如果不给定参数measure，则以字典形式返回所有值；否则返回给定measure对应值的表述字符串.  
  
 '''  
 import statistics  
   
 d\_s={  
 'count':len(data), #样本数  
 'mean':round(statistics.mean(data),decimals), #均值  
 'std':round(statistics.stdev(data),decimals), #标准差  
 'variance': round(statistics.variance(data),decimals), #方差   
 'median':statistics.median(data), #中位数  
 'mode':statistics.mode(data), #众数  
 'min':min(data), #最小值  
 'max':max(data), #最大值   
 }  
   
 if measure:  
 return '{}={}'.format(measure,d\_s[measure])  
 else:  
 return d\_s  
   
def value2values\_comparison(x,lst):  
 '''  
 判断给定的一个值是否在一个列表中，如果在，则返回给定值；如果不在，则返回列表中与其差值绝对值最小的值。  
  
 Parameters  
 ----------  
 x : numerical  
 数值.  
 lst : list(numerical)  
 用于寻找给定值的参考列表.  
  
 Returns  
 -------  
 numerical  
 列表中最接近给定值的值.  
  
 '''  
 import numpy as np  
   
 lst\_mean=np.mean(lst)  
 abs\_difference\_func=lambda value:abs(x)  
 comparisonOF2values=lambda v1,v2:print('x is higher than the average %s of the list.'%lst\_mean) if v1>v2 else print('x is lower than the average %s of the list.'%lst\_mean)   
   
 if x in lst:  
 print("%s in the given list."%x)  
 comparisonOF2values(x,lst\_mean)   
 return x  
 else:  
 print('%.3f is not in the list.'%x)  
 closest\_value=min(lst,key=abs\_difference\_func)  
 print('the nearest value to %s is %s.'%(x,closest\_value))  
 print("\_"\*50)  
 comparisonOF2values(closest\_value,lst\_mean)   
 return closest\_value  
   
   
if \_\_name\_\_=='\_\_main\_\_':   
 #%%  
 #A\_测试-descriptive\_statistics(data,measure=None,decimals=2)  
 ranmen\_price\_lst=[700,850,600,650,980,750,500,890,880,  
 700,890,720,680,650,790,670,680,900,  
 880,720,850,700,780,850,750,780,590,  
 650,580,750,800,550,750,700,600,800,  
 800,880,790,790,780,600,690,680,650,  
 890,930,650,777,700]  
 d\_s\_1=descriptive\_statistics(ranmen\_price\_lst)  
 print(d\_s\_1)  
 print('--'\*30)  
 d\_s\_2=descriptive\_statistics(ranmen\_price\_lst,'std')  
 print(d\_s\_2)  
 d\_s\_3=descriptive\_statistics(ranmen\_price\_lst,measure='mean',decimals=1)  
 print(d\_s\_3)  
 #%%  
 #B\_测试-value2values\_comparison(x,lst)  
 ranmen\_price\_lst=[700,850,600,650,980,750,500,890,880,700,890,720,680,650,790,670,680,900,880,720,850,700,780,850,750,  
 80,590,650,580,750,800,550,750,700,600,800,800,880,790,790,780,600,690,680,650,890,930,650,777,700]  
 price\_x=200.68   
 price\_x\_closestValue=value2values\_comparison(price\_x,ranmen\_price\_lst)   
 print(price\_x\_closestValue)  
 #%%

{'count': 50, 'mean': 743.34, 'std': 108.26, 'variance': 11720.64, 'median': 750.0, 'mode': 700, 'min': 500, 'max': 980}  
------------------------------------------------------------  
std=108.26  
mean=743.3  
200.680 is not in the list.  
the nearest value to 200.68 is 700.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
x is lower than the average 729.34 of the list.  
700

* toolkit4beginner/test/test.py

如果打包推送至PyPI上发布，并通过pip install toolkit4beginner==0.3.3或pip install toolkit4beginner -U安装，则可以使用该包的工具。首先是通过一行调入语句import toolkit4beginner简单的测试，是否包已经正确安装，如果没有提示错误，则已正确安装并调入，否则需要确认之前大安装步骤是否正确，或者源码和打包的整个流程中是否有疏漏，从新安装或者调试后重新打包，以新版本（配置version参数）发布后再安装或更新包。

import toolkit4beginner

下述测试代码是对具体模块的测试。在模块调入方式上，选择了不同的调入方式，包括from toolkit4beginner.graph import graph，从子包（子目录）中调入一个模块；from toolkit4beginner.utility.general\_tools import flatten\_lst，从子包的模块中调入一个方法（函数）；from toolkit4beginner.utility.stats\_tools import \*，一次性调入一个模块中的所有方法。也可以使用import toolkit4beginner as t4b，或者from toolkit4beginner.utility.general\_tools import flatten\_lst as FL等方式为调入的对象重新命名，这尤其适合于有同名的文件或方法，避免同名冲突。

# -\*- coding: utf-8 -\*-  
"""  
Created on Sun Aug 21 15:23:51 2022  
  
@author: Richie Bao-caDesign设计(cadesign.cn)  
"""  
#测试  
#%%  
from toolkit4beginner.graph import graph  
  
test\_score\_lst\_dic={'English': [90, 81, 73, 97, 85, 60, 74, 64, 72, 67, 87, 78, 85, 96, 77, 100, 92, 86],   
 'Chinese': [71, 90, 79, 70, 67, 66, 60, 83, 57, 85, 93, 89, 78, 74, 65, 78, 53, 80],   
 'history': [73, 61, 74, 47, 49, 87, 69, 65, 36, 7, 53, 100, 57, 45, 56, 34, 37, 70],   
 'biology': [59, 73, 47, 38, 63, 56, 75, 53, 80, 50, 41, 62, 44, 26, 91, 35, 53, 68]}  
\_=graph.boxplot\_custom(test\_score\_lst\_dic,  
 figsize=(15,10),  
 fontsize=23,  
 frameOn=['bottom','left'],  
 xlabel='subject',  
 ylabel='score',  
 labelsize='30',  
 tick\_color='r',  
 notch=1,  
 sym='rs',  
 whisker\_linestyle='--',  
 whisker\_linewidth=5,  
 median\_linewidth=5  
 )  
#%%  
test\_EC\_scores={'English': [90, 81, 73, 97, 85, 60, 74, 64, 72, 67, 87, 78, 85, 96, 77, 100, 92, 86],   
 'Chinese': [71, 90, 79, 70, 67, 66, 60, 83, 57, 85, 93, 89, 78, 74, 65, 78, 53, 80], }  
test\_names=['Mason', 'Reece', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P']  
graph.four\_quadrant\_diagram(test\_EC\_scores,  
 # method='median',  
 figsize=(15,15),  
 fontsize=23,  
 frameOn=['bottom','left'],  
 labelsize='30',  
 dot\_size=200,  
 annotation\_fontsize=30,  
 annotation\_lable=test\_names,  
 )   
#%%  
from toolkit4beginner.utility.general\_tools import flatten\_lst  
  
nested\_lst=[['A','B',['C','D'],'E'],[6,[7,8,[9]]]]   
print(flatten\_lst(nested\_lst))   
#%%  
from toolkit4beginner.utility.stats\_tools import \*  
  
ranmen\_price\_lst=[700,850,600,650,980,750,500,890,880,700,890,720,680,650,790,670,680,900,880,720,850,700,780,850,750,  
 80,590,650,580,750,800,550,750,700,600,800,800,880,790,790,780,600,690,680,650,890,930,650,777,700]  
price\_x=200.68   
price\_x\_closestValue=value2values\_comparison(price\_x,ranmen\_price\_lst)   
print(price\_x\_closestValue)  
#------------------------------------------------------------------------------  
ranmen\_price\_lst=[700,850,600,650,980,750,500,890,880,  
 700,890,720,680,650,790,670,680,900,  
 880,720,850,700,780,850,750,780,590,  
 650,580,750,800,550,750,700,600,800,  
 800,880,790,790,780,600,690,680,650,  
 890,930,650,777,700]  
d\_s\_1=descriptive\_statistics(ranmen\_price\_lst)  
print(d\_s\_1)  
print('--'\*30)  
d\_s\_2=descriptive\_statistics(ranmen\_price\_lst,'std')  
print(d\_s\_2)  
d\_s\_3=descriptive\_statistics(ranmen\_price\_lst,measure='mean',decimals=1)  
print(d\_s\_3)  
#%%

['A', 'B', 'C', 'D', 'E', 6, 7, 8, 9]  
200.680 is not in the list.  
the nearest value to 200.68 is 700.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
x is lower than the average 729.34 of the list.  
700  
{'count': 50, 'mean': 743.34, 'std': 108.26, 'variance': 11720.64, 'median': 750.0, 'mode': 700, 'min': 500, 'max': 980}  
------------------------------------------------------------  
std=108.26  
mean=743.3

注释（Notes）：

① 29. Packages（python-course.eu），（<https://python-course.eu/python-tutorial/packages.php>）。

② setuptools，（<https://pypi.org/project/setuptools/>）。

③ PEP 621 – Storing project metadata in pyproject.toml，（<https://peps.python.org/pep-0621/>）。

④ wheel，（<https://pypi.org/project/wheel/>）。

⑤ PyPI.Find, install and publish Python packages with the Python Package Index，（<https://pypi.org/>）。

⑥ twine，用于在 PyPI 上发布 Python 包的实用程序（<https://pypi.org/project/twine/>）。

⑦ toolkit4beginner，（<https://pypi.org/project/toolkit4beginner/0.3.3/>）。

⑧ setuptools documentation，（<https://setuptools.pypa.io/en/latest/>）。

⑨ Configuring setuptools using pyproject.toml files，（<https://setuptools.pypa.io/en/latest/userguide/pyproject_config.html>）。

PC7-ipynb download