py_everything

Release 1.1.1

Play 4 Tutorials

Jul 17, 2021

BASIC:

1 Power of py_everything - 3

2 Contributors - 5

2.1 Modules, Functions, and Classes . . . 5
2.2 Glossary . . . 8
2.3 py_everything . . . 9
2.4 py_everything.automation . . . 10
2.5 py_everything.date_utils . . . 13
2.6 py_everything.fileIO . . . 14
2.7 pyEverything.maths . . . 17
2.8 py_everything.requestsLib . . . 20
2.9 py_everything.search . . . 24
2.10 py_everything.web . . . 26
2.11 setupPyGen . . . 29
2.12 setupPyGen Changelog . . . 31
2.13 Dependencies and Dependents . . . 31
2.14 License . . . 32

Index 33

Welcome to the Documentation for py_everything. You can find all the modules and how to use them here.

py_everything hopes to become a Python package that helps you write **everything** much faster and in a easier way.

CHAPTER

ONE

POWER OF PY_EVERYTHING -

The basic usage for this package is given below:

```
>>> import py_everything
>>> from py_everything import search
>>> search.search_files('python', 'C:\Programming\\')
C:\Programming\python.txt
C:\Programming\projectpython.py
C:\Programming\py_everything-python.docx
>>> my_list = [2, 4, 5, 3, 7, 5, 6, 3 , 12 , 9, 6]
>>> py_everything.maths.avg(my_list)
5.636363636363637
```

CHAPTER

TWO

CONTRIBUTORS -

People who have contributed to this project -

- Play 4 Tutorials(Creator and Maintainer)

2.1 Modules, Functions, and Classes

2.1.1 Detailed Package List

Modules:

- *py_everything*
- *py_everything.automation*
- *py_everything.date_utils*
- *py_everything.fileIO*
- *py_everything.maths*
- *py_everything.requestsLib*
- *py_everything.search*
- *py_everything.web*

Functions:

py_everything:

- *helloWorld*
- *printNoNewline*
- *clearPycache*
- *installModules*

py_everything.automation:

- *emailBot*
- *emailAddressSlicer*
- *ytDownloader*
- *rollDice*
- *timer*

py_everything.date_utils:

- *getDate*
- *getDateTime*
- *getTime*
- *getCustom_Format*

py_everything.fileIO:

- *readFile*
- *writeFile*
- *clearFile*

py_everything.maths:

- *add*
- *subtract*
- *multiply*
- *divide*
- *floatDiv*
- *intDiv*
- *expo*
- *mod*
- *evalExp*
- *avg*

py_everything.requestsLib:

- *getR*
- *postR*
- *putR*
- *deleteR*
- *patchR*
- *optionsR*
- *headR*
- *getContent*
- *getText*
- *getJson*
- *getHeader*
- *getSpecificHeader*

py_everything.search:

- *searchFiles*
- *searchDirs*
- *searchExts*
- *searchList*

py_evrything.web:

- *googleSearch*
- *ytSearch*
- *githubSearch*
- *soSearch*
- *amzInSearch*
- *amzComSearch*
- *pypiSearch*
- *rtdocsSearch*
- *openNewTab*
- *openNewWindow*

Classes:

py_everything.date_utils:

- *Date*

py_everything.fileIO:

- *FileIOBase*

py_everything.maths:

- *MathsBase*
- *MathsAdvanced*

py_everything.requestsLib:

- *ReqLibBase*
- *ReqLibAdvanced*

py_everything.web:

- *webSearchBase*
- *webSearchAdvanced*

2.2 Glossary

2.2.1 D

dependencies

Dependencies Other projects a project is dependent for it to work.

dependents

Dependents Other projects which are depending on this project to work.

directory

Directory A folder.

2.3 py_everything

2.3.1 Import -

How to import the module?

```
>>> import py_everything as pye
```

2.3.2 Your First Program using this package

What's the best way to write your first program than "Hello, World!"

Function name - helloWorld()

No. of Parameters - 0

Parameters - No parameters

Usage -

```
>>> pye.helloWorld()
Hello, World!
```

So, we wrote our first program. Let's move on.

2.3.3 Clear pycache

Function name - clearPycache(path)

No. of Parameters - 1

Parameters - path

Usage -

```
>>> pye.clearPycache(path\to\pycache\folder)
True
```

Provide the pth to the folder where the "**__pycache__**" is located. For example - C:ProgrammingProjectsPython

Note - Do not provide **__pycache__** in the path. That does not clear it. For example - C:ProgrammingProjectsPython__pycache__. This would return False.

2.3.4 Print without newline

Function name - printNoNewline(*args)

No. of Parameters - infinte

Parameters - *args

Usage -

```
>>> pye.printNoNewline("hello", "world", "this is printed without newline", ".")
hello world this is printed without newline .
```

Pass in the words you want to get printed without newline and That will be printed.

2.3.5 Install modules with pip

Function name - installModules(*args)

No. of Parameters - infinte

Parameters - *args

Usage -

```
>>> pye.installModules("py_everything", "requests")
True
```

This function install the given modules using `pip`. If the command is successful it returns True else False.

2.4 py_everything.automation

2.4.1 Import -

How to import the module?

```
>>> import py_everything.automation as pyeAuto
```

2.4.2 Send a mail with Python

Function name - sendMail(send_addr, password, recv_addr, body, server, port, sub='No Subject')

No. of Parameters - 7

Parameters - send_addr, password, recv_addr, body, server, port, sub='No Subject'

Usage -

```
>>> my_addr = "your.email@add.ress"
>>> my_pass = "your password"
>>> to_addr = "recv.er@add.ress"
>>> my_body = "Email body"
>>> my_server = "your.smtp.server"
>>> my_port = "123"
>>> my_sub = "My Subject"
>>> pye.sendMail(my_addr, my_pass, to_addr, my_body, my_server, my_port, my_sub)
True
```

This function sends a mail to the address passed in *recv_addr*. *sub* is a optional parameter. The mail is sent from the value of *sent_addr*. *sent_addr* and *password* are used to authenticate with the *server*. The *port* is also very important. *body*, and *sub* are the Body and Subject of the email, respectively.

Note - Speacial settings must be enabled for this function to work properly. For e.g. - In Gmail you need to enable Less secure app access.

2.4.3 Slice an email address

Function name - emailAddressSlicer(full_addr)

No. of Parameters - 1

Parameters - full_addr

Usage -

```
>>> full_addr = 'long.demo.address@long-domain.com'
>>> pye.emailAddressSlicer(full_addr)
['long.demo.address', 'long-domain.com']
```

This function is usefull to get the username and domain of a email address seperately. `full_addr` takes in the the full email address that is to sliced.

2.4.4 Roll the Dice!

Function name - rollDice(dice1=True)

No. of Parameters - 1

Parameters - dice_1

Usage -

```
>>> pye.rollDice(dice_1=True)
4
>>> pye.rollDice(dice_1=False)
8
>>> pye.rollDice(dice_1=True)
6
>>> pye.rollDice(dice_1=False)
11
```

This function doesn't need explanation but still. If `dice_1` is True then the number will be within th range of 1 to 6 but if `dice_1` is False the range is from 1 to 12. That is `dice_1` signifies the no. of dice being rolled. If True, 1 dice, else 2 die. It is True by default.

2.4.5 Run a timer

Function name - timer(seconds, audio_file)

No. of Parameters - 2

Parameters - seconds, audio_file

Usage -

```
>>> pye.timer(10, 'path/to/audio/file.mp3')
<---After 10 seconds, it plays the audio_file--->
```

This function countsdown `seconds` untill it reaches 0 and then plays `audio_file`. Both parameters are required.

2.4.6 Start or run a app or executable.

Function name - startApp(drive, app_path, exe_name)

No. of Parameters - 3

Parameters - drive, app_path, exe_name

Usage -

```
>>> pye.startApp('C', 'path/to/exe_file/file_name.exe')
True
```

This function uses the 3 parameters to run any executable.

Note - In `exe_path`, the full path, including exe name and extension are to be provided, in `drive`, only the letter is to be provided, the colon(:) is to be ommited. For example - startApp('C', 'C:/Program Files/company/exe_name.exe').

... toctree::

caption Basic:

2.5 py_everything.date_utils

2.5.1 Import -

How to import the module?

```
>>> import py_everything.date_utils as pyeDate
```

2.5.2 Get Current Date

Function Name - getDate()

No. of Parameters - 0

Parameters - None

Usage -

```
>>> pyeDate.getDate()
2021-03-17
```

This function prints the current date.

2.5.3 Get Current Date and Time

Function Name - getDateTime()

No. of Parameters - 0

Parameters - None

Usage -

```
>>> pyeDate.getDateTime()
2021-03-07 18:08:19.018239
```

This function prints the current date and time.

2.5.4 Get Current Time

Function Name - getTime()

No. of Parameters - 0

Parameters - None

Usage -

```
>>> pyeDate.getTime()
18:09:13
```

This function prints the current time.

2.5.5 Get Date and Time in Custom Format

Function Name - getCustomFormat(format)

No. of Parameters - 1

Parameters - format

Usage -

```
>>> pyeDate.getCustomFormat('%H:%M:%S')
18:09:13
```

This function prints the current date or time in `format`.

2.6 py_everything.fileIO

2.6.1 Import -

How to import the module?

```
>>> import py_everything.fileIO as pyeFiles
```

2.6.2 Read Data from a File

Function Name - readFile(filename)

No. of Parameters - 1

Parameter - filename

Usage -

```
>>> pyeFiles.readFile('path/to/file')
Data of the file is returned.
```

This function reads the data of `filename` and returns it.

2.6.3 Write Data to a File

Function Name - writeFile(filename, writeData=”)

No. of Parameters - 2

Parameter - filename, writeData

Usage -

```
>>> pyeFiles.writeFile('path/to/file', writeData="Data to be written to the file.")
True
```

This function reads the file given in `filename`, then clears all its content and then writes `writeData` to the file. If these steps are successful, it returns True.

2.6.4 Clear or Erase Contents of a File

Function Name - writeFile(filename)

No. of Parameters - 1

Parameter - filename

Usage -

```
>>> pyeFiles.clearFile('path/to/file')
True
```

This function reads the file given in `filename`, then clears all its contents. If the steps are successful, it returns True.

2.6.5 Make a New directory or Folder

Function Name - mk_dir(dir_name, path)

No. of Parameters - 2

Parameter - dir_name, path

Usage -

```
>>> pyeFiles.mk_dir('folderName', 'path/to/folder')
True
```

This function makes a new *directory* named `dir_name` in `path`. If steps are successful, returns True.

2.6.6 Make a New File

Function Name - mk_file(file_name, path)

No. of Parameters - 2

Parameter - file_name, path

Usage -

```
>>> pyeFiles.mk_file('fileName.txt', 'path/to/file/location')
True
```

This function makes a new file named `file_name` in `path`. If steps are successful, returns True.

2.6.7 Delete an Empty directory

Function Name - del_dir(path, dir_name)

No. of Parameters - 2

Parameter - path, dir_name

Usage -

```
>>> pyeFiles.del_dir('folderName', 'path/to/folder')
True
```

This function deletes an existing empty *directory* named `dir_name` that is in `path`. If steps are successful, returns True. Raises error if *directory* is not empty or does not exist.

2.6.8 Delete a directory with Items(Recursive Deletion)

Function Name - del_dir_rec(path, dir_name)

No. of Parameters - 2

Parameter - path, dir_name

Usage -

```
>>> pyeFiles.del_dir_rec('folderName', 'path/to/folder')
True
```

This function deletes an existing *directory* named `dir_name` that is in `path`. It deletes it regardless, it is empty or has items. If steps are successful, returns True. Raises error if direectory does not exist.

2.6.9 Delete a File

Function Name - del_file(path, file_name)

No. of Parameters - 2

Parameter - path, file_name

Usage -

```
>>> pyeFiles.del_file('fileName.txt', 'path/to/file/location')
True
```

This function deletes an existing file named `file_name` that is in `path`. If steps are successful, returns True. Raises error if file does not exist.

2.7 pyEverything.maths

2.7.1 Import -

How to import the module?

```
>>> import pyEverything.maths as pyeMaths
```

2.7.2 Add

Function Name - add(num1, num2, *args)

No. of Parameters - infinite

Parameters - num1, num2, *args

Usage -

```
>>> pyeMaths.add(1, 4, 3, 7, 3, 10, 13, 24)
65
```

2.7.3 Subtract

Function Name - subtract(num1, num2, *args)

No. of Parameters - infinite

Parameters - num1, num2, *args

Usage -

```
>>> pyeMaths.subtract(16, 4, 3)
9
```

2.7.4 Multiply

Function Name - multiply(num1, *args)

No. of Parameters - infinite

Parameters - num1, *args

Usage -

```
>>> pyeMaths.multiply(2, 4, 3)
24
```

2.7.5 Divide

Function Name - divide(num1, num2, type)

No. of Parameters - 3

Parameters - num1, num2, type

Usage -

```
>>> pyeMaths.divide(36, 3, 'float')
12.0
>>> pyeMaths.divide(36, 3, 'int')
12
```

2.7.6 Float Division

Function Name - floatDiv(num1, num2)

No. of Parameters - 2

Parameters - num1, num2

Usage -

```
>>> pyeMaths.floatDiv(36, 3)
12.0
```

2.7.7 Integer Division

Function Name - intDiv(num1, num2)

No. of Parameters - 2

Parameters - num1, num2

Usage -

```
>>> pyeMaths.intDiv(36, 3)
12
```

2.7.8 Exponent

Function Name - expo(num1, num2)

No. of Parameters - 2

Parameters - num1, num2

Usage -

```
>>> pyeMaths.expo(2, 4)
16
```

2.7.9 Modulus

Function Name - mod(num1, num2)

No. of Parameters - 2

Parameters - num1, num2

Usage -

```
>>> pyeMaths.mod(17, 2)
1
```

2.7.10 Evaluate Any Mathematics Expression

Function Name - evalExp(exp)

No. of Parameters - 1

Parameters - exp

Usage -

```
>>> pyeMaths.expo('2 + 4 - 1 * 2')
10
```

2.7.11 Average

Function Name - avg(listOfNos)

No. of Parameters - 1

Parameters - listOfNos

Usage -

```
>>> pyeMaths.avg(['2', '3', '1', '5', '8', '4', '3'])
3.7142857142857144
```

2.7.12 Note

These functions have not been explained seperately because they explain themselves.

2.8 py_everything.requestsLib

2.8.1 Import -

How to import the module?

```
>>> import py_everything.requestsLib as pyeRLib
```

2.8.2 Send a GET Request

Function Name - getR(apiUrl)

No. of Parameters - 1

Parameters - apiUrl

Usage -

```
>>> pyeRLib.getR("protocol://api.domain.com/user/1")
<Response [200]>
```

This function uses requests to send a get request to `apiUrl`.

2.8.3 Send a POST Request

Function Name - postR(apiUrl, data=None)

No. of Parameters - 2

Parameters - apiUrl, data

Usage -

```
>>> pyeRLib.postR("protocol://api.domain.com/users", "{\"id\":1}")
<Response [200]>
```

This function uses requests to send a post request to `apiUrl`.

2.8.4 Send a PUT Request

Function Name - putR(apiUrl, data=None)

No. of Parameters - 2

Parameters - apiUrl, data

Usage -

```
>>> pyeRLib.putR("protocol://api.domain.com/users", data="{\"id\":1")
<Response [200]>
```

This function uses requests to send a put request to `apiUrl`.

2.8.5 Send a DELETE Request

Function Name - deleteR(apiUrl)

No. of Parameters - 1

Parameters - apiUrl

Usage -

```
>>> pyeRLib.deleteR("protocol://api.domain.com/user/1")
<Response [200]>
```

This function uses requests to send a delete request to `apiUrl`.

2.8.6 Send a PATCH Request

Function Name - patchR(apiUrl, data=None)

No. of Parameters - 2

Parameters - apiUrl, data

Usage -

```
>>> pyeRLib.patchR("protocol://api.domain.com/users", data="\"id\":1")
<Response [200]>
```

This function uses requests to send a patch request to `apiUrl`.

2.8.7 Send a OPTIONS Request

Function Name - optionsR(apiUrl)

No. of Parameters - 1

Parameters - apiUrl

Usage -

```
>>> pyeRLib.optionsR("protocol://api.domain.com/users")
<Response [200]>
```

This function uses requests to send a options request to `apiUrl`.

2.8.8 Send a HEAD Request

Function Name - headR(apiUrl)

No. of Parameters - 1

Parameters - apiUrl

Usage -

```
>>> pyeRLib.headR("protocol://api.domain.com/users")
<Response [200]>
```

This function uses requests to send a head request to `apiUrl`.

2.8.9 Get Content from a Response

Function Name - getContent(response)

No. of Parameters - 1

Parameters - response

Usage -

```
>>> pyeRLib.getContent(pyeRLib.getR("protocol://api.domain.com/user/1"))
<--Content of the reponse-->
```

This function uses requests to get the content of `response`.

2.8.10 Get Text from a Response

Function Name - getText(response)

No. of Parameters - 1

Parameters - response

Usage -

```
>>> pyeRLib.getText(pyeRLib.getR("protocol://api.domain.com/user/1"))
<--Text of the reponse-->
```

This function uses requests to get the text returned to `response`.

2.8.11 Get JSON from a Response

Function Name - getJson(response)

No. of Parameters - 1

Parameters - response

Usage -

```
>>> pyeRLib.getJson(pyeRLib.getR("protocol://api.domain.com/user/1"))
<--JSON of the reponse-->
```

This function uses requests to get the JSON returned to `response`.

2.8.12 Get Headers from a Response

Function Name - getHeader(response)

No. of Parameters - 1

Parameters - response

Usage -

```
>>> pyeRLib.getHeader(pyeRLib.getR("protocol://api.domain.com/user/1"))
<--Headers of the reponse-->
```

This function uses requests to get the Headers returned to `response`.

2.8.13 Get a Specific Header from a Response

Function Name - getSpecificHeader(response, headerName)

No. of Parameters - 2

Parameters - response, headerName

Usage -

```
>>> pyeRLib.getSpecificHeader(pyeRLib.getR("protocol://api.domain.com/user/1"))
<--Content of headerName of the reponse-->
```

This function uses requests to get the content of `headerName` returned to `response`.

2.9 py_everything.search

2.9.1 Import -

How to import the module?

```
>>> import py_everything.search as pyeSearch
```

2.9.2 Search Files

Function Name - searchFiles(keyword, path)

No. of Parameters - 2

Parameters - keyword, path

Usage -

```
>>> pyeSearch.searchFiles("mykeyword", "D:")
['D:\mykeyword.txt', 'D:\file-with-mykeyword.pdf', 'D:\myfolder\pymykeyword.py']
```

This function searches for files with `keyword` in them in `path`. And returns a list containing the full path of those files.

2.9.3 Search Folders

Function Name - searchDirs(keyword, path)

No. of Parameters - 2

Parameters - keyword, path

Usage -

```
>>> pyeSearch.searchDirs("myfolder", "D:")
['D:\folder\myfolder', 'D:\myfolder-py']
```

This function searches for folders or directories with `keyword` in them in `path`. And returns a list containing the full path of those folders or directories.

2.9.4 Search by Extension

Function Name - searchExts(ext, path)

No. of Parameters - 2

Parameters - ext, path

Usage -

```
>>> pyeSearch.searchExts("txt", "D:")
['D:\folder\file.txt', 'D:\py.txt']
```

This function searches for files with the extension `ext` in `path`. And returns a list containing the full path of those files.

2.9.5 Search in Lists

Function Name - searchList(listOfTerms, query, filter='in')

No. of Parameters - 3

Parameters - listOfTerms, query, filter

Usage -

```
>>> mylist = ["python", "pipypi", "py", "endpy", "other", "file", "everything"]
>>> pyeSearch.searchList(mylist, "py", filter="in")
['python', 'pipypi', 'py', 'endpy']
>>> pyeSearch.searchList(mylist, "py", filter="start")
['python', 'py']
>>> pyeSearch.searchList(mylist, "py", filter="end")
['endpy']
>>> pyeSearch.searchList(mylist, "py", filter="exact")
['py']
```

This function searches for items with `query` in them in `listOfItem`. And returns a list containing the items. `filter` can have 4 values, i.e., 'in', 'start', 'end', 'exact'. 'in' returns items which have `query` in them. 'start' returns items which start with `query`. 'end' returns items which end with `query`. 'exact' returns items which are exactly the `query`.

2.10 py_everything.web

2.10.1 Import -

How to import the module?

```
>>> import py_everything.web as pyeWeb
```

2.10.2 Search Google

Function Name - googleSearch(query)

No. of Parameters - 1

Parameters - query

Usage -

```
>>> pyeWeb.googleSearch("py_everything")
<--It opens a tab in your browser and shows the results-->
```

This function searches Google for `query`.

2.10.3 Search Youtube

Function Name - ytSearch(query)

No. of Parameters - 1

Parameters - query

Usage -

```
>>> pyeWeb.ytSearch("py_everything")
<--It opens a tab in your browser and shows the results-->
```

This function searches YouTube for `query`.

2.10.4 Search GitHub

Function Name - githubSearch(query)

No. of Parameters - 1

Parameters - query

Usage -

```
>>> pyeWeb.githubSearch("py_everything")
<--It opens a tab in your browser and shows the results-->
```

This function searches GitHub for `query`.

2.10.5 Search Stack Overflow

Function Name - soSearch(query)

No. of Parameters - 1

Parameters - query

Usage -

```
>>> pyeWeb.soSearch("py_everything")
<--It opens a tab in your browser and shows the results-->
```

This function searches Stack Overflow for `query`.

2.10.6 Search Amazon.in

Function Name - amzInSearch(query)

No. of Parameters - 1

Parameters - query

Usage -

```
>>> pyeWeb.amzInSearch("py_everything")
<--It opens a tab in your browser and shows the results-->
```

This function searches Amazon.in for `query`.

2.10.7 Search Amazon.com

Function Name - amzComSearch(query)

No. of Parameters - 1

Parameters - query

Usage -

```
>>> pyeWeb.amzComSearch("py_everything")
<--It opens a tab in your browser and shows the results-->
```

This function searches Amazon.com for `query`.

2.10.8 Search PyPI

Function Name - pypiSearch(query)

No. of Parameters - 1

Parameters - query

Usage -

```
>>> pyeWeb.pypiSearch("py_everything")
<--It opens a tab in your browser and shows the results-->
```

This function searches PyPI for `query`.

2.10.9 Search Read The Docs

Function Name - rtdocsSearch(query)

No. of Parameters - 1

Parameters - query

Usage -

```
>>> pyeWeb.rtdocsSearch("py_everything")
<--It opens a tab in your browser and shows the results-->
```

This function searches Read the Docs for `query`.

2.10.10 Search in a New Tab

Function Name - openNewTab(url, query)

No. of Parameters - 2

Parameters - url, query

Usage -

```
>>> google_url = 'https://google.com/search?q='
>>> pyeWeb.openNewTab(google_url, "py_everything")
<--It opens a tab in your browser and shows the results-->
```

This function searches Google for query in a new tab.

2.10.11 Search in a New Window

Function Name - openNewWindow(url, query)

No. of Parameters - 2

Parameters - url, query

Usage -

```
>>> gh_url = 'https://github.com/search?q='
>>> pyeWeb.openNewWindow(gh_url, "py_everything")
<--It opens a window in your browser and shows the results-->
```

This function searches GitHub for query in a new window.

2.11 setupPyGen

2.11.1 Basic Usage:

```
$ ls
package/
$ cd package/
$ ls -a
. ..
$ setupPyGen -g True -t True --gitignore True
<--Follow the prompts(packages entered - new, old)-->
$ ls -a
. .. .gitignore LICENSE README.md setup.py .git/ new/ old/ tests/
$ cat setup.py
from setuptools import setup

readme_file = open("README.md", "r").read()

setup(
   name="package-name",
   version="1.0.0",
   description="Given Project Description",
   long_description=readme_file,
   long_description_content_type="text/markdown",
   author="Author Name",
   author_email="name@example.com",
```

(continues on next page)

(continued from previous page)

```
    packages=[new, old],
    install_requires=[],
    license="MIT License",
    url="https://github.com/play4Tutorials/py_everything/",
    python_requires='>=3.5'
)
```

2.11.2 Flags:

There are different flags for setupPyGen. These flags take True or nothing.

- -g or –git
- -t or –tests
- –gitignore

All of these flags are optional. Rest of the data is taken input after running the command.

2.11.3 -g or –git

The -g or –git flag is used to initialize a Git repository after the project structure and setup.py has been generated. It is usually combined with the –gitignore falg for best results.

2.11.4 –gitignore

The –gitignore flag generates a .gitignore file in the project structure after everything else. It gives best results when used with the -g or –git flag.

2.11.5 -t or –tests

The -t or –tests flag is used to generate a tests directory in the project structure for unit tests.

2.11.6 Flags Usage:

```
$ setupPyGen -g True --tests True --gitignore True
<--Folow the prompts(entered packages - new, old)-->
$ ls -A
.gitignore LICENSE README.md setup.py .git/ new/ old/ tests/

$ setupPyGen -g True --gitignore True
<--Folow the prompts(entered packages - new, old)-->
$ ls -A
.gitignore LICENSE README.md setup.py .git/ new/ old/

$ setupPyGen -g True -t True
<--Folow the prompts(entered packages - new, old)-->
$ ls -A
LICENSE README.md setup.py .git/ new/ old/ tests/
```

(continues on next page)

(continued from previous page)

```
$ setupPyGen
<--Folow the prompts(entered packages - new, old)-->
$ ls -A
LICENSE README.md setup.py new/ old/
```

2.11.7 Note:

setupPyGen, is a command-line utility seperate from the rest of the package.

It cannot be run using *$ python -m setupPyGen*. It gives an error. It can only be run using *$ setupPyGen*.

It's help utility can be accessed by using the command *$ setupPyGen -h* or *$setupPyGen –help*.

If you want to enable a flag just use "-flag True", for e.g., - *$ setupPyGen -t True*. All flags are disabled by default.

Do not disable flags manually, such as *$ setupPyGen -t False*, this still generates a tests/ directory.

2.12 setupPyGen Changelog

2.12.1 v1.0.0

- Initial Release
- Generate setup.py
- Generate Python Package Project Structure
- Start git repository
- Add tests/ folder
- Add .gitignore
- Create README.md
- Add LICENSE

2.13 Dependencies and Dependents

2.13.1 py_everything depends on -

- requests
- playsound
- pytube

2.13.2 py_everything is depended upon by -

2.14 License

2.14.1 Need Help?

Need help with anything? Join the GitHub Discussions.

INDEX

D

Dependencies, 8
dependencies, 8
Dependents, 8
dependents, 8
Directory, 8
directory, 8