py_everything

Release 2.0.0

PyBash

Apr 07, 2022

BASIC:

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. Without importing many libraries for different tasks. Do them with the help of one.

CHAPTER

ONE

POWER OF PY_EVERYTHING -

The basic usage for this package is given below:

```
>>> import py_everything
>>> from py_everything import search
>>> search.searchFiles('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 -

- pybash(Creator and Maintainer)
- Farid(Contributor)
- Morgan-Phoenix(Contributor and Collaborator)
- tsukuyomm(Contributor)

2.1 py_everything

Source code: py_everything/__init__.py

This module contains some basic functions. This is the base module of the library.

py_everything.**helloWorld**()

Super Simple and Basic function that prints "Hello, World!"

py_everything.**printNoNewline**(**args*)

Prints text without newlines. Very basic function.

Parameters ***args** – The text you want to print without newlines

Note: You cannot customize what is printed instead of the newline.

py_everything.**clearPycache**(*path*)

Deletes *__pycache__* folder from *path*.

Parameters **path** (*str*) – Full path to the folder which contains *__pycache__*

Returns **bool** True if *__pycache__* is deleted successfully.

Raises ***error.pycacheNotFoundError*** – This exception is raised if `path` does not contain `__pycache__`.

py_everything.**installModules**(**args*)

Install modules using `pip`, while execution.

Parameters ***args** – Modules you want to install.

Returns **bool** True if all modules were installed successfully.

Raises **error.installModulesFailedError** – This exception is raised if all modules could not be installed successfully. Occurs if package doesn't exist.

py_everything.**alphabet**()

Get a list of all alphabets in lowercase.

Returns list List containing all alphabets in lowercase in alphabetical order.

py_everything.**alphabetCaps**()

Get a list of all alphabets in uppercase.

Returns list List containing all alphabets in uppercase in alphabetical order.

py_everything.**alphabetStr**()

Get a string of all alphabets in lowercase.

Returns str String containing all alphabets in lowercase in alphabetical order.

py_everything.**alphabetCapsStr**()

Get a string of all alphabets in uppercase.

Returns str String containing all alphabets in uppercase in alphabetical order.

py_everything.**nums**()

Get a list of all numbers(0-9).

Returns list List containing all numbers(0-9) in ascending order.

py_everything.**syms**()

Get a list of all symbols.

Returns list List containing all symbols.

2.2 py_everything.automation

Source code: py_everything/automation.py

This module contains methods that automate certain things or tasks such as sending a mail.

py_everything.automation.**sendEmail**(*sendAddr*, *password*, *recvAddr*, *body*, *server*, *port*, *sub='No Subject'*)

Sends email to `recvAddr` from `sendAddr`. With `body` as mail body and `sub` as mail subject. Uses `server` and `port` to send the mail.

Parameters

- **sendAddr** (*str*) – The address you want the mail to be sent from.
- **password** (*str*) – To login to the email account.
- **recvAddr** (*str*) – The address to which the mail is to be sent.
- **body** (*str*) – The main body of the email.
- **server** (*str*) – The server through which the mail should be sent.
- **port** (*str*) – The port at which the server is listening.
- **sub** (*str*) – The optional subject of the mail. Defaults to 'No Subject' if not specified.

Returns bool True if mail gets sent successfully.

Note: Less secure app access should be turned on for Gmail. IMAP/POP Forwarding should be enabled in mail settings for this to work. Alos, the server and port should be correct.

---

py_everything.automation.**emailAddressSlicer**(*fullAddr*)

Slices an email address and returns username and domain separately.

Parameters **fullAddr** (*str*) – The full address you want to slice.

Returns tuple Contaning username and domain..;

py_everything.automation.**rollDice**(*dice1=True*)

Rolls dice and returns value between 1 and 6 if dice1=True else returns value between 1 and 12.

Parameters **dice1** (*str*) – Boolean to understand if 1 dice to roll or 2 dice.

Returns int Value between 1 and 6 or 1 and 12.

py_everything.automation.**timer**(*seconds*, *audioFile*)

Starts a timer for `seconds` and plays `audioFile` when finished.

Parameters **seconds** (*int*) – How many seconds should the timer be for.

py_everything.automation.**startApp**(*exePath*)

Starts `exePath`.

Parameters **exePath** (*str*) – Full path to the exe to be launched.

Returns bool True if exe starts successfully.

Raises ***error.startAppFailedError*** – This exception is raised if exe was not started successfully. Maybe due to an incorrect path.

2.3 py_everything.bencrypt

Source code: py_everything/bencrypt.py

This module deals with encryption using the enrocrypt library

New in version 2.1.0.

py_everything.bencrypt.**encrypt**(*data*)

Returns List containing specially formatted encrypted data for `data`.

returns List List containing specially formatted encrypted data for `data`

py_everything.bencrypt.**decrypt**(*key*, *data*)

Returns decrypted data using `key`.

returns bytes Str/bytes containing decrypted data using key.

Note: This requires you to provide the key and data separately in 2 arguments from the encrypted List.

py_everything.bencrypt.**listDecrypt**(*encryptedList*)

Returns decrypted data using key from `encryptedList`.

returns bytes Str/bytes containing decrypted data using key.

py_everything.bencrypt.**encryptFile**(*filepath*, *keyFilepath*)

Returns bool depending on encryption successful or not.

returns bool True if encryption successful, Flase if not

py_everything.bencrypt.**decryptFile**(*filepath*, *keyFilepath*)

Returns bool depending on decryption successful or not.

returns bool True if decryption successful, Flase if not

2.4 py_everything.conversion

Source code: py_everything/conversion.py

This module conatins classes for conversion like, Mass, Length, etc. And it deals with conversion of units.

class py_everything.conversion.**Mass**(*unit*, *amount*)

This class is used for creating an object for Mass values and units

```
>>> from py_everything.conversion import Mass, convert
>>> from py_everything.units import mg, g
>>> mymass = Mass(units.mg(), 1000)
>>> mymass2 = Mass(units.g(), 1000)
>>> converted = convert(mymass, mymass2)
>>> converted
1.0
```

Parameters

- **unit** (*Union*) – Any mass unit class from units module.
- **amount** (*int*) – Value of unit.

Note: Even though amount is given input to both Mass classes, the amount in fromType is only used.

class py_everything.conversion.**Volume**(*unit*, *amount*)

This class is used for creating an object for Volume values and units

```
>>> from py_everything.conversion import Volume, convert
>>> from py_everything.units import l, ml
>>> mymass = Mass(units.l(), 1)
>>> mymass2 = Mass(units.ml(), 1)
>>> converted = convert(mymass, mymass2)
>>> converted
1000.0
```

Parameters

- **unit** (*Union*) – Any volume unit class from units module.
- **amount** (*int*) – Value of unit.

Note: Even though amount is given input to both Volume classes, the amount in fromType is only used.

class py_everything.conversion.**Length**(*unit*, *amount*)

This class is used for creating an object for Length values and units

```
>>> from py_everything.conversion import Length, convert
>>> from py_everything.units import mm, m
>>> mymass = Mass(units.mm(), 500)
>>> mymass2 = Mass(units.m(), 1000)
>>> converted = convert(mymass, mymass2)
>>> converted
0.5
```

Parameters

- **unit** (*Union*) – Any length unit class from units module.
- **amount** (*int*) – Value of unit.

Note: Even though amount is given input to both Length classes, the amount in fromType is only used.

py_everything.conversion.**convert**(*fromType*, *toType*)

Converts value from unit to another.

Parameters

- **fromType** (*Union*) – Unit to convert from.
- **toType** (*Union*) – Unit to convert to.

Returns float Value after conversion.

2.5 py_everything.dateUtils

Source code: py_everything/dateUrils.py

This module deals date and time. Like, fetching current date, time, etc.

py_everything.dateUtils.**getDate**()

This method fetches the date the program is being executed on.

Returns The date program is being executed on.

py_everything.dateUtils.**getDateTime**()

This method fetches the date and time the program is being executed on.

Returns The date and time program is being executed on.

Note: This method returns a float for seconds of the time, like, 12 seconds would be 12.45365. It is very precise.

py_everything.dateUtils.**getTime**()

This method fetches the time the program is being executed on.

Returns The time program is being executed on.

py_everything.dateUtils.**getCustomFormat**(*format*)

This method fetches the date and/or time in a custom format.

Parameters **format** (*str*) – The format in which the date and/or time should be returned.

Returns The date and/or current time.

Note: strftime format is used in `format`. To know more about it see this and this.

2.6 py_everything.error

Source code: py_everything/error.py

class py_everything.error.**pycacheNotFoundError**

Exception raised when *__pycache__* is not found.

class py_everything.error.**instalModulesFailedError**

Exception raised when modules can't be installed successfully and fails.

class py_everything.error.**startAppFailedError**

Exception raised when exe launch fails.

class py_everything.error.**InvalidKeyListError**

Exception raised when key list is invalid in `sencrypt`.

New in version 2.0.0.

class py_everything.error.**InvalidSymbolKeyError**

Exception raised when symbol key is invalid in `sencrypt`.

New in version 2.0.0.

class py_everything.error.**InvalidOperationPerformedError**

Exception raised when unsupported operation is performed on a path

New in version 2.1.0.

class py_everything.error.**UnknownPathTypeError**

Exception raised when path type can't be determined

New in version 2.1.0.

class py_everything.error.**UnknownDivisionTypeError**

Exception raised when division type can't be determined

New in version 2.1.0.

2.7 py_everything.fileIO

Source code: py_everything/fileIO.py

This module deals with files and their input/output operations. With a wide range of functions.

py_everything.fileIO.**readFile**(*fileName*)

This method reads data from `fileName`.

Parameters **fileName** (*str*) – Full path to the file to be read.

Returns str Data of the file.

py_everything.fileIO.**writeFile**(*fileName*, *writeData*)

This method writes new data - `writeData` on to `fileName`

Parameters

- **fileName** (*str*) – Full path to the file to writen to.
- **writeData** (*str*) – Data to be written on to the file.

Returns bool True if data was successfully written to file.

Changed in version 2.0.0: Raises `TypeError` if `writeData` type is not `str`

Note: This method deletes any previous data on the file. Before writing to it.

---

py_everything.fileIO.**clearFile**(*fileName*)

This method removes all data from `fileName`.

Parameters **fileName** (*str*) – Full path to the file to be cleared.

Returns bool True if file is cleared successfully.

py_everything.fileIO.**mkDir**(*dirName*, *path*)

Creates a new directory named `dirName` inside `path`.

Parameters

- **dirName** (*str*) – Name of the directory to be created.
- **path** (*str*) – Full path where directory is to be created.

Returns bool True if directory is created successfully.

py_everything.fileIO.**mkFile**(*fileName*, *path*)

Creates a new file named `fileName` inside `path`.

Parameters

- **fileName** (*str*) – Name of the file to be created.
- **path** (*str*) – Full path where file is to be created.

Returns bool True if file is created successfully.

py_everything.fileIO.**delDir**(*path*, *dirName*)

Deletes an existing directory named `dirName` from `path`.

Parameters

- **dirName** (*str*) – Name of the directory to be deleted.

- **path** (*str*) – Full path where directory is located.

Returns bool True if directory is deleted successfully.

Note: This function is for empty directories only, for directories containing files or subfolders, see the next method.

---

py_everything.fileIO.**delDirRec**(*path*, *dirName*)

Deletes an existing directory named `dirName` from `path` recursively.

Parameters

- **dirName** (*str*) – Name of the directory to be deleted.
- **path** (*str*) – Full path where directory is located.

Returns bool True if directory is deleted successfully.

py_everything.fileIO.**delFile**(*path*, *fileName*)

Deletes an existing file named `fileName` from `path`.

Parameters

- **dirName** (*str*) – Name of the file to be deleted.
- **path** (*str*) – Full path where file is located.

Returns bool True if file is deleted successfully.

2.8 py_everything.htmlXml

Source code: py_everything/htmlXml.py

This module deals with HTML/XML Files. This module will be extended in later releases. With functions that fetch tags from document for you to a class allowing all methods in one! This module doesn't check if the HTML/XML file is valid or not. It will return matches if a certain tag is not closed. This module was added in version 2.0.0

Changed in version 2.3.0: All functions in this module return a tuple `(lineNo, line)`

class py_everything.htmlXml.**HTMLObject**(*fileName*)

This class access to all methods without having to give the fileName everytime.

```
>>> from py_everything.html import HTMLObject
>>> myHtml = HTMLObject('C:/index.html')
>>> divs = myHtml.getElementsByTag('div')
>>> divs
["<div id='app'>This is main app</div>", "<div>Other part of HTML</div>"]
>>> title = myHtml.getElementByTag('title')
>>> title
['<title>Demo Website</title>']
>>> mainApp = myHtml.getElementById('app)
>>> mainApp
["<div id='app'>This is main app</div>"]
```

Parameters **fileName** (*str*) – A string containing full path to HTML/XML file.

getElementsByTag(*tagName*)

Searches HTML/XML file for given tagName.

Parameters **tagName** (*str*) – The tag you want to search for.

Returns tuple A tuple containg line number and the line in following order - (lineNo, line)

Note: The whole line is returned if a match is found. And the tag is not validated.

getElementsById(*idName*)

Searches HTML/XML file for given tags with the id of idName.

Parameters **idName** (*str*) – The id you want to search for.

Returns tuple A tuple containg line number and the line in following order - (lineNo, line)

Note: The whole line is returned if a match is found. And the tag is not validated.

getElementsByClass(*className*)

Searches HTML/XML file for given tags with the class of className.

Parameters **className** (*str*) – The class you want to search for.

Returns tuple A tuple containg line number and the line in following order - (lineNo, line)

Note: The whole line is returned if a match is found. And the tag is not validated.

getElementByTag(*tagName*)

Searches HTML/XML file for given tagName. And returns only the first match.

Parameters **tagName** (*str*) – The tag you want to search for.

Returns list A list containing first match in str.

Note: The whole line is returned if a match is found. And the tag is not validated.

getElementById(*idName*)

Searches HTML/XML file for given tags with the id of idName. And returns only the first match.

Parameters **idName** (*str*) – The id you want to search for.

Returns list A list containing first match in str.

Note: The whole line is returned if a match is found. And the tag is not validated.

getElementByClass(*className*)

Searches HTML/XML file for given tags with the class of className. And returns only the first match.

Parameters **className** (*str*) – The class you want to search for.

Returns list A list containing first match in str.

Note: The whole line is returned if a match is found. And the tag is not validated.

py_everything.htmlXml.**getElementsByTag**(*tagName*, *fileName*)

Searches HTML/XML file `fileName` for given `tagName`.

Parameters

- **tagName** (*str*) – The tag you want to search for.
- **fileName** (*str*) – A string containing full path to HTML/XML file.

Returns tuple A tuple containg line number and the line in following order - `(lineNo, line)`

Note: The whole line is returned if a match is found. And the tag is not validated.

py_everything.htmlXml.**getElementsById**(*idName*, *fileName*)

Searches HTML/XML file `fileName` for given tags with the id of `idName`.

Parameters

- **idName** (*str*) – The id you want to search for.
- **fileName** (*str*) – A string containing full path to HTML/XML file.

Returns tuple A tuple containg line number and the line in following order - `(lineNo, line)`

Note: The whole line is returned if a match is found. And the tag is not validated.

py_everything.htmlXml.**getElementsByClass**(*className*, *fileName*)

Searches HTML/XML file `fileName` for given tags with the class of `className`.

Parameters

- **className** (*str*) – The class you want to search for.
- **fileName** (*str*) – A string containing full path to HTML/XML file.

Returns tuple A tuple containg line number and the line in following order - `(lineNo, line)`

Note: The whole line is returned if a match is found. And the tag is not validated.

py_everything.htmlXml.**getElementByTag**(*tagName*, *fileName*)

Searches HTML/XML file `fileName` for given `tagName`. And returns only the first match.

Parameters

- **tagName** (*str*) – The tag you want to search for.
- **fileName** (*str*) – A string containing full path to HTML/XML file.

Returns list A list containing first match in str.

Note: The whole line is returned if a match is found. And the tag is not validated.

py_everything.htmlXml.**getElementById**(*idName*, *fileName*)

Searches HTML/XML file `fileName` for given tags with the id of `idName`. And returns only the first match.

Parameters

- **idName** (*str*) – The id you want to search for.
- **fileName** (*str*) – A string containing full path to HTML/XML file.

Returns list A list containing first match in str.

Note: The whole line is returned if a match is found. And the tag is not validated.

---

py_everything.htmlXml.**getElementByClass**(*className*, *fileName*)

Searches HTML/XML file `fileName` for given tags with the class of `className`. And returns only the first match.

Parameters

- **className** (*str*) – The class you want to search for.
- **fileName** (*str*) – A string containing full path to HTML/XML file.

Returns list A list containing first match in str.

Note: The whole line is returned if a match is found. And the tag is not validated.

2.9 py_everything.maths

Source code: py_everything/maths.py

This module deals with mathematical functions and operations.

py_everything.maths.**add**(*num1*, *num2*, **args*)

Function for adding 2 or more numbers.

Parameters

- **num1** (*Union*) – First Number.
- **num2** (*Union*) – Second Number.
- ***args** – Rest numbers.

Returns Union Result

py_everything.maths.**subtract**(*num1*, *num2*, **args*)

Function for subtracting 2 or more numbers.

Parameters

- **num1** (*Union*) – First Number.
- **num2** (*Union*) – Second Number.
- ***args** – Rest numbers.

Returns Union Result

py_everything.maths.**multiply**(*num1*, *num2*, **args*)

Function for multipling 2 or more numbers.

Parameters

- **num1** (*Union*) – First Number.
- **num2** (*Union*) – Second Number.
- ***args** – Rest numbers.

Returns Union Result

py_everything.maths.**divide**(*num1*, *num2*, *type*)

Function for dividing 2 numbers.

Parameters

- **num1** (*Union*) – First Number.
- **num2** (*Union*) – Second Number.
- **type** (*str*) – Integer division or float division.

Returns Union Result

Raises *error.UnknownDivisionTypeError* – Raised if division type can't be determined.

py_everything.maths.**floatDiv**(*num1*, *num2*)

Parameters

- **num1** (*Union*) – First Number.
- **num2** (*Union*) – Second Number.

Returns Union Result

py_everything.maths.**intDiv**(*num1*, *num2*)

Parameters

- **num1** (*Union*) – First Number.
- **num2** (*Union*) – Second Number.

Returns Union Result

py_everything.maths.**expo**(*num1*, *num2*)

Parameters

- **num1** (*Union*) – First Number.
- **num2** (*Union*) – Second Number.

Returns Union Result

py_everything.maths.**mod**(*num1*, *num2*)

Parameters

- **num1** (*Union*) – First Number.
- **num2** (*Union*) – Second Number.

Returns Union Result

py_everything.maths.**evalExp**(*exp*)

Parameters **exp** (*Union*) – Mathematical Expression

py_everything.maths.**avg**(*listOfNos*)

Parameters **listOfNos** (*Union*) – List Of Nos. for average.

Returns float Average of nos.

py_everything.maths.**factorial**(*num*)

Parameters **num** (*Union*) – Number for Factorial.

Returns int Result of factorial

py_everything.maths.**ceil**(*num*)

Parameters **num** (*Union*) – Number for rounding up.

Returns int Result

py_everything.maths.**floor**(*num*)

Parameters **num** (*Union*) – Number for rounding down.

Returns int Result

2.10 py_everything.mensuration

Source code: py_everything/mensuration.py

This module conatins functions for mensuration.

py_everything.mensuration.**areaRect**(*length*, *breadth*)

Function to find the area of a rectangle

Parameters

- **length** (*float*) – Length of the rectangle
- **breadth** (*float*) – Breadth of the rectangle

Returns float area Area of the rectabgle

py_everything.mensuration.**perimeterRect**(*length*, *breadth*)

Function to find the perimeter of a rectangle

Parameters

- **length** (*float*) – Length of the rectangle
- **breadth** (*float*) – Breadth of the rectangle

Returns float perimeter Perimeter of the rectangle

py_everything.mensuration.**areaSqr**(*side*)

Function to find the area of a square.

Parameters **side** (*float*) – Side of the square

Returns float area Area of the square

py_everything.mensuration.**perimeterSqr**(*side*)

Function to find the perimeter of a square

Parameters **side** (*float*) – Side of the square

Returns float perimeter Perimeter of the square

py_everything.mensuration.**areaTriangle**(*side*)

Function to find the area of a triangle

Parameters

- **base** (*float*) – Base of the triangle
- **height** (*float*) – Height of the triangle

Returns float area Area of the triangle

py_everything.mensuration.**perimeterTriangle**(*side1*, *side2*, *base*)

Function to find the perimeter of a triangle

Parameters

- **side1** (*float*) – Side 1 of the triangle
- **side2** (*float*) – Side 2 of the triangle

Returns float perimeter Perimeter of the triangle

py_everything.mensuration.**areaCirc**(*radius*)

Function to find the area of a circle

Parameters **radius** (*float*) – Radius of the circle

Returns float area Area of the circle

py_everything.mensuration.**circumferenceCirc**(*radius*)

Function to find the circumference of a circle

Parameters **radius** (*float*) – Radius of the circle

Returns float area Circumference of the circle

py_everything.mensuration.**volCyl**(*radius*, *height*)

Functio to find the volume of a cylinder

Parameters

- **radius** (*float*) – Radius of the cylinder
- **height** (*float*) – Height of the cylinder

Returns float volume Volume of the cylinder

py_everything.mensuration.**volCone**(*radius*, *height*)

Function to find the volume of a cone

Parameters

- **radius** (*float*) – Radius of the cone
- **height** (*float*) – Height of the cone

Returns float volume Volume of the cone

py_everything.mensuration.**volSphere**()

Function to find the volume of a sphere

Parameters **radius** (*float*) – Radius of the sphere

Returns float volume Volume of the sphere

py_everything.mensuration.**volCube**()

Function to find the volume of a cube

Parameters **edge** (*float*) – Edge of the cube

Returns float volume Volume of the sphere

py_everything.mensuration.**volCuboid**()

Function to find the volume of a cuboid

Parameters

- **length** (*float*) – Length of the cuboid
- **breadth** (*float*) – Breadth of the cuboid
- **height** (*float*) – Height of the cuboid

Returns float volume Volume of the cuboid

py_everything.mensuration.**pival**()

Function to get the value of pi

Returns float pi Value of pi

py_everything.mensuration.**eval_**()

Function to get the value of e

Returns float e Value of e

Note: The name of the function is `eval_` and not just `eval`. `eval()` is an predefined function in python. Do not confuse this.

2.11 py_everything.rand

Source code: py_everything/rand.py

This module Random generators for many things such as a random hex code generator, random RGB color generator, random float generator, and much more!

py_everything.htmlXml.**randhex**()

Generates a random hex color code such as `#2A34FD`

Returns string The hex color code as a string

py_everything.htmlXml.**randRGB**()

Generates a random RGB color and returns that as a tuple in the format - `(red, green, blue)`

Returns tuple The RGB color as a tuple

py_everything.htmlXml.**randletter**()

Generates a random alphabet and returns that as a string.

Returns string The random alphabet

py_everything.htmlXml.**randint**(*start*, *end*)

Generates a random integer from among a given range and returns that

Parameters

- **start** (*int*) – Start value of range
- **end** (*int*) – End value of range

Returns int The random integer

py_everything.htmlXml.**randfloat**(*start*, *end*)

Generates a random float between given range

Parameters

- **start** (*float*) – Start value of range
- **end** (*float*) – End value of range

Returns float The random float

py_everything.htmlXml.**randbool**()

Generates a random boolean, either `True` or `False`

Returns bool The random boolean

py_everything.htmlXml.**randboolList**(*len*)

Generates a list of random boolean values

Parameters **len** (*int*) – Length of list

Returns list The list of randomly generated boolean values

py_everything.htmlXml.**randstring**(*string*)

Returns a random letter from a given string.

Parameters **string** (*str*) – The string to choose random letter from

Returns str The random letter chosen from the string

2.12 py_everything.regex

Source code: py_everything/regex.py

This module containing many usefull regular expressions.

`url`: Regex to match any web URL(irrespective of protocol/URI, subdomain, url path, query parameters)

`filepath`: Regex to match any local filepath(irrespective of OS)

`number`: Regex to match any number(irrespective of length)

`anything`: Regex to match anything

`string`: Regex to match any string(data within quotes(double or single))

`email`: Regex to match any email address

`alphanumeric`: Regex to match any alphanumeric text

`password`: Regex to match complex passwords(password validator)

`ip`: Regex to match any IPv4 Address

`date`: Regex to match any date in format(ddMMYY) with separators of -, ., or /

`time12`: Regex to match any time in format(HH:MM) with optional 0 in HH 12-Hour

`time24`: Regex to match any time in format(HH:MM) with optional 0 in HH 24-Hour

`html`: Regex to match any HTML tag with attributes

`hexCode`: Regex to match any hex code color

`emptyString`: Regex to match any empty string

2.13 py_everything.path

Source code: py_everything/path.py

This module deals with paths, local or web.

New in version 2.1.0.

class py_everything.path.**Path**(*path*)

This class contains all functions related to Path.

The REGEX's used to check the path can also be used, you need to import them.

Parameters **path** (*str*) – String containing full path(web or local)

Raises *error.UnknownPathTypeError* – Raised if path type can't be determined.

getType()

Returns type of file as ``str``('local' or 'web')

Returns str Type of file('local' or 'web')

getRawPath()

Returns raw input path as-is without any modifications.

Returns str Raw Input Path as-is

Raises *error.InvalidOperationPerformedError* – This is raised if the path type is web

getRealPath()

Returns real path based on system type and os.

Returns str Real Path based on system type and operating system

Raises *error.InvalidOperationPerformedError* – This is raised if the path type is web

isFile()

Returns boolean depending on if the path is to a file or a folder.

Returns bool True if the path is to a file

Raises *error.InvalidOperationPerformedError* – This is raised if the path type is web

isDir()

Returns boolean depending on if the path is to a file or a folder.

Returns bool True if the path is to a folder/directory

Raises *error.InvalidOperationPerformedError* – This is raised if the path type is web

getRelativePath()

Returns path relative to `os.curdir`

Returns str Path relative to `os.curdir`

Raises *error.InvalidOperationPerformedError* – This is raised if the path type is web

getLastAccessTime()

Returns last accessed time for file/folder.

Raises *error.InvalidOperationPerformedError* – This is raised if the path type is web

getLastModifiedTime()

Returns last modified time for file/folder.

Raises *error.InvalidOperationPerformedError* – This is raised if the path type is web

openInBrowser()

Opens URL in default browser.

Raises *error.InvalidOperationPerformedError* – This is raised if the path type is local

getRequest()

Returns Requests Response Object.

Returns Response object of get request to URL

Raises *error.InvalidOperationPerformedError* – This is raised if the path type is local

getRequestStatusCode()

Returns status code for get request to URL.

Returns int Status code

Raises *error.InvalidOperationPerformedError* – This is raised if the path type is local

2.14 py_everything.search

Source code: py_everything/search.py

This module deals with search operations, like files, lists, etc.

py_everything.search.**searchFiles**(*keyword*, *path*)

Searches `path` for files matching with `keyword`.

Parameters

- **keyword** (*str*) – Word to match files with
- **path** (*str*) – Full path to directory to search in

Returns list List of matches

py_everything.search.**searchDirs**(*keyword*, *path*)

Searches path for directories matching with keyword.

Parameters

- **keyword** (*str*) – Word to match directories with
- **path** (*str*) – Full path to directory to search in

Returns list List of matches

py_everything.search.**searchExts**(*keyword*, *path*)

Searches path for file extensions matching with keyword.

Parameters

- **keyword** (*str*) – Extension to match file extensions with
- **path** (*str*) – Full path to directory to search in

Returns list List of matches

py_everything.search.**searchList**(*listOfTerms*, *query*, *filter='in'*)

Searches listOfTerms for terms matching with query.

Parameters

- **listOfTerms** (*str*) – List to search in
- **query** (*str*) – Word to verify matches with
- **filter** – Specify way if searching. Choices - 'in', 'start', 'end', 'exact'.

Returns list List of matches

Note: filter is set to 'in' by default. 'in' - Checks if query is in term. 'start' - Checks if term starts with query. 'end' - Checks if term ends with query. 'exact' - Checks if term == query.

2.15 py_everything.sencrypt

Source code: py_everything/sencrypt.py

This module deals with Encryption. Currently only string encryption is supported but file encryption will be supported soon.

py_everything.sencrypt.**genCharKeys**()

This generates 4 character keys and returns the list containing them. These keys are required for encryption.

Returns list List of keys for encryption

py_everything.sencrypt.**genSymKey**()

This generates a symbol key and returns the same. These keys are required for encryption.

Returns str Symbol key for encryption

py_everything.sencrypt.**checkCharKeys**(*keyList*)

Checks if character keys are valid.

Parameters **keyList** – List of keys

Raises *error.InvalidKeyListError* – Raised when keyList contains invalid.

py_everything.sencrypt.**checkSymKey**(*symKey*)

Checks if symbol key is valid.

Parameters **symKey** – Symbol key

Raises *error.InvalidSymbolKeyError* – Raised when symKey is invalid.

class py_everything.sencrypt.**SuperEncrypt**(*keyCharLsit*, *keySym*)

This class creates a SuperEncrypt() object to encrypt and decrypt using keys.

```
>>> from py_everything.sencrypt import SuperEncrypt
>>> import py_everything.sencrypt as se
>>> charKeys = se.genCharKeys()
>>> symbolKey = se.genSymKey()
>>> seObj = SuperEncrypt(charKeys, symbolKey)
>>> text = 'my super secret text'
>>> encrypted = seObj.encrypt(text)
>>> encrypted
'...'
>>> decrypted = seObj.decrypt(encrypted)
>>> decrypted
'my super secret text'
```

Parameters

- **keyCharList** – List of character keys
- **keySym** (*str*) – Symbolkeys

encrypt(*msg*)

Encrypts msg using provided keys.

Parameters **msg** (*str*) – Text to be encrypted.

Returns str Encrypted string.

decrypt(*msg*)

Decrypts msg using provided keys.

Parameters **msg** (*str*) – String to be decrypted.

Returns str Decrypted text.

2.16 py_everything.units

Source code: py_everything/units.py

class py_everything.units.**mg**

Class for milligram unit.

class py_everything.units.**cg**

Class for centigram unit.

class py_everything.units.**dg**

Class for decigram unit.

class py_everything.units.**g**
Class for gram unit.

class py_everything.units.**dag**
Class for dekagram unit.

class py_everything.units.**hg**
Class for hectagram unit.

class py_everything.units.**kg**
Class for kilogram unit.

class py_everything.units.**ml**
Class for millilitre unit.

class py_everything.units.**cl**
Class for centilitre unit.

class py_everything.units.**dl**
Class for decilitre unit.

class py_everything.units.**l**
Class for litre unit.

class py_everything.units.**dal**
Class for dekalitre unit.

class py_everything.units.**hl**
Class for hectalitre unit.

class py_everything.units.**kl**
Class for kilolitre unit.

class py_everything.units.**mm**
Class for millimeter unit.

class py_everything.units.**cm**
Class for centimeter unit.

class py_everything.units.**dm**
Class for decimeter unit.

class py_everything.units.**m**
Class for meter unit.

class py_everything.units.**dam**
Class for dekameter unit.

class py_everything.units.**hm**
Class for hectameter unit.

class py_everything.units.**km**
Class for kilometer unit.

2.17 py_everything.web

Source code: py_everything/web.py

py_everything.web.**googleSearch**(*query*)

Searches Google for query

Parameters **query** (*str*) – Query to search for

py_everything.web.**ytSearch**(*query*)

Searches YouTube for query

Parameters **query** (*str*) – Query to search for

py_everything.web.**githubSearch**(*query*)

Searches GitHub for query

Parameters **query** (*str*) – Query to search for

py_everything.web.**soSearch**(*query*)

Searches StackOverflow for query

Parameters **query** (*str*) – Query to search for

py_everything.web.**amz_inSearch**(*query*)

Searches amazon.in for query

Parameters **query** (*str*) – Query to search for

py_everything.web.**amz_comSearch**(*query*)

Searches amazon.com for query

Parameters **query** (*str*) – Query to search for

py_everything.web.**pypiSearch**(*query*)

Searches PyPI for query

Parameters **query** (*str*) – Query to search for

py_everything.web.**rtdocsSearch**(*query*)

Searches Read The Docs for query

Parameters **query** (*str*) – Query to search for

py_everything.web.**openNewTab**(*query*)

Searches url for query in new tab

Parameters

- **url** (*str*) – URL to search in
- **query** (*str*) – Query to search for

py_everything.web.**openNewWindow**(*url*, *query*)

Searches url for query in new window

Parameters

- **url** (*str*) – URL to search in
- **query** (*str*) – Query to search for

2.18 setupPyGen

2.18.1 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",
   packages=[new, old],
   install_requires=[],
   license="MIT License",
   url="https://github.com/play4Tutorials/py_everything/",
   python_requires='>=3.5'
)
```

2.18.2 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.

2.18.3 -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.

2.18.4 –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.

2.18.5 -t or –tests

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

2.18.6 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/

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

2.18.7 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.

2.19 setupPyGen Changelog

2.19.1 v1.0.1

- Added support for `find_packages()`

2.19.2 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

2.20 gitIt

2.20.1 Basic Usage:

```
$ ls
repo/
$ cd repo/
$ ls -a
. ..
$ gitIt -gh --docs --issue --c --greet
<--Follow the prompts(prompt values - repo-name, description-for-repo, MIT)-->
$ ls -a
. .. .git/ .github/ docs/ README.md LICENSE .gitignore
$ cd .github/

$ ls -a
. .. SECURITY.md workflows/ ISSUE_TEMPLATE/
$ cd workflows/

$ ls -a
. .. greet.yml
$ cd ..

$ cd ISSUE_TEMPLATE/

$ ls -a
. .. bug-report.md feature-or-enhancement-request.md config.yml
$ cd ../..
```

(continues on next page)

(continued from previous page)

```
$ cat README.md
# repo-name

 description-for-repo

 License - MIT
```

2.20.2 Flags:

There are many flags for gitIt. They can be used as per requirements.

- -gh or –github
- -d or –docs
- -s or –security
- -i or –issue
- -c or –config
- –greet

All of these flags are optional. But very few data is taken input after running.

2.20.3 -gh or –github

The -gh or –github flag is used to generate the .github folder in the structure. It is usually combined with the -i or –issue flag for best results.

2.20.4 -d or –docs

The -d or –docs flag generates a docs/ folder in the project structure.

2.20.5 -s or –security

The -s or –security flag is used to generate a SECURITY.md file in .github/ for the security policy. It is prefilled with placeholder data.

2.20.6 -i or –issue

The -i or –issue flag is used to generate issue templates in .github/. Bug report and feature request templates with placeholder data are generated by default. Works when used with -gh or –github flag.

2.20.7 -c or –config

The -c or –config flag is used to generate confg.yml in .github/ISSUE_TEMPLATE/. config.yml is generated with placeholder data. Works when used with -gh and -i flags.

2.20.8 –greet

The –greet flag is used to generate greet.yml in .github/workflows/. greet.yml is generated with data(not placeholder). Works when used with -gh flag.

2.20.9 Note:

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

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

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

If you want to enable a flag just use "-flag", for e.g., - *$ gitIt -t*. Do not specify True or False.

2.21 gitIt Changelog

2.21.1 v1.0.0

- Initial Release
- Generate README.md
- Generate Full git repository structure
- Start git repository
- GitHub Friendly Repository
- Add .gitignore
- Create Basic GitHub Actions Workflow
- Add LICENSE

2.22 py_everything depends on -

- playsound
- pytube

2.23 py_everything is depended upon by -

2.24 License

2.24.1 Need Help?

Need help with anything? Join the GitHub Discussions.

PYTHON MODULE INDEX

p

INDEX

I

K

L

M

N

O

P

R

S

T

U

V

W

Y