JoshAPI

Openic Development

Josh API Documentation

Version 1.2.10

JOSHUA MULIK

8-7-2020

JoshAPI

DISCLAIMER

CONTENTS

This manual documents the API used by C, C++ and Python programmers who want to write extension modules, programs, or projects with JoshAPI. This documentation is a companion to the web doc-string page which documents the API features of JoshAPI but does not describe general principles and does not go into detail on the use of such functions.

This documentation assumes basic knowledge on computers and programming.

INTRODUCTION

This Application Programmer's Interface (API) for Python is designed to give users, programmers and testers an opportunity to enhance their program, decrease the amount of testing required and to make Python easier to learn and program.

Writing in python is a relatively well-understood process, where a 'recipe' or 'cook-book' approach works well. JoshAPI comes with tools to automate the repair and testing of programs.

JoshAPI is hosted publicly on GITHUB at:
https://github.com/JoshyDEWstive/joshapi/

1.1 Coding Standards

JoshAPI assumes and recommends the use of the Python programming standard of PEP 8. These guidelines apply regardless of the version of JoshAPI you are using. Following these conventions is necessary for your own work with and without JoshAPI.

More information on the PEP 8 coding standard can be found here:
https://www.python.org/dev/peps/pep-0008/

1.2 System Requirements

Minimum	**Recommended**
CPU: Single Core *1.0GHz*	CPU: Dual Core 2.0GHz
Memory: *1Gb*	Memory: *4Gb*
Graphics Card: *Not Required*	Graphics Card: *64Mb Memory*
Storage: *100 MB*	Storage: *200 MB*
Operating System:	Operating System:
Windows 7 or later, macOS, or Linux*	*Windows 10 or later, macOS, or Linux*

Note: Hardware requirements also take in account the requirements required to run *Python* 3.8.

To use JoshAPI Python 3.7 is required to be installed on the device.
If your device does not have Python installed it can be installed here:
https://www.python.org/downloads/

Press the yellow download button at the top of the page and follow the required prompts.

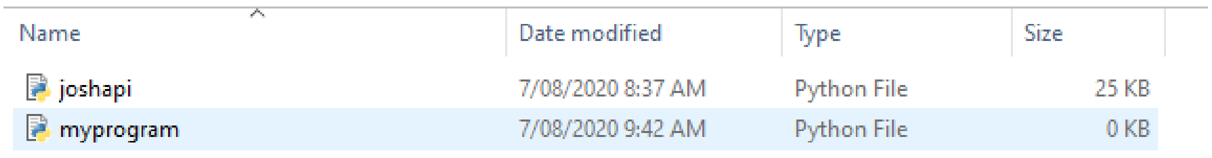
If modules required by JoshAPI are not installed, it will be installed automatically when you first import or use JoshAPI.

1.3 Including and using JoshAPI

Before starting to program with JoshAPI you must ensure that either:
JoshAPI is in the same folder as your Python program
OR
JoshAPI has been installed in your Python running directory.

1.3.1 In the same folder

JoshAPI has to be in the same folder as your Python program to work for it to work IF JoshAPI has not been installed in your Python running directory.



Name	Date modified	Type	Size
joshapi	7/08/2020 8:37 AM	Python File	25 KB
myprogram	7/08/2020 9:42 AM	Python File	0 KB

The below example shows *joshapi.py* in the same folder/directory as our example program *myprogram.py*.

1.3.2 Installing JoshAPI in your running directory

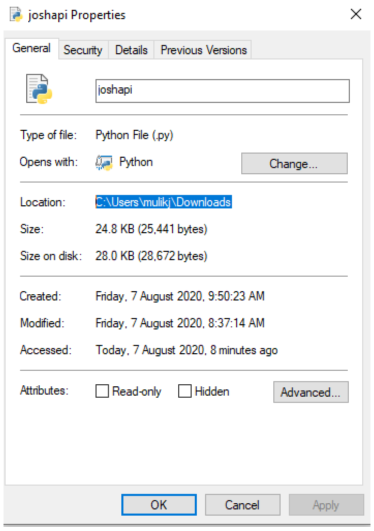
Note: This step will only work in the Windows operating system and will NOT work in Linux or MacOS as of version *1.2.9*.

Installing JoshAPI in your Python running directory will allow it to be used anywhere on your computer without having the need of having the JoshAPI file in the same folder as your program.

To install JoshAPI into your running directory follow the instructions below:

1. Open the folder that you downloaded *joshapi.py* to.
2. Open command prompt by right clicking the start button, pressing ‘Seach’ and typing ‘cmd‘ and clicking the first result that says ‘Command Prompt’
3. Go back to where you downloaded the *joshapi.py* file. Right click it and press ‘Properties’

4. Highlight the path next to 'Location' and press CTRL+C



joshapi Properties

General | Security | Details | Previous Versions

joshapi

Type of file: Python File (.py)
Opens with: Python
Location: C:\Users\mulikj\Downloads
Size: 24.8 KB (25,441 bytes)
Size on disk: 28.0 KB (28,672 bytes)
Created: Friday, 7 August 2020, 9:50:23 AM
Modified: Friday, 7 August 2020, 8:37:14 AM
Accessed: Today, 7 August 2020, 8 minutes ago
Attributes: ☐ Read-only ☐ Hidden

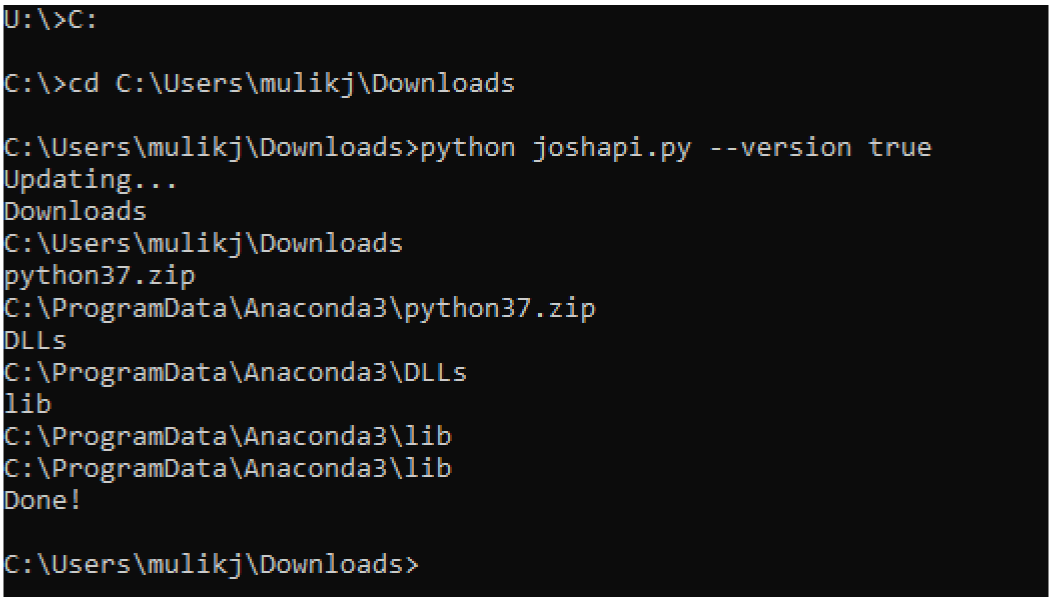
5. Go back into your command prompt and type '*cd* ' (with the space) and paste your path with CTRL+V and press 'Enter'

Note: This will not work if you are not on the same drive as your file. If the path displayed when you first enter into command prompt has a different drive (drives are represented by a letter then a colon, for example: "C:") than your path, type the drive and press 'Enter'.

---

6. Type '*python joshapi.py –version true*' this will install the latest version of JoshAPI into your running directory!
7. Done!

After following the steps listed above your command prompt should look like this:



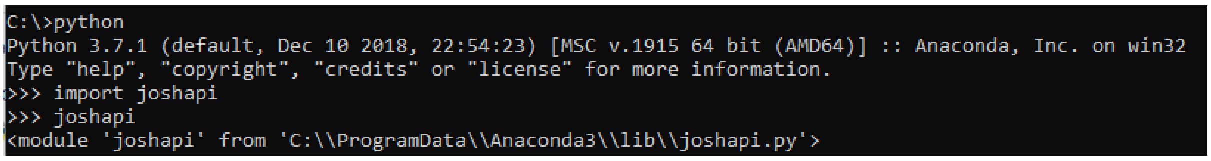
```
U:\>C:

C:\>cd C:\Users\mulikj\Downloads

C:\Users\mulikj\Downloads>python joshapi.py --version true
Updating...
Downloads
C:\Users\mulikj\Downloads
python37.zip
C:\ProgramData\Anaconda3\python37.zip
DLLs
C:\ProgramData\Anaconda3\DLLs
lib
C:\ProgramData\Anaconda3\lib
C:\ProgramData\Anaconda3\lib
Done!

C:\Users\mulikj\Downloads>
```

If JoshAPI has been installed properly the last message sent to the user will be 'Done!'.

To ensure that JoshAPI has been installed correctly you can try install it from another directory (You can't do it in the same directory as you installed it because it will use the file in that folder).

1. In the command prompt terminal, type '*cd C:\*'

2. Then type '*python'* to open the Python shell
3. In the shell, type '*import joshapi'*
4. If python responds with '*<module 'joshapi' from '____'>'* then the installation has worked correctly, if it displays anything otherwise than it has not and you should repeat the steps above.



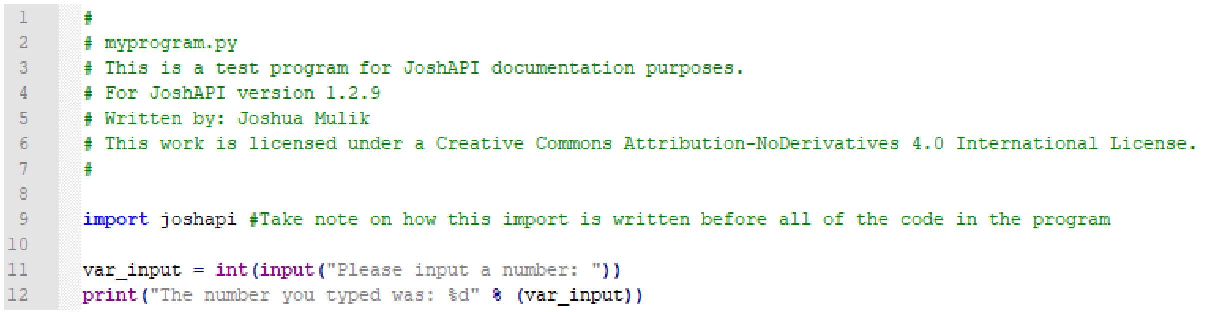
```
C:\>python
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import joshapi
>>> joshapi
<module 'joshapi' from 'C:\\ProgramData\\Anaconda3\\lib\\joshapi.py'>
```

The sample output from the steps above, on this device JoshAPI has been installed correctly.

1.3.3 Including JoshAPI in your program

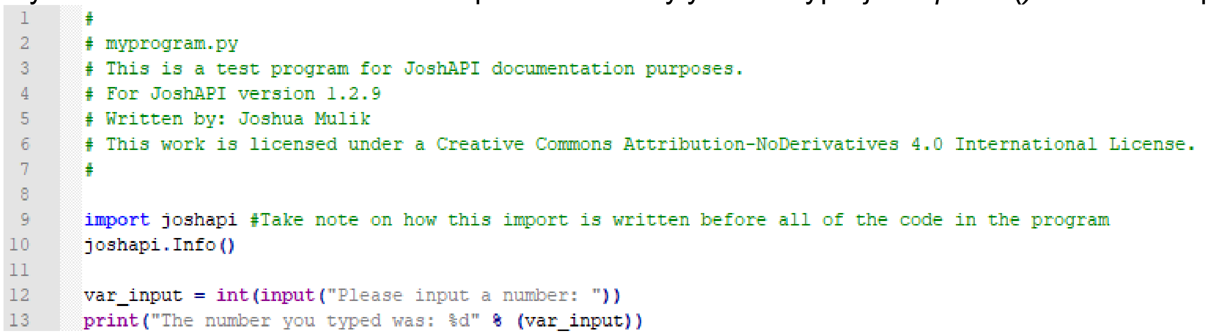
This assumes that you have done step 1.3.1 or 1.3.2, if you have not done those steps, please go back and complete them or this will NOT work.

If you want to use JoshAPI in your program you must include it first. To include it in your program, type '*import joshapi'* at the top of your program, before your code.



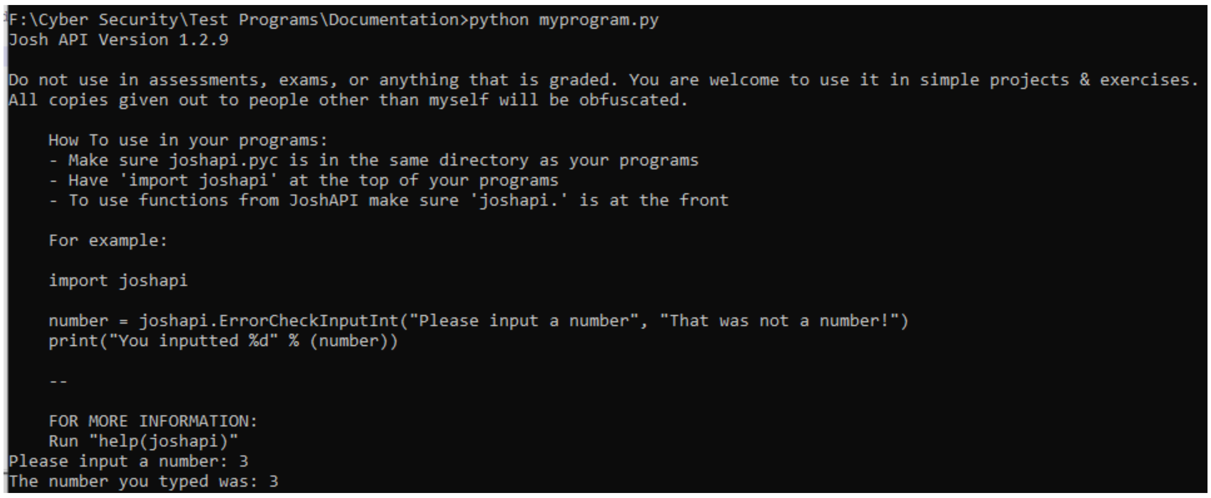
```python
#
# myprogram.py
# This is a test program for JoshAPI documentation purposes.
# For JoshAPI version 1.2.9
# Written by: Joshua Mulik
# This work is licensed under a Creative Commons Attribution-NoDerivatives 4.0 International License.
#

import joshapi #Take note on how this import is written before all of the code in the program

var_input = int(input("Please input a number: "))
print("The number you typed was: %d" % (var_input))
```

If you want to test if JoshAPI was imported correctly you can type '*joshapi.Info()*' after the import.



```python
#
# myprogram.py
# This is a test program for JoshAPI documentation purposes.
# For JoshAPI version 1.2.9
# Written by: Joshua Mulik
# This work is licensed under a Creative Commons Attribution-NoDerivatives 4.0 International License.
#

import joshapi #Take note on how this import is written before all of the code in the program
joshapi.Info()

var_input = int(input("Please input a number: "))
print("The number you typed was: %d" % (var_input))
```

Sample output:



```
F:\Cyber Security\Test Programs\Documentation>python myprogram.py
Josh API Version 1.2.9

Do not use in assessments, exams, or anything that is graded. You are welcome to use it in simple projects & exercises.
All copies given out to people other than myself will be obfuscated.

    How To use in your programs:
    - Make sure joshapi.pyc is in the same directory as your programs
    - Have 'import joshapi' at the top of your programs
    - To use functions from JoshAPI make sure 'joshapi.' is at the front

    For example:

    import joshapi

    number = joshapi.ErrorCheckInputInt("Please input a number", "That was not a number!")
    print("You inputted %d" % (number))

    --

    FOR MORE INFORMATION:
    Run "help(joshapi)"
Please input a number: 3
The number you typed was: 3
```

1.3.4 Using JoshAPI

This assumes that step 1.3.3 has been done, if not go back and ensure they have been completed.

To use the functions and variables provided by JoshAPI you must ensure that '*joshapi.*' is placed before every function and variable.

Correct:

```
joshapi.Info()
```

Incorrect:

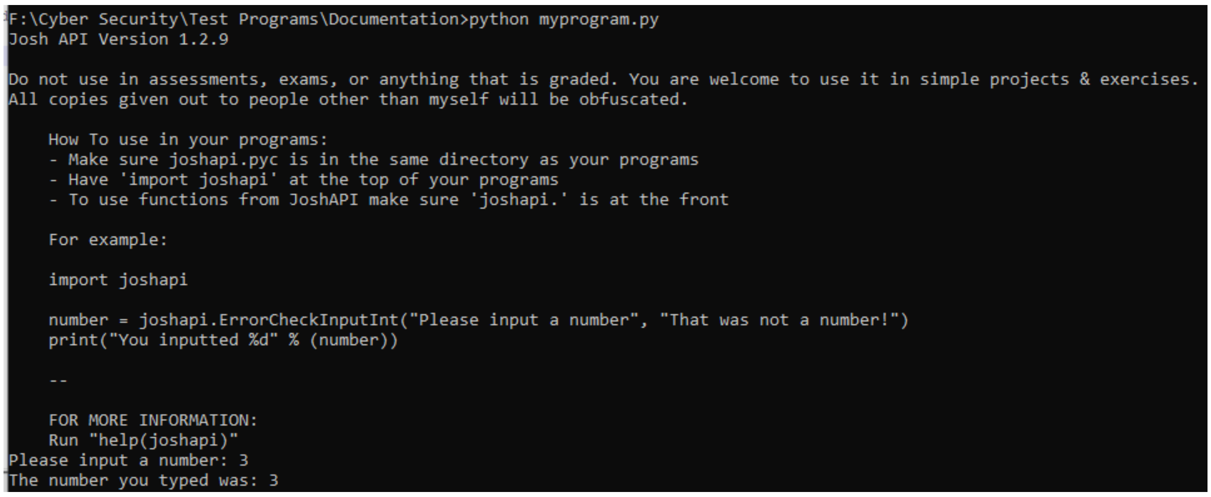
```
Info()
```

If you do not include the '*joshapi.*' section, the program will not know where the function is coming from. If you do not want to include this, and are only using JoshAPI a couple of times in your script you can only import the functions you need.

How to import functions that you need is shown in the script segment below:

```
#
# myprogram.py
# This is a test program for JoshAPI documentation purposes.
# For JoshAPI version 1.2.9
# Written by: Joshua Mulik
# This work is licensed under a Creative Commons Attribution-NoDerivatives 4.0 International License.
#

from joshapi import Info #This will only import the function Info() from JoshAPI
Info()

var_input = int(input("Please input a number: "))
print("The number you typed was: %d" % (var_input))
```

Sample output:



```
F:\Cyber Security\Test Programs\Documentation>python myprogram.py
Josh API Version 1.2.9

Do not use in assessments, exams, or anything that is graded. You are welcome to use it in simple projects & exercises.
All copies given out to people other than myself will be obfuscated.

    How To use in your programs:
    - Make sure joshapi.pyc is in the same directory as your programs
    - Have 'import joshapi' at the top of your programs
    - To use functions from JoshAPI make sure 'joshapi.' is at the front

    For example:

    import joshapi

    number = joshapi.ErrorCheckInputInt("Please input a number", "That was not a number!")
    print("You inputted %d" % (number))

    --

    FOR MORE INFORMATION:
    Run "help(joshapi)"
Please input a number: 3
The number you typed was: 3
```

CORE FUNCTIONS

Everything past this point has assumed that JoshAPI has been installed using the methods listed in section 1.3.

2.0 Function List

Function		**Section (Page)**
.		
A	addToWordList	2.4
B	binaryToDecimal binaryToHex binaryToOctal	2.3.1
C	clear_screen	2.7
D	decimalToBinary decimalToOctal decimalToHex drawArc drawAxis drawCosineWave drawExpoGraph drawQuadratic drawSineWave drawStrings drawTanGrpah	2.3.4 2.5
E	errorCheckInputInt errorCheckInputFloat errorCheckIntRange errorCheckFloatRange errorCheckIntRangeInclusive errorCheckFloatRangeInclusive errorCheckCertainInput	2.1.1 2.1.2

2.1 Input Error Checking

JoshAPI has included a variety of functions that assist the programmer in only getting desired inputs from their users and testers.

2.1.1 Error checking with numbers

JoshAPI provides functions to ensure a user is only inputting numbers. There are two variations of these functions, one if for floats and another is for pure integers. Overall there are six (6) functions that are for error checking with numbers.

There are three main functions:

- errorCheckInputInt
- errorCheckIntRange
- errorCheckIntRangeInclusive

Function	Usage

errorCheckInputInt
Returns Integer
errorCheckInputFloat
Returns Float
 	
Returns an error checked input value and ensures that it is an integer/float.
 	
Keyword arguments:
	`#Usage`
	`errorCheckInputInt(out,err)`
	
	`#Example`
	
	`import joshapi`
	
	`#Integers`
	`input_i = joshapi.errorCheckInputInt(`
	`“Enter a number”,`
	`”That was not a number”)`

Out = Message presented to the user when ran
Err = Message presented to the user when an incorrect input is inputted.
	`#Float`
	`input_f = joshapi.errorCheckInputFloat(`
	`“Enter a number”,`
	`”That was not a number”)`

errorCheckIntRange
Returns Integer
errorCheckFloatRange
Returns Float
 	
Returns an error checked input, it ensures that it is an int/float and it is between the minimum and maximum int/float (Exclusive).
 	
Keyword arguments:
Out = Message presented to the user when ran
Err = Message presented to the user when an incorrect input is inputted.
Min = Minimum number (not inclusive)
Max = Maximum number (not inclusive)
	`#Usage`
	`errorCheckInputIntRange(out,err,min,max)`
	`errorCheckInputFloatRange(out,err,min,max)`
	`#Example`
	
	`import joshapi`
	
	`#Integers`
	`input_i = joshapi.errorCheckInputIntRange(`
	`“Enter a number”,`
	`”That was not a number”,`
	`0,`
	`100)`
	
	`#Float`
	`input_f = joshapi.errorCheckInputFloatRange(`
	`“Enter a number”,`
	`”That was not a number”,`
	`0,`
	`100)`

errorCheckIntRangeInclusive
Returns Integer
errorCheckFloatRangeInclusive
Returns Float
 	
Returns an error checked input, it ensures that it is an int/float and it is between the minimum and maximum int/float (Inclusive).
 	
Keyword arguments:
Out = Message presented to the user when ran
Err = Message presented to the user when an incorrect input is inputted.
Min = Minimum number
Max = Maximum number
	`#Usage`
	`errorCheckInputIntRangeInclusive(out,err,min,max)`
	`errorCheckInputFloatRangeInclusive(out,err,min,max)`
	`#Example`
	
	`import joshapi`
	
	`#Integers`
	`input_i = joshapi.errorCheckInputIntRangeInclusive(`
	`“Enter a number”,`
	`”That was not a number”,`
	`0,`
	`100)`
	
	`#Float`
	`input_f =`
	`joshapi.errorCheckInputFloatRangeInclusive(`
	`“Enter a number”,`
	`”That was not a number”,`
	`0,`
	`100)`

2.1.2 Error checking with certain inputs

This function will only allow a set of inputs from the user, for example it can be used in basic menu screen.

Function	**Usage**

errorCheckCertainInput
Return String
 	
Returns an error checked input value that is in the list of possible inputs.
 	
Keyword arguments:
Out = Message presented to the user when ran
Err = Message presented to the user when an incorrect input is inputted.
Inputs = List of allowed inputs
	`#Usage`
	`errorCheckCertainInput(out,err,inputs)`
	
	`#Example`
	
	`possibleInputs = [`
	`“test”,`
	`“test2”]`
	`input_s = joshapi.errorCheckCertainInput(`
	`“Enter a valid input “,`
	`“Not a valid input”,`
	`possibleInputs)`

select_list
Return string
 	
Displays a selection list to the user, allows the user to use the arrow keys to select an option. ENTER to select, UP and DOWN to move and ESC to cancel/quit.
 	
Keyword arguments:
modifiers = dictionary of strings to display
selectList = list of possible inputs
	`#Usage`
	`select_list(modfiers,selectList)`
	
	`#Example`
	
	`selectList = ["Option1","Option2","Option3"]`
	`modifiers = {"top" : "Select an Option: ",`
	`"bottom" : "Press ESC to escape",`
	`"tab" : "> "}`
	
	`getUserInput =`
	`select_list(modifiers,selectList)`
	
	`print("You picked %s " % (getUserInput))`

2.2 Random Numbers

JoshAPI comes with functions that can automatically generate random numbers for you and eliminate the extra code that you need to do so.

Function	Usage

generateRandomNumber
Returns Integer
 	
Generates a random number with the length of the digits, the seed is based on the current time in milliseconds.
 	
Keyword arguments:
digits = How long the random number will be
	`#Usage`
	`generateRandomNumber(digits)`
	
	`#Example`
	
	`rnum =joshapi.generateRandomNumber(4)`

generateRandomNumberSeed
Returns Integer
 	
Generates a random number with the length of the digits with the seed provided.
 	
Keyword arguments:
digits = How long the random number will be
seed = Seed to use for the random number generator
	`#Usage`
	`generateRandomNumberSeed(digits,seed)`
	
	`#Example`
	
	`rnum = joshapi.generateRandomNumberSeed(4,42)`

generateRandomRangeNumber
Returns Integer
 	
Generates a random number between two digits, the seed is based on the current time.
 	
Keyword arguments:
r1 = Minimum number
r2 = Maximum number
	`#Usage`
	`generateRandomRangeNumber(r1,r2)`
	
	`#Example`
	
	`rnm = joshapi.generateRandomRangeNumber(4,42)`

2.3 Converters

JoshAPI comes with a number of converters for you to use that can convert between all basic computer numbering systems (Binary, Decimal, Octal and Hexadecimal)

2.3.1 Binary

Function	**Usage**

binaryToDecimal
Returns Integer
 	
Converts binary to decimal
 	
Keyword arguments:
Bi = Binary number
	`#Usage`
	`binaryToDecimal(bi)`
	`#Example`
	`number = joshapi.binaryToDecimal("101010")`

binaryToHex
Returns Hexadecimal
 	
Converts binary to hexadecimal
 	
Keyword arguments:
Bi = Binary number
	`#Usage`
	`binaryToHex(bi)`
	`#Example`
	`hexi = joshapi.binaryToHex("101010")`

binaryToOctal
Returns Octal
 	
Converts binary to octal
 	
Keyword arguments:
Bi = Binary number
	`#Usage`
	`binaryToOctal(bi)`
	`#Example`
	`in_oct = joshapi.binaryToOctal("101010")`

2.3.2 Hexadecimal

Function	**Usage**

hexToBinary
Returns Binary
 	
Converts hexadecimal to binary.
 	
Keyword arguments:
he = hexadecimal number
	`#Usage`
	`hexToBinary(he)`
	`#Example`
	`in_bin = joshapi.hexToBinary("2A")`

hexToDecimal
Returns Integer
 	
Converts hexadecimal to decimal.
 	
Keyword arguments:
he = hexadecimal number
	`#Usage`
	`hexToDecimal(he)`
	`#Example`
	`in_dec = joshapi.hexToDecimal ("2A")`

hexToOctal
Returns Octal
 	
Converts hexadecimal to octal.
 	
Keyword arguments:
he = hexadecimal number
	`#Usage`
	`hexToOctal(he)`
	`#Example`
	`in_oct = joshapi.hexToOctal("2A")`

2.3.3 Octal

Function	Usage

octalToBinary
Returns Binary
 	
Converts octal to binary.
 	
Keyword arguments:
Oc = octal number
	`#Usage`
	
	`octalToBinary(oc)`
	
	`#Example`
	
	`in_bin = joshapi.octalToBinary("52")`

octalToDecimal
Returns Integer
 	
Converts octal to decimal.
 	
Keyword arguments:
Oc = octal number
	`#Usage`
	
	`octalToDecimal(oc)`
	
	`#Example`
	
	`in_dec = joshapi.octalToDecimal("52")`

octalToHex
Returns Hexadecimal
 	
Converts octal to hexadecimal.
 	
Keyword arguments:
Oc = octal number
	`#Usage`
	
	`octalToHex(oc)`
	
	`#Example`
	
	`in_hex = joshapi.octalToHex("52")`

2.3.4 Decimal

Function	**Usage**

decimalToBinary
Return Binary
 	
Convert decimal to binary
 	
Keyword arguments:
de = Decimal number
	`#Usage`
	`decimalToBinary(de)`
	`#Example`
	`in_bin = joshapi.decimalToBinary(42)`

decimalToOctal
Return Octal
 	
Convert decimal to octal
 	
Keyword arguments:
de = Decimal number
	`#Usage`
	`decimalToOctal(de)`
	`#Example`
	`in_oct = joshapi.decimalToOctal(42)`

decimalToHex
Return Hexadecimal
 	
Convert decimal to hexadecimal
 	
Keyword arguments:
de = Decimal number
	`#Usage`
	`decimalToHex(de)`
	`#Example`
	`in_hex = joshapi.decimalToHex(42)`

2.4 Word Management

JoshAPI has functionality to create long lists of words that are gathered from websites, some examples of its use can be seen in hangman games.

Function	Usage

addToWordList
Generates a file
 	
Reads a website and generates a file of all the different words in that website.
 	
Keyword arguments:
url = website URL
saveTo = file to save to (do not include '.txt')
	`#Usage`
	
	`addToWordList(url,saveTo)`
	
	`#Examples`
	
	`saveTo = "elephants"`
	`url = "https://en.wikipedia.org/wiki/Elephant"`
	
	`joshapi.addToWordList(url,saveTo)`

generateWordList
Generates a file
 	
Generates a wordlist of approximately 432000 different words based on the top 15 Wikipedia articles[1].
 	
Keyword arguments:
saveTo = file to save to (do not include '.txt')
	`#Usage`
	
	`generateWordList(saveTo)`
	
	`#Examples`
	
	`saveTo = "knowledge"`
	`joshapi.generateWordList(saveTo)`

is_ascii
Returns Boolean
 	
Returns true if the given string is an appropriate ASCII string.
 	
Keyword arguments:
s = string to test
	`#Usage`
	
	`is_ascii(s)`
	
	`#Examples`
	
	`works = joshapi.is_ascii(`
	`"This is an ASCII string"`
	`)`

listToString
Returns String
 	
Turns a list of strings to a large single ASCII list.
 	
Keyword arguments:
li = List in
	`#Usage`
	
	`listToString(li)`
	
	`#Examples`
	
	`li = [`
	`"Lol",`
	`"Next World"`
	`]`
	
	`strings = joshapi.listToString(li)`

2.5 Turtle Functions

JoshAPI also has some basic Turtle functions to draw basic mathematical equations.

Function	Usage

drawArc
 	
Using a turtle it draws an arc
 	
Keyword arguments:
turt = turtle to draw with
x = X value to draw the arc
y = Y value to draw the arc
degrees = how wide the arc is in degrees rotation = rotate the arc
color = arc color
pensize = width of the pen
	`#Usage`
	
	`drawArc(`
	`turt, x, y, degrees, rotation, color, pensize)`
	
	`#Examples`
	
	`import turtle`
	
	`turt = turtle.Turtle()`
	`joshapi.drawArc(turt,0,0,45,60,”green”,10)`

drawAxis
 	
Using a turtle it draws a 2D Cartesian plane.
 	
Keyword arguments:
turt = turtle to draw with
scale = width of the axis
	`#Usage`
	
	`drawAxis(turt, scale)`
	
	`#Examples`
	
	`import turtle`
	
	`turt = turtle.Turtle()`
	`joshapi.drawAxis(turt,100)`

drawCosineWave
 	
Using a turtle it draws a cosine graph.
Formula: y = a * cos(bx) + c
 	
Keyword arguments:
turt = turtle to draw with
a = a on the formula
b = b on the formula
c = c on the formula
r = range of the graph (100 is from -100 X to 100 X)
color = graph color
accuracy = scale to draw by (every 1 X, every 0.5 x etc)
	`#Usage`
	
	`drawCosineWave(`
	`turt,a,b,c,r,colour,accuracy,scale`
	`)`
	
	`#Examples`
	
	`import turtle`
	
	`turt = turtle.Turtle()`
	`joshapi.drawCosineWave(`
	`turt,1,2,3,25,”red”,1`
	`)`

drawExpoGraph
 	
Using a turtle it draws an exponential curve/s
Formula: y = a^(bx)+c
 	
Keyword arguments:
turt = turtle to draw with
	`#Usage`
	
	`drawExpoGraph`
	`(`
	`turt, a, b, c, r, colour, accuracy, scale`
	`)`

a = a on the formula
b = b on the formula
c = c on the formula
r = range of the graph (100 is from -100 X to 100 X)
colour = graph colour
accuracy – What scale to draw by (every 1 X, every 0.5 x etc)
	`#Examples`
	
	`import turtle`
	
	`turt = turtle.Turtle()`
	`joshapi.drawExpoGraph(`
	`turt,1,2,3, 25,”red”,1`
	`)`

drawQuadratic
 	
Using a turtle it draws a quadratic graph.
Formula: y = ax^2 + bx + c
 	
Keyword arguments:
turt = turtle to draw with
a = a on the formula
b = b on the formula
c = c on the formula
r = range of the graph (100 is from -100 X to 100 X)
colour = graph colour
accuracy = scale to draw by (every 1 X, every 0.5 x etc)
	`#Usage`
	
	`drawQuadratic`
	`(`
	`turt, a, b, c, r, colour, accuracy, scale`
	`)`
	
	`#Examples`
	
	`import turtle`
	
	`turt = turtle.Turtle()`
	`joshapi.drawQuadratic(`
	`turt,1,2,3, 25,”red”,1`
	`)`

drawSineWave
 	
Using a turtle it draws a sine graph.
Formula: y = a * sin(bx) + c
 	
Keyword arguments:
turt = turtle to draw with
a = a on the formula
b = b on the formula
c = c on the formula
r = range of the graph (100 is from -100 X to 100 X)
color = graph color
accuracy = scale to draw by (every 1 X, every 0.5 x etc)
	`#Usage`
	
	`drawSineWave(`
	`turt,a,b,c,r,colour,accuracy,scale`
	`)`
	
	`#Examples`
	
	`import turtle`
	
	`turt = turtle.Turtle()`
	`joshapi.drawSineWave(`
	`turt,1,2,3,25,”red”,1`
	`)`

drawStrings
 	
Using a turtle it draws a list of strings, each on a new line relative to the font size.
 	
Keyword arguments:
turt = turtle to draw with
startX = draw at X pos
startY = draw at Y pos
strings = list of strings to display
	`#Usage`
	
	`drawStrings`
	`(`
	`turt, startX, startY, strings, color,`
	`fontSize`
	`)`
	`#Examples`
	
	`import turtle`

colour = string colour
fontSize = fontsize of strings
	`turt = turtle.Turtle()`
	`strings = [`
	`“lol”,`
	`“new line”`
	`]`
	`joshapi.drawStrings(`
	`turt,0,0,strings,”black”,2`
	`)`

drawTanGraph
 	
Using a turtle it draws a tan graph.
Formula: y = a * tan(bx) + c
 	
Keyword arguments:
turt = turtle to draw with
a = a on the formula
b = b on the formula
c = c on the formula
r = range of the graph (100 is from -100 X to 100 X)
color = graph color
accuracy = scale to draw by (every 1 X, every 0.5 x etc)
	`#Usage`
	
	`drawTanGraph(`
	`turt,a,b,c,r,colour,accuracy,scale`
	`)`
	
	`#Examples`
	
	`import turtle`
	
	`turt = turtle.Turtle()`
	`joshapi.drawTanGraph(`
	`turt,1,2,3,25,”red”,1`
	`)`

JoshAPI

2.6 Encryption and Security

Function	Usage

makeCaesarCipher
Returns String
Generates a Caesar cipher
 	
Keyword arguments:
inString – String to encrypt
count – Encryption distance
	`#Usage`
	
	`makeCaesarCipher(inString,count)`
	
	`#Examples`
	
	`enc = joshapi.makeCaesarCipher("EncryptMe",2)`
	`print(enc)`

makeCaesarCipherQuadratic
Returns String
Generates a Caesar cipher using a quadratic formula
Y=3X^2 + 2X + 6
 	
Keyword arguments:
inString – String to encrypt
count – Encryption distance
	`#Usage`
	
	`makeCaesarCipher(inString,count)`
	
	`#Examples`
	
	`enc = joshapi.makeCaesarCipherQuadratic(`
	`"EncryptMe",2`
	`)`
	`print(enc)`

2.7 Other Functions

Function	Usage

Info
 	
Provides the version and basic information for the JoshAPI that is installed on the machine.
 	
Function Name Variations
info
	`#Usage`
	
	`Info()`
	`#or`
	`info()`

updateJoshAPI
 	
Installs the latest online public version of JoshAPI onto your machine.
	`#Usage`
	
	`updateJoshAPI()`

clear_screen
 	
Clears the current command prompt screen, works in Linux, Mac and Windows.
	`#Usage`
	
	`select_list()`

sl_on_press
 	
Private utlitiy function for select_list(). Listens for keystrokes and adapts table to demonstrate that change.
 	
Keyword arguments:
key - key that has been inputted
	`#Usage`
	`#For use with listeners, not meant to be used by itself`
	`sl_on_press(key)`

EXAMPLES

3.1 Simple Number Guessing Game

Note: Method categories used in this category are:
- Random Numbers (2.2)
- Input Error Checking (2.1)

The aim of this game is to guess a number between 1 and 100. If you guess the number correctly you win, the game tells you if the game is too big or too small (compared to the target number). If you type 0 the game escapes.

JoshAPI is used in this game to generate a random number between 1 and 100 and to ensure that the user input is an Integer between 0 and 100.

Sample Play 1 (Correct play):

```
C:]> python ngg.py
Enter an Integer between 0 and 100: 50
Guess is too small!
Enter an Integer between 0 and 100: 75
Guess is too big
Enter an Integer between 0 and 100: 60
Guess is too big
Enter an Integer between 0 and 100: 55
Woohoo you won!
It took you 4 turns
```

Sample Play 2 (Game exit):

```
C:\> python ngg.py
Enter an Integer between 0 and 100: 50
Guess is too small!
Enter an Integer between 0 and 100: 0
You exited the game.
```

Sample Play 3 (Incorrect Inputs):

```
python ngg.py
Enter an Integer between 0 and 100: -1
That was not a number between 0 and 100, try again!
Enter an Integer between 0 and 100: a
That was not a number between 0 and 100, try again!
Enter an Integer between 0 and 100: 2.0
That was not a number between 0 and 100, try again!
Enter an Integer between 0 and 100: 0
You exited the game.
```

Source:

```
import joshapi

#Get a random number between 1 and 100
random_number = joshapi.generateRandomRangeNumber(1,100)

#How many tries the user has had
user_tries = 0

#Error check the users input to ensure it is an integer between 0 and 100
def getInput():
    user_input = joshapi.errorCheckIntRangeInclusive(
        "Enter an Integer between 0 and 100: ",
        "That was not a number between 0 and 100, try again!",
        0,
        100
        )
    return user_input

while(True):
    user_input = getInput()

    if(user_input == 0):
        print("You exited the game.")
        break
    user_tries += 1
    if(user_input > random_number):
        print("Guess is too big")
    elif(user_input < random_number):
        print("Guess is too small!")
    elif(user_input == random_number):
        print("Woohoo you won!")
        print("It took you %d turns" % (user_tries))
        break
```

CHANGELOG

Version 1.2.10

Legend

+ Added
= Error/Bug Fix
- Removed
! Note or Information
/ To be added in next version
| Change

Fixes

= errorCheckIntRangeInclusive updated to use renamed functions
= errorCheckIntRange updated to use renamed functions
= errorCheckFloatRangeInclusive updated to use renamed functions
= errorCheckFloatRange updated to use renamed functions
= Extra whitespace removed
= Version update message now displays correct message
= When installing new python packages they will be imported without having to restart the program

Changes

| Previous version of JoshAPI is saved before the new version is installed.

Additions

+ Added a selection list function (select_list())

FOOTNOTES

1. generateWordList(saveTo)’s words come from the following free to access and secure websites:

- https://en.wikipedia.org/wiki/Spanish_colonization_of_the_Americas
- https://en.wikipedia.org/wiki/Pet%C3%A9n
- https://en.wikipedia.org/wiki/Guatemala
- https://en.wikipedia.org/wiki/Maya_civilization
- https://en.wikipedia.org/wiki/Nojpet%C3%A9n
- https://en.wikipedia.org/wiki/Air_raids_on_Japan
- https://en.wikipedia.org/wiki/Operation_Matterhorn
- https://en.wikipedia.org/wiki/Michael_Jackson
- https://en.wikipedia.org/wiki/Byzantine_navy
- https://en.wikipedia.org/wiki/Pope_Pius_XII
- https://en.wikipedia.org/wiki/Military_history_of_Puerto_Rico
- https://en.wikipedia.org/wiki/History_of_Poland_(1945%E2%80%9389)
- https://en.wikipedia.org/wiki/Manhattan_Project
- https://en.wikipedia.org/wiki/Elvis_Presley'