Pysha

Release 0.1.2

Arshiatmi

Sep 03, 2022

CONTENTS:

CHAPTER

ONE

PYSHA

1.1 auth module

class auth.**Auth**(*first_prompt='Enter Your Username : '*, *second_prompt='Enter Your Password : '*, *mode=0*, *mask='*'*)

Bases: object

Authentication Class * Mode 0 -> Not Show Password * Mode 1 -> Star Password * Mode 2 -> Normal Input

auth(*c=True*, *mask_color=\x1b[39m'*, *inp_color=\x1b[39m'*, *mask='*'*)

Argumets : * c -> If Its True colorize Function Will Be Executed On Prompts. * mask_color -> Set Color For Password Mask. * inp_color -> Set Color For Username Mask. * mask -> Set Mask Character.

1.2 classes module

class classes.**Case**(*tar*)

Bases: object

Define Case Of Switch-Case. Just In Case You Want To Use It Like Case(x).

class classes.**Cond**(*condition*)

Bases: object

Thats Just Normal One line Conditions That Exists in C++. (Inspired From C++) You Can Use It Like : * Cond(condition)(if_its_true, if_its_false) For Example : * Cond(i > j)(i,j) Or Cool Things Like : * Cond(i > j)(minus,add)(i,j) Thats Your Creativity That Makes Your Application Cool :)

class classes.**CrossPlatformer**(*supported_os=['linux', 'windows', 'mac']*)

Bases: object

This Class Will Make An Environment For You To Have Easier And Better Experience :)

add_os_commands(*name*, *commands: dict*)

Set Command For User Os. Arguments Are : * name : Name Of Command * commands : Dictionary Of Commands For Example : * add_os_commands("git", {"linux": "git", "windows": "git.exe"}) Or : * add_os_commands("clear", {"linux": "clear", "windows": "cls", "mac": "clear"})

get_os_command(*name*)

Get Command For User Os. Arguments : * name : Command Name For Example :

If You Set add_os_commands("clear", {"linux": "clear", "windows": "cls", "mac": "clear"}) : * get_os_command("clear")

Will Return You The Command Depends On Your Os.

get_os_name()

Get User Os Name

get_os_version()

Get User Os Version

is_os_supported(*os_name*)

Check If An Os Is Supported

class classes.**Default**

Bases: object

Default Case Of Switch-Case. Its Not Doing Anything But You Can Use It On Your Switch Cases.

class classes.**Loop**(**args*)

Bases: object

This Option Is Unique Too Pysha. Its Not From Anywhere Else But Anyway I Think Its Not That Cool :/ You Can Use It Like : `Loop(x,y)("Enter Number[_1_][_2_] :",Loop.Modes.Input) # Input Mode Is Default`

Modes : * Input = 0 * Print = 1

Input Is Where You Want To Get Input Just Not Print The Text In x * y Times. It Will Return x * y Array Contains Inputs. Print Is Where You Want To Just Print The Text In x * y Times.

class Modes(*value*)

Bases: Enum

An enumeration.

Input = 0

Print = 1

class classes.**PercentPrinter**(*chars=100*, *pass_color='\x1b[39m'*, *loading_color='\x1b[39m'*)

Bases: object

Pretty Percent Printer For Loading Some Progress Or Downloading :) In Fact, Its Progressbar Itself :)

config(*char_ok*, *char_loading*)

Set char_ok And char_loading For Progressbar.

finish(*show=True*)

Finishes The Progressbar.

increase(*p=1*, *show=True*)

Increses Progressbar In Percents That You Set.

property percent

show(*char_ok='#'*, *char_loading='-'*, *end='\n'*)

This Function Just Shows The Progressbar.

class classes.**Switch**(*var*)

Bases: object

The Famous Switch-Case :)

case(*tar*, *func*, *args=()*)

cases(*all_funcs*)

This Function Should Get A Dictionary That Key Is Case And Value Is Function That Will Run On That Case. You Can Pass Lots Of Types For Example : ```

Case(5):

[
func, arguments, kwargs

]

` Or : `

Case(1):
" print("yay") "

` Or : `

Case(6):

lambda x,y:
print(x,y)

``` For More Models You Can See Examples.

**class** classes.**Vars**

Bases: object

In Case You Want To Define Some Variables In lambdas, You Can Set Attributes Of This Class.

**class** classes.**command**

Bases: object

This Class Will Be Use For Fun Part Of This Framework :)

**condition**(*check*, *p=True*, *\*\*kwargs*)

This Function Act As c++ One Line Condition. You Can Use This Function Like :

*condition('i > j ? "i Is Grater" : "j Is Grater"',i=20,j=23)*

Output Will Be :

*"j Is Grater"*

Notes & Warnings :

*-> '(Quotes) Are Not Working, Always Use "(Double Quotes)*

**exe**(*string*, *priority=0*, *strip_response=True*)

In Progress But This Function Should Get Text Between () Of $(cmd) And Run It In Terminal Then Replace The Output With $(cmd) And Should Get #(cmd) Too And Run It As Python Code And Replace This Output Too.

**Problem Now :**
Use This 2 Thing Together.

Args :

- string -> The String Pattern That You Want To Make.
- **priority -> If Its 0 First System Commands Will Replace Then**
  Python Commands Will Replace. And If Its 1 First Python Commands Will Replace Then System Commands Will Be Replaced. Default Is 0.
```

- **strip_response -> Default Is True. It Wanted To Know That Your**
  String That Replaced Should Be striped ? (Remove Addtion New Lines And Spaces In Start And End Of String ?)

Example :

- exe("You Current Directory :

  $(dir)

Finished :)")

loop(*cmd*, *mode='p'*, *c=True*)

2 Types Of Loop Currently Exists. i(Input) and p(Print) Mode. In print mode you can make a print loop like that : `<i:3>{You Are In _i_ Loop}` And This Will Be Printed Like That : `You Are In 0 Loop` `You Are In 1 Loop` `You Are In 2 Loop`

And In Input Mode You Can Do This : `<i:3>{Enter _i_ Number : }` It Will Get 3 Input Like This : `Enter 0 Number : (input0)` `Enter 1 Number : (input1)` `Enter 2 Number : (input2)` And Will Return *[input0,input1,input2]*

More Options : `<i:3>{Enter _i++_ Number : }` `<i:3>{Enter _i--_ Number : }` `<i:2,j:2>{Enter Row _i++_ And Column _j++_ Number : }` `<i:2>{Enter (Fore.CYAN)[_i++_] Number : }`

Notes : * Just Maded Maximum For 2 Variables Not More.

property mode

1.3 colors module

colors.**back** = {'black': '\x1b[40m', 'blue': '\x1b[44m', 'cyan': '\x1b[46m', 'green': '\x1b[42m', 'light_black': '\x1b[100m', 'light_blue': '\x1b[104m', 'light_cyan': '\x1b[106m', 'light_green': '\x1b[102m', 'light_magenta': '\x1b[105m', 'light_red': '\x1b[101m', 'light_white': '\x1b[107m', 'light_yellow': '\x1b[103m', 'magenta': '\x1b[45m', 'red': '\x1b[41m', 'reset': '\x1b[49m', 'white': '\x1b[47m', 'yellow': '\x1b[43m'}

This Is back Dictionary That Contains Lots Of Colors That You Can Use For Background Of Text's Color!

colors.**fore** = {'black': '\x1b[30m', 'blue': '\x1b[34m', 'cyan': '\x1b[36m', 'green': '\x1b[32m', 'light_black': '\x1b[90m', 'light_blue': '\x1b[94m', 'light_cyan': '\x1b[96m', 'light_green': '\x1b[92m', 'light_magenta': '\x1b[95m', 'light_red': '\x1b[91m', 'light_white': '\x1b[97m', 'light_yellow': '\x1b[93m', 'magenta': '\x1b[35m', 'red': '\x1b[31m', 'reset': '\x1b[39m', 'white': '\x1b[37m', 'yellow': '\x1b[33m'}

This Is fore Dictionary That Contains Lots Of Colors That You Can Use For Text Color!

1.4 data_structures module

class data_structures.**FIFO**(*inf=1*, *capacity=0*)

Bases: *Queue*

You Can Use This Insted Of Queue. All Methods Of Queue Are Available Here.

class data_structures.**LIFO**(*inf=1*, *capacity=0*)

Bases: *Stack*

You Can Use This Insted Of Stack. All Methods Of Stack Are Available Here.

class data_structures.**PyshaDict**

Bases: `dict`

Thats Just Normal Dictionary Just With 2 Or 3 More Option. PyshaDict Operators : * ~myDict # Returns Dictionary Inverse * myDict - "elem" # Returns Dicionary With Removed "elem" Key.

property inverse

You Can Get inverse Of A Dictionary. For Example Dict -> *{"nickname":"MrMegaExpert"}*, Inverse -> *{"MrMegaExpert":"nickname"}* You Can Set And Change Dicitonary In Inverse Mode ! You Can Get Inverse Of A Dictionary In 2 Ways : * myDict.inverse * ~myDict

class data_structures.**PyshaList**(*iterable=()*, */*)

Bases: `list`

PyshaList That Have Some Advanteges To Normal List.

count_deep(*tar*)

Deep Count. If You Pass A list Of String, It Will Check If The tar (Argument That Passed To This Function) Is Inside The list itself or is inside the elements of list or not. If you pass a list with complex dimentions it will count inside of all of elements even insidest element ! and it will count it for you.

lshift(*times*)

Left Shift In Times That You Specify

static make_array(*dimentions*)

Make Array In Complex Dimentions. You Can Use This Method Like : `` `PyshaList.make_array([1,2,4,5,4,32])` ``

rshift(*times*)

Right Shift In Times That You Specify

set_array(*dimentions*, *value*)

Set Array In Complex Dimentions. You Can Use This Method Like : `` ` array = PyshaList.make_array([1,2,4,5,4,32]) # Just A Complex Array array.set_array([0,2,4,0,4,3], "test") ` `` Or Like This : `` `array[0,2,4,0,4,3] = "test"` `` Cool, Right ? :)

shift_index(*index*, *left_shift=True*, *how_many=1*)

You Can Shift Your Array Better That Just lshift And rshift Methods. You Can Define Which Index To Shift, Type Of Shift (left or right) And how_many To Shift.

Arguments : * index, # Index That You Want To Shift * left_shift=True, # Left Or Right Shift That You Can Specifiy. Default Is Left Shift * how_many=1 # How Many Shifts That You Want. Default Is One

class data_structures.**PyshaString**

Bases: `str`

In PyshaString You Have Lots Of Methods To Do Some Things.

exists(*data_to_search: list*, *line_start=-1*, *line_end=-1*, *start_index=-1*, *end_index=-1*)

This Function Will Check If A List Of Strings Exists In Another String. In Specific Part Of It Or Not. It Will Get a Part Of String And Will Check It With A List Of Strings.

Args : * data_to_replace -> A List Of Data That You Want To Check If Exists In Another String.

- line_start -> Line Start Number (Starts With 0)
- line_end -> Line End Number (Starts With 0)
- start_index -> Start Index Number (Starts With 0)
- end_index -> End Index Number (Starts With 0)

find_first(*data_to_search: list*, *return_type=<class 'int'>*)

In This Function You Can Search That an String Contains Some Other String From A List.

Args : * data_to_search -> Strings That You Want To Search For. For Example

Search In List : [" ", ":", ","] And If Your String Is "Hello, Im Arshia" It will Return 5 Thats Index Of ",". If Your String Is "Arshia:Hey" It Will Return 6 Thats Index Of ":".

- **return_type -> You Can pass int To Return Index Of Found String. Or Pass Other**
  Things Like str To Return Found String.

find_last(*data_to_search: list*, *return_type=<class 'int'>*)

This Function Is Exactly find_first But It Will Return Last Index Of Found String.

Args : * data_to_search -> Strings That You Want To Search For. For Example

Search In List : [" ", ":", ","] And If Your String Is "Hello, Im Arshia" It will Return 10 Thats Index Of " ". If Your String Is "Arshia:Hey" It Will Return 6 Thats Index Of ":".

- **return_type -> You Can pass int To Return Index Of Found String. Or Pass Other**
  Things Like str To Return Found String.

inside(*data_to_search: ~data_structures.ListOrDict*, *search_mode='key'*, *return_type=<class 'bool'>*)

In This Function You Can Search That If A String Exists In Dict (Or Inside Of All Elements). Or If A String Exists In List And All Of Its Elements.

Args : * data_to_search -> Data That Wanted To Search In That. For Example

Search In List : ["test", "some", "Hello World"] Or Search In dict : {"test": "Hello World", "some": "Hello World"} -> (search_mode must be "value" or "both" in dict input) if string is Hello It Will Return True !

- **search_mode -> If You Passed A Dictionary You Can Choose Between "key" Or "value"**
  Or "all"/"both" To Search In Both Of Them. It Does Not Do Anything If You Passed A List.
- **return_type -> If You Passed A Dictionary You Can Choose Between bool and dict or ...**
  if you pass bool, it will return True or False If It Finds Your String. if you pass dict, it will return a dict with key and value of found element. if you pass pass anything else, it will return oposite of search_mode. For Example If You passed "key" And You Found Your String, It Will Return "value".

replace_array(*data_to_replace: list*, *replace_with: ListOrString*)

In This Function You Can Replace An Array With Another Array ! (in a string). Its Possible That You Replace An Array Of String With A Single String Too.

Args : * data_to_replace -> Data That You Wanted To Be Replaced For Example If Data Is

["Hello", "World"] And replace_with Argument Is ["Hi", "There"] All Of "Hello" And "World" Will Be Replaced With "Hi" And "There".

- replace_with -> Described In data_to_replace Argument.

replace_dict(*data_to_replace: dict*)

This Is replace_array But You Can Pass Dictionary To Replace With.

Args : * data_to_replace -> A Dictionary Of Data To Replace. For Example

{"Hello": "Hi", "World": "There"} Will Replace "Hello" With "Hi" And "World" With "There". (All Of String)

replace_with_escape(*string_to_find*, *to_replace*, *escape_char=\\'*, *remove_escapes=False*)

This Function Will Replace Any String That You Want But You Can Define Some Parts Not To Replace With Define Escapes !

Args : * string_to_find -> String That You Want To Search For In String.

- to_replace -> String That You Want To Be Replaced With string_to_find.
- escape_char -> Escape Character That You Want To Define. (Default Is Character)
- **remove_escapes -> You Can Define That After Replacing, Escape Characters Should Remove** Or Not. (Default Is False)

class data_structures.**Queue**(*inf=1*, *capacity=0*)

Bases: object

The Common Queue Data Structure That You Can Use Here.

add(*elem*)

Add Value To Queue.

display()

Display The Queue.

remove()

Remove From Queue (Returns The Removed Value)

property storage

class data_structures.**Stack**(*inf=1*, *capacity=0*)

Bases: object

The Common Stack Data Structure That You Can Use Here.

display()

Display Stack Data

pop()

Pop The Value From Stack (Returns The Removed Value)

push(*elem*)

Push The Value To Stack

property storage

1.5 decorators module

decorators.**ignore**(*exceptions='*'*)

This Function Acts As A Decorator For Ignoring Some Exceptions That You Define. For Example You Can Set That If Running A Function Have ZeroDivisionError, Function Ignore The Problem And Not Make Exception.

Agrs : * exceptions That Should Be A List Of Exceptions You Want To Ignore Or '*' Character * That Means All Of Exceptions Must Ignore.

class decorators.**interface**(*target_class*)

Bases: object

You Can Make Interfaces (Inspired From PHP !) . In Fact You Can Define An Structure That All Of Classes That Extends That Interface Must Have This Structure That You Defined.

You Can Use It Like Decorator. For Example :

``` @interface class UserStructure:

name = None family = None username = None password = None

@interface(UserStructure) class SpecialUsers:

name = None family = None username = None password = None

@interface(UserStructure) class Users:

name = None family = None username = None password = None

```

has_attr(*attr*)

Checks If This Class Has Specific Attribute Or Not. (Checks For Every Attribute Like Methods,Variables,...)

instances: dict = {}

is_allowed_structure()

Checks If Has Allowed Structure Or Not.

decorators.**once**(*func*)

This Function Acts As A Decorator For Process Initialization. Every Function That Has This Decorator Will Be Called Before The Main Function Once. Important Thing Is That Function Must Not Have Any Arguments Or Kwarguments To Be Called.

Important Thing : * This Function Will Be Called Before The Main Function. * Function Must Not Have Any Arguments Or Kwarguments To Be Called.

decorators.**retry**(*count*, *exceptions='*'*)

This Function Acts As A Decorator For Retring On Some Exceptions That You Define. For Example You Can Set That If Running A Function Have HTTPConnectionError, Function Retries And Run Again. You Can Define Count Of Retries Too.

Agrs : * count : Count Of Retries For Target Function * exceptions : a list contains all exceptions that should retry on them or '*' character

means all of exceptions should be retried.

decorators.**timer**(*string*)

This Function Acts As A Decorator For Check Spent Time On Executing Functions. Args: Gets One Argument For Printing Format. For Example : ```

@time(“Spent Time Was _T_”) def test():

return True

``` And Output Of This Code After Calling test() Will Be Something Like That :

`Spent Time Was 0:00:00.100404`

Important Parameters : * _T_ : Will Be Replaced With [Hour]:[Minute]:[Second].[MiliSecond] * _H_ : Will Be Replaced With [Hour] * _M_ : Will Be Replaced With [Minute] * _S_ : Will Be Replaced With [Second] * _MS_ : Will Be Replaced With [MiliSecond]

Important : **You Can Escape All Of This Parameters With ()**

## 1.6 enums module

**class** enums.**Algorithms**(*value*)

Bases: Enum

The Algorythms That You Can Make Encryption Chain From.

**Base16 = 5**

**Base32 = 4**

**Base64 = 1**

**Base85 = 3**

**Cipher = 2**

**ROT13 = 6**

**XOR = 0**

## 1.7 exceptions module

**exception** exceptions.**AlgorithmError**

Bases: Exception

**exception** exceptions.**ConditionError**

Bases: Exception

**exception** exceptions.**LoopError**

Bases: Exception

**exception** exceptions.**ModeError**

Bases: Exception

**exception** exceptions.**QueueError**

Bases: Exception

**exception** exceptions.**SecurityError**

Bases: Exception
```

exception exceptions.**StackError**

Bases: Exception

exceptions.**exception_create**(*name*)

Create Your Customized Exception With Name That You Specify !

1.8 functions module

functions.**append_file**(*file_name: str*, *text: str*) → bool

You Can Append to A File.

Args : * file_name * text Returns True If Its Successfully Done, Else Return False

functions.**between**(*string*, *string_to_check*, *start_sign=''*, *end_sign=''*, *exact=False*, *case_sensetive=True*)

Checks If An String Exists Between Two Characters In Another String.

Args : * string -> A Simple String. (We Need To Check If This String Exists

In Another String Or Not) For Example If You Want To Check If 'hello' is between '' Characters, You Need To Pass hello As This Argument.

- **string_to_check -> A Big String That We Want To Check And Dive In :)**
  For Example 'hello My Name Is Arshia And "hello" Is ok Now As i think :)' And Pass It As This Argument.
- **start_sign -> This Should Be The Start Character That You Are Looking For.**
  For Example You Can Pass '' As This Argument (Continue Of Previous Part Example) . But In Another Example That You Want To Check A Tag For Example html Tag You Can Pass "<" As Start.
- **end_sign -> This Should Be The Start Character That You Are Looking For.**
  For Example You Can Pass '' As This Argument (Continue Of Previous Part Example) . Continue Of Tag Example You Can Pass ">" As End.
- **exact -> Basically, Checks For <html> Not <*html*>.This Argument Will Check**
  For exact Value. For Example If This Option Was False And You Want To Check For Tags In String "This <html Test Fake Text> Is Ok." It Will Return True But If This Option Is True,For That String Will Return False. Because It Looks For Exact "<html>" Not < Then Anything Then html Then Again Anything And > Character.
- **case_sensetive -> If Its True, Lower Case Or Upper Case In Words Are Important.**
  For Example If Its True, "Html" Is Different From "html" But If Its False, "Html" And "html" Are Equal.

Example : 1. between(":",'The : Is Not Between "" Chars.','') Returns : 1. False 2. between(":",'The ":" Is Between "" Chars !','') Returns : 2. True

Returns True If Its Found In target Text Else, Returns False.

functions.**between_index**(*string*, *string_to_check*, *start_sign=''*, *end_sign=''*, *exact=False*, *case_sensetive=True*, *start=0*)

Acts Like Between Function But It Will Return Index Insted Of Boolean Value.

functions.**colorize**(*string: str*) → str

It Will Colorize The Input. Example : `colorize("(Fore.GREEN)[Hello !]")` Output : * Green Foreground -> **|Hello !|**

Possible Colors: ``` fore = {

"green" : Fore.GREEN, "black":Fore.BLACK, "red":Fore.RED, "yellow":Fore.YELLOW, "blue":Fore.BLUE, "cyan":Fore.CYAN, "light_black":Fore.LIGHTBLACK_EX, "light_blue":Fore.LIGHTBLUE_EX, "light_cyan":Fore.LIGHTCYAN_EX, "light_green":Fore.LIGHTGREEN_EX, "light_magenta":Fore.LIGHTMAGENTA_EX, "light_red":Fore.LIGHTRED_EX, "light_white":Fore.LIGHTWHITE_EX, "light_yellow":Fore.LIGHTYELLOW_EX, "magenta":Fore.MAGENTA, "reset":Fore.RESET, "white":Fore.WHITE

}

back = {

"green":Back.GREEN, "black":Back.BLACK, "red":Back.RED, "yellow":Back.YELLOW, "blue":Back.BLUE, "cyan":Back.CYAN, "light_black":Back.LIGHTBLACK_EX, "light_blue":Back.LIGHTBLUE_EX, "light_cyan":Back.LIGHTCYAN_EX, "light_green":Back.LIGHTGREEN_EX, "light_magenta":Back.LIGHTMAGENTA_EX, "light_red":Back.LIGHTRED_EX, "light_white":Back.LIGHTWHITE_EX, "light_yellow":Back.LIGHTYELLOW_EX, "magenta":Back.MAGENTA, "reset":Back.RESET, "white":Back.WHITE

You Can Always Access The Colors Like That : * pysha.fore[...] * pysha.back[...]

functions.**colorprompt**(*prompt: str*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*, *char_color='\x1b[39m'*)

This Will Act Like Input. But Characters That User Enters Will Be Colored :) Args : * prompt -> The Text For Print And Wait For Input.

- out -> Just Standard Output. DO NOT TOUCH THIS :/
- **char_color -> The Color Of Character. Each Character Will Have This** Color.

functions.**condition**(*check: str*, *p=True*, ***kwargs*) → NoneOrStrOrInt

Descrioptions Are Available In 'classes.py' And In command Class.

functions.**create_dir**(*dir_name: str*, *create_parents=False*) → bool

Tries To Create A Directory.

Args : * dir_name -> Directory Name. You Can Pass An String Like 'test' Or

'test/test1/test2' For Directory. But If You Want To Make Parents Too, You Have To Pass Next Argument Manually Too.

- **create_parents(Default:False) -> If Its False, Parents Will Not Be** Created If They Are Not Exists. And If Its True, Parents Will Be Created Too !

Returns True If Its Successfully Done, Else Return False

functions.**delay_loop**(*delay: int*, *func: Callable*, *count: int = -1*, **args*, ***kwargs*)

This Function Will Try To Loop A Function In Specific count/infinite count.

Example : * delay_loop(10,func,5,args,kwargs)

In Every 10 Seconds Function func(args,kwargs) Will Be Called 5 Times.

functions.**display**(*temp: dict*, *string: str*, *c=True*) → None

Related To Loop Command. It Will Print String And Replace _i_ , _i++_ ... With The Value.

Args : * temp -> Carries Variables. * string -> The String Pattern That Should Be Printed In Output. * c -> If Its True, Answer Will colorized. And Else Won't Be Colorized.

functions.**escape_translator**(*target*, *char: str*) → list

This Function Will Help You To Recover Escaped Characters That Wrongly Splitted. Args : * target -> A List Of Splitted String/ A String That You Want To Escape.

- char -> The Character That You Want To Escape.

Example : `escape_translator("Hello Lets Escape All < That Have \ Like \< . All \< Will Be Escaped.","<")`

Out : *['Hello Lets Escape All ', ' That Have Like < . All < Will Be Escaped.']*

functions.**get_ans**(*temp: dict*, *string: str*, *c=True*) → str

Related To Loop Command. It Will Get Input And Replace _i_ , _i++_ ... With The Value.

Args :

- temp -> Carries Variables.
- string -> The String Pattern That Should Be Printed In Output.
- c -> If Its True, Answer Will colorized. And Else Won't Be Colorized.

functions.**index_split**(*string*, *ind*)

Split String By Index. Example : `index_split("Hello My Name Is Arshia",10)` Returns : `['Hello My N', 'me Is Arshia']`

functions.**loop**(*cmd: str*, *mode='p'*, *c=True*) → NoneOrStr

All Descriptions From This Function Came In *classes.py* - command Class.

functions.**make_array**(*dimentions*)

Make An Array In Dimentions That You Gave.

functions.**make_possibles**(*data*)

Tries To Get Possible Situations Of For Loops (Max 3 Variable).

functions.**merge**(**args*, *force_list=False*, *dont_change=False*)

This Function Will Merge Some List Or Some Dict Or Some Dict And List ! Example : `merge(["a","b"],{"c":"d","e":"f"},["g","h"])` Returns : `{"a":"","b":"","c":"","d":"","e":"","f":"","g":"","h":""}`

Example : `merge(["a","b"],{"c":"d","e":"f"},["g","h"],force_list=True)` Returns : `["a","b","c","e","d","f","g","h"]`

Example : `merge(["a","b"],{"c":"d","e":"f"},["g","h"],force_list=True,dont_change=True)` Returns : `["a","b","c","d","e","f","g","h"]`

Example: `merge(["a","b"],["c","d","e","f"],["g","h"])` Returns : `["a","b","c","d","e","f","g","h"]`

In Every 10 Seconds Function func(args,kwargs) Will Be Called 5 Times.

functions.**not_between_index**(*string*, *string_to_check*, *start_sign=""*, *end_sign=""*, *case_sensetive=True*, *start=0*)

Reverse Of Between Plus Index Of Target.

Args : * string -> A Simple String. (We Need To Check If This String Exists

In Another String Or Not) For Example If You Want To Check If 'hello' is not between " Characters, You Need To Pass hello As This Argument.

- **string_to_check -> A Big String That We Want To Check And Dive In :)**
  For Example 'hello My Name Is Arshia And "hello" Is ok Now As i think :)' And Pass It As This Argument.
- **start_sign -> This Should Be The Start Character That You Are Looking For.**
  For Example You Can Pass " As This Argument (Continue Of Previous Part Example) . But In Another Example That You Want To Check Text That Does Not In A Tag For Example html Tag You Can Pass "<" As Start.
- **end_sign -> This Should Be The Start Character That You Are Looking For.**
  For Example You Can Pass " As This Argument (Continue Of Previous Part Example) . Continue Of Tag Example You Can Pass ">" As End.
- **case_sensetive -> If Its True, Lower Case Or Upper Case In Words Are Important.**
  For Example If Its True, "Html" Is Different From "html" But If Its False, "Html" And "html" Are Equal.

Example : * not_between_index(":",'Arshia Said : "Hello ! There Is : Hiding Here :)"','"','"') * Returns :

12

- not_between_index(":",'":" Is Not Out Of "" Sign. But Now : It Is.','"','"')
- **Returns :**
  35

Returns True If Its Found Out Of target Sign, Else Returns False.

functions.**passprompt**(*prompt: str*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*, *mask_color='\x1b[39m'*, *mask='*'*) → str

This Will Get Password And Replace Characters With A Mask For Example '*'. Args : * prompt -> The Text For Print And Wait For Input.

- out -> Just Standard Output. DO NOT TOUCH THIS :/
- **mask_color -> The Color Of Mask. For Example You Can Print Red Stars For**
  Each Character :))
- **mask -> The Mask. Every Character That User Type, Will Be Replaced With**
  This Character :)

functions.**read_file**(*file_name: str*, *mode='s'*) → ListOrString

You Can Read Data From A File :)

Args : * file_name

- **mode(Default:'s')**
  [The Mode Can Be 's' Or 'l' Means List Or String.] If You Want To Get Output As String Mode 's' Will Be Good For You And If You Want To Get As list Mode 'l' Is Good For You.

Returns list of lines Of The File If mode Is 'l', string of lines If mode is 's'.

functions.**replace_in_file**(*file_name: str*, *data_to_replace: dict*) → bool

Tries To Replace A Dictionary.

Args : * file_name -> File Name. You Can Pass File Name That You Want Data

To Change.

- **data_to_replace -> Data That You Want To Replace As Dictinary.**
  For Example : {'test':'test1'} Will Replace All 'test' With 'test1' In The File.

Returns True If Its Successfully Done, Else Return False

`functions.rm_dir`(*dir_name: str*, *force=False*) → bool

Tries To Remove A Directory.

Args : * dir_name -> Directory Name. You Can Pass An String Like 'test' Or

'test/test1/test2' For Directory. If You Want To Force Remove That Childs Will Remove Too, Pass Next Argument Manually.

- **force(Default:False) -> If Its False And If directory Has Childs,**
  Directory Won't Be Removed.But If Its False Doesn't Matter What We Are Cold-Blood Murderes And We Kill Them All :)

Returns True If Its Successfully Done, Else Return False

`functions.set_array`(*array*, *dimentions*, *value*)

Set An Array Value In Complicated Dimentions.

`functions.write_file`(*file_name: str*, *text: ListOrString*) → bool

You Can Write Into A File :)

Args : * file_name * text Returns True If Its Successfully Done, Else Return False

1.9 pysha module

`pysha.banner`(*text*, *font=''*, *p=True*)

This Function Directly Use pyfiglet Library For Making A Banner.

Args: * text -> The Text That You Want To Convert To A Banner.

- **font -> Target Font From Figlet. If You Pass Empty Font Will Be Default**
  Figlet Font.
- **p -> This Parameter Will Set Print Mode. If Its True It Will Print**
  And If Its False It Will Return Banner As String.

Example : * Code `banner("pysha","chunky")`

- Output

```` ```______ __ | __ .–.–.——.| |--.---.-. | __/| | |__ --|| | _ | |___| |___ |_____||__|__|___._| ````

|_____|```

`pysha.l`(*char=('=', '\x1b[39m')*, *count=30*, *p=True*)

This Function Will Draw A Line In Terminal.

Kwargs: * char -> You Can Pass (char='-') Then Color Will Be Normal Color. Or

You Can Pass (char=('-',Fore.[color])) Then Color Will Be [color]. Default Is ('=',Fore.RESET)

- count -> Count Of Characters That Line Have.
- **p -> This Parameter Will Set Print Mode. If Its True It Will Print**
  And If Its False It Will Return Line As String.
````

pysha.**pp**(**args*, *curly_c='\x1b[39m'*, *colon_c='\x1b[39m'*, *quote_c='\x1b[39m'*, *mode='k'*, *end='\n'*)

Abbreviation of Pretty Print.

Args: *You Can Give As Much As You Can For Print :)*

Kwargs: * curly_c : Set Curly Brackets Colors or "{}" Color. * colon_c : Set Colon Colors or ":" Color. * quote_c : Set Quote Colors or "'" Color. * mode : Set Mode Of pp. Modes Are 'k' And 'i'. You Can Read About

Them In Modes Part.

- end : This Will Set End Of Print. Default Is '

'.

Modes : * k : This Will Use all colors That You Specified As Kwargs. * i : This Will Ignore Colors(curly_c,colon_c,quote_c) That You

Specified In Kwargs.

pysha.**rect**(**args*, *text_color='\x1b[39m'*, *first_line=('=', '\x1b[39m')*, *sep=('|', '\x1b[39m')*, *last_line=('=', '\x1b[39m')*, *distance_up=1*, *distance_down=1*, *length=30*, *p=True*)

This Function Will Draw A Rectangle Plus Your Target Text Middle Of It.

Args:

Just Your Texts. You Can Pass It Like rect("hello","Thanks") Or rect("hello Thanks")

kwargs: * text_color -> Color Of Your Text.

- **first_line -> This Parameter Will Set First Line Character And Color.**
  You Can Pass A Character, Default Text Color Will Be Setted. Or You Can Pass A Tuple That Contains Character And Color Like ('-',Fore.RED).
- **sep -> This Parameter Will Set Corners Character And Color.**
  You Can Pass A Character, Default Text Color Will Be Setted. Or You Can Pass A Tuple That Contains Character And Color Like ('-',Fore.RED).
- **last_line -> This Parameter Will Set Last Line Character And Color.**
  You Can Pass A Character, Default Text Color Will Be Setted. Or You Can Pass A Tuple That Contains Character And Color Like ('-',Fore.RED).
- **distance_up -> This Parameter Will Set Distance Between Text And First Line.**
  You Can Pass An Integer That Is Line Numbers Of Distance.
- **distance_down -> This Parameter Will Set Distance Between Text And Last Line.**
  You Can Pass An Integer That Is Line Numbers Of Distance.
- **length -> This Parameter Will Set Number Of Characters In First/Last**
  Line. You Can Pass An Integer That Is Character Count Of First/Last Line.
- **p -> This Parameter Will Set Print Mode. If Its True It Will Print**
  And If Its False It Will Return Rectangle As String.

Example : * Code `rect("Hello Welcome To Pysha :)",first_line='-',last_line='-', distance_up=2,distance_down=2)`

- Output

`------------------------------ | | | | | Hello | | Welcome To Pysha :) | | | | | ------------------------------`

pysha.**xp**(**args*, *prompt=(", '\x1b[39m')*)

This Function Will pp The Arguments. You Can Set A Prompt If You Want. (Prompt Will Not Be pp) After pp And Prompt It Will Get Input And Returns Input. You Can Change Prompt Color By Pass (prompt,Fore.[color]) To Prompt.

Kwargs: * prompt -> You Can Pass (prompt='Enter Your Name : ') And It Will Act As

input('Enter Your Name : '). Or You Can Pass (prompt=('Name : ',Fore.RED)) And It Will Print 'Name : ' In Red Color.

Example : * Code `name = xp("(Fore.RED)[W3LC0M3] To (Fore.GREEN)[T3ST] Application. ",prompt=('Name : ',Fore.CYAN)) pp(name)` * Output `_RED ->|W3LC0M3|_ To _GREEN -> |T3ST|_ Application. _CYAN -> |Name :|_ (input) (input)`

1.10 security module

class security.**Crypto**(*enc_ls*, *dec_ls*, *key*)

Bases: object

Crypto Class Is Made For Do Encryption And Decryption More Coder Friendly :)

dec(*s*)

enc(*s*)

class security.**Load**(*file_name*)

Bases: object

A Class To Load Saved Variables. Example :

``` def test():

print(name)

Load("file.txt")(test)

# Or handler = Load("file.txt") print(handler.vars) # All Variables As Dictionary

# Or Load("file.txt")() print(name) # If You Saved name It Will Be Shown :) ```

**load**(*module=None*)

**vars**()

**class** security.**Save**(*file_name*, *\*\*kwargs*)

Bases: object

A Class To Save Some Variables With Some Simple Encryption. Example :

``` Save("file.txt",name="test",family="123",...)()

# Or handler = Save("file.txt") handler.add_var("name","Arshia") handler.add_vars({"nickname":"MrMegaExpert","age":21}) handler.save() # Or handler()

# Or handler = Save("file.txt") handler["name"] = "Arshia" handler.save() # Or handler() ```

**add_var**(*key*, *value*)

**add_vars**(***kwargs*)

**save**()

security.**make_enc**(*alg*, *key=b''*)

make_enc Function Will Make You An instance Of Crypto Class That Contains A Chain Of Encryption Algorithm. In Fact You Are Free To Use Just One Or More. Args/Kwargs : * alg -> The Algorithm That Can Be Like Algorithms.XOR Or Like

[Algorithms.XOR,Algorithms.Base64]

- key -> The Target Key That You Want To Use In Encryption Process.

Example :

```
`a = make_enc([Algorithms.XOR,Algorithms.Base64],10) a.enc("Hello") # Encrypt
"Hello" Equals "Qm9mZmU=" a.dec("Qm9mZmU=") # Decrypt "Qm9mZmU" Equals "Hello"`
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

T

V

W

X