**Developer Guide**

**Developer Guide**

**PY Basic Alone**

Es una pequeña librería que tiene el objetivo de simplificar algunas tareas básicas en python; añadir nuevos tipos, chequeadores y bucles.

**Caracteristicas**

* Arquitectura modular
* Orientación a objetos con Estados
* Tiene múltiples dependencias; para rehusar librerías existentes y confiables

**Funciones**

**Random Bool**

Función que genera un valor boolean aleatorio. Utiliza la librería random.

def random\_bool() -> bool:

**Random Without**

Funcion que genera un numero aleatorio en el rango sin repetir los numeros enviados en la lista indicando si el rango es exclusivo o no.

def random\_without(range: int,

        repeated\_int\_ar: list,

        is\_exclusive:bool = False

        ) -> int:

**Random Probability**

Función que genera un resultado boolean verdadero sí la generación de un número aleatorio se encuentra por debajo del Rango enviado. Sirve para simular una probabilidad porcentual.

def random\_probability(porcentage: int)\

          -> bool:

**Repeat**

Función de alto orden que repite cada determinada cantidad de segundos la función enviada una cantidad determinada de veces en el hilo actual.

def repeat(repetition: int,

           seconds: int | float,

           function):

**Map**

Función maper modificada que permite iterar sobre todos los elementos anidados de una estructura de datos profunda; además puede recorrer tanto diccionarios como arrays.

# return array or dict

    def map(array, function):

**Iterator**

      Clase iterador para representar un recorredor de arrays que almacena los indices tras cada iteracion. para usarlo solo deben llamarse a su metodo next dentro del condicional del bucle y tomar sus elementos con get.

def iterator(array, is\_cycle = False):

**Get abc tuple**

Función que obtiene una tupla con todo el alfabeto latino inglés.

def get\_abc\_tuple():

**Get number tuple**

Función que obtiene una dupla con todos los símbolos numéricos arábigos.

def get\_number\_tuple():

**Get animal array**

Función que obtiene un array string predefinido con una lista de nombre de animales en inglés.

def get\_animal\_array():

**Get init array**

    Funcion que sirve para crear un array nuevo y lleno del dato indicado. Segun la clave enviada como format añadira un dato diferente, estas son:

\* abc para abecedario

\* number para numero

    ademas la clave order indica el orden de los datos de menor a mayor o alreves; las claves son normal e invert.

def init\_array(range, value = None):

**Calculate mid**

Función que calcula el promedio una lista de números.

def calculate\_mid(array:list)->float:

**Deep**

Función que calcula el máximo nivel de profundidad de un array enviado.

def deep(array) -> int:

**In Range**

Función que identifica si un número se encuentra entre el intervalo enviado retornando un verdadero si lo es; designando también si es inclusivo o inclusivo la comprobación.

def in\_range(number: int, range: list)\

        -> bool:

**Part from percent**

Función que Calcula la parte de un porcentaje enviado.

def part\_from\_percent(percent: int,

        total: int) -> float:

**Percent From Part**

Función que Calcula el porcentaje de una parte enviada.

def percent\_from\_part(part: int,

        total: int)-> float:

**Total From Part**

Función que Calcula el total de una parte enviada.

def total\_from\_part(percent,

        part: int) -> float:

**Valid String**

función para validar la cadena enviada; esto valida si la cadena es una cadena, no está vacía y es mayor que el mínimo (opcional).

def valid\_string(data: str, min\_size = -1,

        is\_strict:bool = False):

**Randint list**

Genera una lista de números aleatorios en el rango enviado. Usa la librería random.

def randint\_list(size: int,

        range\_arr: list) -> list:

**Set Inrange**

Funcion que ajusta un valor en el rango limite enviado.

def set\_inrange(value: int,

            range\_arr: list):

**Sum in range**

Funcion que suma dos valores en un rango limitado, si supera o es inferior al rango se limita a ajustar el limite.

# return number

    def sum\_in\_range(value\_a, value\_b,

                    range\_arr):

**Dict to array**

Función que convierte los elementos de un diccionario en una lista.

def dict\_to\_array(dict) -> list:

**Valid Index**

Funcion estricta de validacion para

    indices de arrays. Enva el tamaño

    para no generar una copia inesesaria

    del array.

# not return

def valid\_index(index: int, leng: int):

**Sum Array**

Funcion que suma los elementos de un

    array.

def sum\_array(array):

**Percent to index**

La función proporcionada percent\_to\_index(percent, limit)calcula el índice correspondiente a un valor porcentual determinado dentro de un rango específico definido por el límite.

def percent\_to\_index(percent, limit):

**Is Error**

Funcion que analiza si un valor es un

    error de retorno como  -1, none, void

    string o void array

def is\_error(value):

        return is\_error(value)

**Random without repeat**

Funcion que genera un numero aleatorio

    en el rango sin repetir los numeros

    enviados en la lista indicando si es

    exclusivo o no.

# return int

    def random\_without\_repeat(range\_arr,

        repeated\_int\_ar,

        is\_exclusive = False):

**Random IP**

Genera números IP aleatorios.

def random\_ip():

**Missing Numbers**

Funcion que calcula todos los numeros que faltan

    en una lista hasta el numero mas alto, y retorna

    una lista de rangos array con esos numeros faltantes.

def missing\_numbers(number\_list):

**Switch**

Clase interruptor para realizar cambios

        automaticos a variables boolean

        con cada llamada. Para usarla solo

        debe crearse un objeto con un nombre estilo

        boolean y comprobarlo con is\_true().

def switch():

**Counter call**

Tipo de dato especilizado para contar

        de forma agil un numero de llamadas.

        Se utiliza llamando al metodo is\_end en

        un condicional, con cada llamada sumara

        un numero hasta llegar al limite indicado,

        una vez alcanzado el limite se formateara.

def counter\_call(limit: int):

**Probability**

Es un objeto que sirve para calcular probabilidades

        multiples. Para usarlo deben enviarse con el metodo add

        claves con su probabilidad entera; y el metodo calculate

        generara un resultado basado en esas probabilidades.

def probability():

**Compare Strings**

Funcion que calcula las similitudes entre dos strings

    a partir de sus caracteres similares y su posicion

    en la cadena; despues hace una media entre ambos

    porcentajes.

# return int

    def compare\_strings(strig\_primal,

            string\_to\_compare):

**Related Dict**

Esta clase es un diccionario especial que permite

        busquedas de claves relacionadas. Su estructura

        es un diccionario de nodos que contienen claves.

        Para usarlo debes asignar con set una clave

        multiple separada con dos puntos :, y un dato.

        pueden modificarse los datos con set(key, dato)

        y obtener el dato con get(key)

def related\_dict(similar\_percent: int = 70):

**Cut From**

Funcion que corta el string enviado

    despues de cierto string indicado

    returnando la primera parte del string.

def cut\_from(string: str, caracter: str)\

    -> str:

**Cut Until**

Función que corta el streaming enviado hasta el String indicado.

def cut\_until(string: str, caracter: str)\

    -> str:

**Divide string**

Funcion que divide un string en el

    indice indicado.

# return string\_array

def divide\_string(index:int, string:str)\

-> list:

**Next word**

Funcion que busca la siguiente palabra al

    string enviado en el texto. GEMINI autor.

# return string

    def next\_word(text: str,

            searched\_string: str):

**Get Between**

Funcion que tome todo el string que

siguen despues de un string hasta llegar

al caracter indicado.

def get\_between(base:str, start:str,

        end:str) -> str:

**Insert String**

Funcion que sirve para añadir un string

    antes de la pocision indicada en el

    string enviado.

def insert\_string(index:int, base\_string:str,

        new\_string:str) -> str:

**Normalize**

Elimina las mayúsculas, espacios estra

    y tildes de un texto.

def normalize(text: str) -> str:

**Split by**

Funcion que divide un string por los

    string enviados en el array. Utiliza

    la libreria re

def split\_by(string, dividers\_arr: list):

**Simil Search**

Funcion que busca una palabra similar basado en

    su letras similares y su posicion en la cadena,

    basado en el porcentage enviado.

    # XXX No funciona bien

def simil\_search(array:list, element:str,

        required\_percent:int) -> str:

**Get Description**

Funcion que extrae de un string una lista de

elementos separadas por el separator string;

iniciando desde el init\_string hasta el

end\_string.

def get\_description(string,

        separator\_string, range\_string\_array):

**Array to dict**

Function that converts the elements of

    an array into the keys of a dictionary;

    with the data sent as an element.

    EXAMPLE:

        array = ['a', 'b', 'c', 'd', 'e']

        result:list = Basic.array\_to\_dict(array)

        print(result)

    OUTPUT:

        {'a': None, 'b': None, 'c': None,

        'd': None, 'e': None}

    """

def array\_to\_dict(array:list,

    data\_type = None)->dict:

**Seek**

Function that returns an element of

the array if the sent function returns

true. If the element does not exist,

it returns a value of none.

def seek(function, array:list) ->list:

**Strip Array**

Function that applies the strip function

    to all strings in a string array.

def strip\_array(array:list[str])\

        ->list[str]:

**Update**

function that replaces an element of the

    sent array with the new element, looking

    for an element that returns true in the

    sent function.

def update(function, array:list, element)\

        ->list:

**Whats Ranges**

La función proporcionada whats\_ranges(ranges\_named:dict, number:int) -> list[str]analiza un diccionario de rangos con nombre y un valor entero para determinar en qué rangos se encuentra el número.

def whats\_ranges(ranges\_named:dict,

            number:int)->list[str]:

**Count keys**

Función que cuenta el número de repeticiones

de un dato en el array enviado, retornando

un diccionario con la información resultante.

def count\_keys(keys\_array:list[str])\

        ->dict[int]:

**Crono**

Crea una medida de tiempo que puede llamarse utilizando el método stop para obtener un tiempo contado, como un cronómetro.

def crono():

**Pause**

Envoltorio para la función time sleep.

def pause(seconds:int):

**Splitip**

Función que divide un string por palabras,

lo convierte en un array, elimina los

espacios extra y las palabras void;

retornando una raíz con las palabras

limpias.

def splitip(text:str, divider:str)\

            ->list[str]:

**Repeat in thread**

Función que repite una acción enviada en un

intervalo de segundos usando un hilo propio.

def repeat\_in\_thread(interval:int,

        action, limit:int = -1)->None:

**Route in nested dict**

Función que busca en profundidad una serie

de claves ordenadas por profundidad a

través de diccionarios de diccionarios

anidados.

def route\_in\_nested\_dict(seek\_string:str,

        dict:dict[dict]) ->list[str]:

**Randon point**

Genera puntos aleatorios En un Rango

def random\_point(range\_ar):

**Random Point Arr**

def random\_point\_ar(quantity, range\_ar):

**Randindex**

def randindex(array):

**Randrange**

Envoltorio para la función random int

def randrange(range\_arr):

**Clean console**

def clean\_console():

**Has All**

Función que itera sobre los elementos del

array enviado buscando si todos los

elementos del array retornan true a la

función enviada.

def has\_all(array, function):

**Has Some**

Función que itera sobre los elementos del

array enviado buscando si alguno de los

elementos del array retornan true a la

función enviada.

def has\_some(array, function):

**Has None**

Función que itera sobre los elementos del

array enviado buscando si ninguno de los

elementos del array retornan true a la

función enviada.

def has\_none(array, function):

**New Thread**

Decorador para crear de forma fácil y ágil

nuevos hilos.

def new\_thread(function):

**Range Space**

Clase para almacenar representaciones

    de espacios utilizando intervalos de

    ejes límites.

def range\_space():

**Move Point**

Función que desplaza un a otro punto sin

superar los límites que los ejes enviados

como intervalos.

# return point

    def move\_point(

        old\_point:list[int],

        new\_point:list[int],

        range\_space

        ) -> list[int]:

**Is number**

Comprueba si el dato enviado el de un tipo

numérico.

def is\_number(data) -> bool:

**Play Sound**

def play\_sound(route:str)->None:

**Is Byte**

def is\_byte(data)-> bool:

**Is RGB**

def is\_rgb(data):