_2structures/_0structures.py

```
"""_________Lists, Tuples, Set, Dicts _________"""
#Lists[]         Ordenada     mutable       repetible
#Tuples()        Ordenada     inmutable     repetible
#Set{} set()     Desordenada  inmutable     irrepetible    añadible
#Dicts{:}        Ordenados    mutables      irrepetibles   clave-valor
"""___________Funciones Comunes_____________"""
x=0
# Operaciones Generales
len()
type()
x.copy()        # (no a tuplas).
# Ordenación y Orden
sorted()
reversed()      # (no a conjuntos ni diccionarios).
# Funciones de Agregación
min()
max()
sum()
# Evaluación y Verificación
all()
any()
# Iteración y Enumeración
enumerate()     # (no a conjuntos ni diccionarios).
zip()
map()
filter()
# Funciones de Espera y Tiempo
import time
time.sleep(1)
# Funciones de Importación
from functools import reduce
reduce()
from collections import Counter
Counter()       # Aplica a listas, tuplas (no a conjuntos ni diccionarios, sin repetidos).

# Listas : Todas
# Tuplas: Exceptuando las que requieren mutabilidad (como copy()).
# Conjuntos: Aplica la mayoría, pero no aquellas que requieren orden (reversed(), enumerate()).
# Diccionarios: No aplican directamente, pero pueden aplicarse a claves/valores.

#-----------------------------------------List-----------------------------------------
my_list = [3, 1, 4, 1, 5, 9, 2, 6]
caca = ["KA", "KI", "QU"]
my_list[3]                      #1
sublista = [1, 2, 3, 4, 5][1:3] #(2,3) Devuelve un objeto de corte que es usado para rebanar objetos iterables.
lista_listas= my_list,caca
pellizquito=lista_listas[1][2]  # QU
copia_my_list= my_list[:]

```

```
my_list.sort()            #[1, 1, 2, 3, 4, 5, 6, 9]          Modifica la original//Sorted crea copia
my_list.sort(reverse=True) #[9, 6, 5, 4, 3, 2, 1, 1]          Sort/reverse da lista
my_list.reverse()         #[6, 2, 9, 5, 1, 4, 1, 3]

my_list.append(33)        #[6, 2, 9, 5, 1, 4, 1, 3, 33]
my_list.insert(2,"CHO")   #[3, 1, 'CHO', 4, 1, 5, 9, 2, 6]
my_list.extend(caca)      #[3, 1, 4, 1, 5, 9, 2, 6, 'KA', 'KI', 'QU']
my_list += [10, 11, 12]   # Extender la lista con otra lista
my_list = ["Uno" if x == 1 else x for x in my_list]  #Reemplazar todos los elementos con valor 1 por "Uno"

my_list.remove("KA")      #[3, 4, 1, 5, 9, 2, 6, "KI", "QU"] Elimina la primera ocurrencia del elemento indicado.
my_list.pop(3)            #1 Si no se indica elimina el ultimo indice, si se especifica el indice pues el indice.
my_list.clear()           # Elimina todos los elementos de la lista, dejándola vacía.

"Pepe" in my_list         #False
my_list.count(1)          #2
my_list.index(4)          #2 Primera aparición de 4 en la lista.             No se puede rindex
my_list.index(1,1,7)      #3 Encuentra el primer 1 empezando desde la 2º posicion y acabando en la 6º

#-----------------------------------------Tuple-----------------------------------------------------
tupla = (1, 2, 3, 4, 1, 2, 1)
tupla_unitaria=("Juan",)
tupla_iguales=(0,)*3                # (0,0,0)
tumpla_unitaria=(1,)                # Necesaria la ,

h,y,j,k="H","o","l","a"            #Desempaquetado Tuplas
print(h,y,j,k)                     # H o l a
language = 'Python'
a,b,c,d,e,f = language
print(a,b,c,d,e,f)                 # P y t h o n

print(tupla.index(3))  # 2
print(tupla.count(1))  # 3
#-----------------------------------------Set-------------------------------------------------------
my_set1= {2, 4, 7, 23, 5}
my_set2= {7, 14, "JU", 23, True}
my_set2.add(6)            #{7, 6, 14, "JU", 23, True}
my_set1.update(my_set2)   #{True, 2, 4, 5, 7, 14, 'JU', 23}    modifica la original.
my_set1.update(my_list)   #{1, 2, 3, 4, 5, 6, 'CHO', 7, 9, 'QU', 23, 33, 'KA', 'KI'} set + otro conjunto(lista..)
my_seta=my_set1.union(my_set2)                                  #crea una nueva

my_set2.pop()             #{23, 'JU', 7, 14}      elimina un elemento aleatorio y lo puedes printear.
my_set2.remove(7)         #{14, "JU", 23, True}   elimina un elemento concreto. Da error si no existe.
```

```
93  my_set2.discard(7)         #{14, "JU", 23, True}   descarta el elemento, pero no da error si no existe.
94  my_set2.clear()            #la deja vacia
95
96  set1 = {1, 2, 3, 4, 5, 6}
97  set2 = {4, 5, 6, 7, 8, 9}
98  set3 = {2, 3, 1, 6}
99
100 union              = set1.union(set2)                        #{1,2,3,4,5,6,4,5,6,7,8,9} No modifica la original
101 union2             = set1 | set2
102
103 interseccion       = set1.intersection(set2, set3)           # {2, 3}
104 interseccion2      = set1 & set2
105
106 diferencia         = set1.difference(set2, set3)             # {1}
107 diferencia2        = set1 - set2
108
109 symetric_diff      = set1.symmetric_difference(set2)         # {1, 2, 3, 6, 7, 8}  no en ambos
110 symetric_diff2     = set1 ^ set2
111
112 count=sum(1 for element in my_set1 if element==4)        #Count
113 print(2 in set1)              # True.  Verifica si el número 2 está en el conjunto
114 print(set1.isdisjoint(set3))  # False. Hay elementos en comun.
115 print(set3.issubset(set1))    # True.  Verifica si set3 es un subconjunto de set1
116 print(set3.issuperset(set1))  # False. Verifica si set3 es un superconjunto de set1
117
118 #----------------------------------------Dictionary-------------------------------------------------
119 my_dict= {"coche":"Toyota","Edad":32,"Culombiana":True,"comida":"Salmorejo"}
120 my_dict["Idioma"]="Frances"           #añade Idioma: "Frances"
121 my_dict["Idioma"]="Español"           #Sustituye "Frances" por "Español"
122
123 my_dict.update({"Color": "Rojo", "Modelo": "Corolla"}) # {'coche': 'Toyota', 'Culombiana': True, 'Color': 'Rojo', 'Modelo': 'Corolla'}
124 eliminado=my_dict.pop("coche")                          #"Toyota elimina la clave "coche" y guarda el valor.
125
126 for clave, valor in my_dict.items():
127     print(clave, valor)
128 my_dict.keys()                      #dict_keys(['coche', 'Edad', 'Culombiana'])
129 my_dict.values()                    #dict_values(['Toyota', 32, True])
130 my_dict.items()                     #dict_items([('coche', 'Toyota'), ('Edad', 32), ('Culombiana', True)])
131 my_dict.get('coche')                #Toyota     accede al valor de una clave y no da error si está vacia.
132 my_dict.get("Color","No encontrada")                  #Devuelve NO si "Color" no está en el diccionario.
133 my_dict.append()
134 # zip(keys,values)                  Intentar mismo tamaño
135 for clave, valor in my_dict.items():
136     print("Clave: ", clave)
137     print("Valor: ", valor)
138     print("")
```

```
139
140 pizza = {"ingredientes": ["cheese", "tomato", "pepperoni", "mushrooms", "onions", "olives"]}
141
142 for ingrediente in pizza["ingredientes"]:
143     print(ingrediente)
144
145 registro = {"Paco":10,"Pepe":5,"Juan":7}
146 jugador_mas_alto=max(registro,key=registro.get)
147 print(jugador_mas_alto)
148 """------------------------------------------------------------------------"""
149
```