

Funciones y sus parámetros

```
def nombrefuncion(parametro1,parametro2):
```

Declarar lista y diccionario

```
lista=[]  
diccionario={}  
tupla = (1, "Hello", 3.4)
```

- Recordad que las tuplas son inmutables.

Añadir a una lista

```
lista.append(x) #Al final de la lista añade el elemento x  
lista2.append(lista) # Una lista2 se añade "lista", estando primero lista2 y luego lista.
```

Unir dos listas

```
from heapq import merge  
nuevalista=merge(l1,l2)
```

Conjuntos

```
set(lista) #Convertir lista a conjunto
```

- Operaciones conjuntos Python 3: unión exclusiva con ^, unión normal con | y la intersección con &..

Ordenar una lista (Simple)

```
nombres.sort() #Ascendente  
nombres.sort(reverse=True) #Descendente
```

Ordenar una lista (Key value)

```
my_alphabet = ['a', 'b', 'c']  
def custom_key(word): # Función que define un valor único a ordenar de cada elemento  
    numbers = []  
    for letter in word:  
        numbers.append(my_alphabet.index(letter))  
    return numbers  
x=['cbaba', 'ababa', 'bbaa']  
x.sort(key=custom_key) #Ordenamos indicando como tendremos valor usado al ordenar
```

Diccionario ordenado

```
from collections import OrderedDict
from operator import itemgetter

d = {"aa": 3, "bb": 4, "cc": 2, "dd": 1}
print(OrderedDict(sorted(d.items(), itemgetter(1), True)))
```

Convertir a lista

```
x=list(loquesea)
```

Copiar lista

```
from copy import copy, deepcopy
p2 = deepcopy(p)
#Otra forma
p2=p[:]
```

Copiar diccionario

```
original = dict(a=1, b=2, c=dict(d=4, e=5))
new = original.copy()
```

Eliminar duplicados de una lista

```
#Elimina entradas duplicadas de una lista
def eliminarDuplicadosLista(l):
    return list(set(l))
```

Lista de rangos numéricos

```
range(0, 10,1)
Me saca una lista con los números del 0 al 9 (10 no incluido) con incremento 1
```

Rellenar con ceros

```
import numpy as np
p = np.zeros((2,1))
# Ceros en 2 filas y una columna
p([[ 0.],[ 0.]])
```

Completa - Autor: Sergi García Barea

Parámetros desde consola

```
import sys
nombreMaximoAntiguo=sys.argv[1]
```

Split y map

```
R,C,L,H=map(int, input().split())

#Divide la entrada en tokens y la mapea a cada variable
for _ in range(R):
    pizza.append(input())
#Lee R filas completas (una string por fila)
```

Números aleatorios

```
import random
x=random.randint(0,10) #Numero entre 0 y 10 ambos incluidos
eje=random.sample(list1,3) #3 ejemplos aleatorios de list1
random.shuffle(mylist) #desordena la lista
```

Expresiones regulares

```
# match → Comprueba que una cadena cumpla una expresión regular
# search → Busca primera ocurrencia de expresión regular, devuelve None si no encuentra
# split → separa por expresión regular ; sub → sustituye la expresión regular
# “\” Señala una secuencia especial “\d”
# “.” Cualquier carácter (excepto salto de línea) “he..o”
# “^” Comienza con “^hello”
# “$” Acaba con “world$”
# “*” Cero o más ocurrencias “aix*”
# “+” Una o más ocurrencias “aix+”
# {} Exactamente el número especificado de ocurrencias “al{2}”
# | Uno u otro (exclusivo) “falls|stays”
# () Para agrupar
```

Ejemplo:

```
import re
s1=input()
f(re.match("[a-z][A-Z]*$",s1)): # Verdadero si empieza por minúscula y el resto son mayúsculas
    print(s1[0].upper()+s1[1:].lower())
elif(re.match("[A-Z][A-Z]*$",s1)): # Verdadero si son todo mayúsculas
    print(s1.lower())
```

Cargar datos JSON

```
import json
import sys
import os
if os.path.isfile(nombreMaximoAntiguo+'Precalculos.json'):
    with open(nombreMaximoAntiguo+'Precalculos.json') as data_file:
        dicYaSolucionados = json.load(data_file)
Guardar datos en JSON
#Finalmente guardamos el precalculo
with open(sys.argv[1]+'Precalculos.json', 'w') as outfile:
    json.dump(dicYaSolucionados, outfile)
```

Funciones estadísticas Numpy

```
import numpy as np
print("Media de ingredientes "+str(np.mean(ingredientePorCadaPizza)))
print("-> Desviación típica: "+ str(np.std(ingredientePorCadaPizza)))
print("-> Percentil 25: "+ str(np.percentile(ingredientePorCadaPizza, 25)))
print("-> Perc 50 (Mediana): "+ str(np.percentile(ingredientePorCadaPizza, 50)))
print("-> Percentil 75: "+ str(np.percentile(ingredientePorCadaPizza, 75)))
```

Dibujado con GNUPlot

```
import matplotlib.pyplot as plt
#Imprimo histograma de cuantas v
plt.title(''+sys.argv[1]+' - Histograma aparición ingredientes en pizzas')
plt.style.use('ggplot')
plt.xlabel("Frecuencia ingrediente")
plt.ylabel("Ingredientes con misma frecuencia")
plt.hist(listaIngredientesOrdenadosSoloNumeros)
plt.savefig(sys.argv[1]+"-histograma.png")
#Borramos Lienzo
plt.clf()

#Dibujamos diagrama circular
plt.title(''+sys.argv[1]+' - Circular aparición ingredientes en pizzas')
plt.pie(listaIngredientesOrdenadosSoloNumeros, autopct="%1.1f%%")
plt.savefig(sys.argv[1]+"-circulo.png")
```



Permutaciones

```
import itertools
```

```
#EJEMPLOS
```

```
itertools.product('ABCD', repeat=2)
```

```
# AA AB AC AD BA BB BC BD CA CB CC CD DA DB DC DD
```

```
# product() p, q, ... [repeat=1]: cartesian product, equivalent to a nested for-loop
```

```
itertools.permutations('ABCD', 2)
```

```
# AB AC AD BA BC BD CA CB CD DA DB DC
```

```
# permutations() p[, r]: r-length tuples, all possible orderings, no repeated elements
```

```
itertools.combinations('ABCD', 2)
```

```
# AB AC AD BC BD CD
```

```
combinations() p, r: r-length tuples, in sorted order, no repeated elements
```

```
itertools.combinations_with_replacement('ABCD', 2)
```

```
# AA AB AC AD BB BC BD CC CD DD
```

```
# combinations_with_replacement() p, r: r-length tuples, in sorted order, with repeated elements
```