


13 Estructuras de datos concisas.

Objetivo. ...

Funciones de Python: ...

[MACTI-Algebra_Lineal_01](#) by [Luis M. de la Cruz](#) is licensed under [Attribution-ShareAlike 4.0 International](#) 

14 Listas concisas

En matemáticas podemos definir un conjunto como sigue:

$$S = \{x^2 : x \in (0, 1, 2, \dots, 9)\} = \{0, 1, 4, \dots, 81\}$$

En Python es posible crear este conjunto usando lo que conoce como *list comprehensions* (generación corta de listas) como sigue:

```
S = [x**2 for x in range(10)]  
print(S)
```

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

Las listas concisas son usadas para construir listas de una manera muy concisa, natural y fácil, como lo hace un matemático. La forma precisa de construir listas concisas es como sigue:

```
[ expresion for i in S if predicado ]
```

Donde **expresion** es una expresión que se va a aplicar a cada elemento **i** de la secuencia **S**; opcionalmente, es posible aplicar el **predicado** antes de aplicar la **expresion** a cada elemento **i**.

14.1 Ejemplo 1.

Usando listas concisas, crear el siguiente conjunto:

$$M = \{\sqrt{x} : x \in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) \text{ y } x \text{ es par}\} = \{\sqrt{2}, \sqrt{4}, \sqrt{6}, \sqrt{8}, \sqrt{10}\}$$

```
from math import sqrt  
  
M = [sqrt(x) for x in range(2,11) if x%2 == 0]  
print(M)
```

[1.4142135623730951, 2.0, 2.449489742783178, 2.8284271247461903, 3.1622776601683795]

En el ejemplo anterior se distingue lo siguiente:

1. La secuencia de entrada: `range(2,11)` (`[2, 3, 4, 5, 6, 7, 8, 9, 10]`).
2. La etiqueta `i` que representa los miembros de la secuencia de entrada.
3. La expresión de predicado: `if x % 2 == 0`.
4. La expresión de salida `sqrt(x)` que produce los elementos de la lista resultado, los cuales provienen de los miembros de la secuencia de entrada que satisfacen el predicado.

14.2 Ejemplo 2.

Obtener todos los enteros de la siguiente lista, elevarlos al cuadrado y poner el resultado en una lista:

```
lista = [1,'4',9,'luiggi',0,4,('mike','dela+')]

```

```
lista = [1,'4',9,'luiggi',0,4,('mike','dela+')]

```

```
resultado = [x**2 for x in lista if isinstance(x, int)]
print( resultado )

```

[1, 81, 0, 16]

14.3 Ejemplo 3.

Crear la siguiente lista:

$$V = (2^0, 2^1, 2^2, \dots, 2^{12}) = (1, 2, 4, 8, \dots, 4096)$$

```
[2**x for x in range(13)]

```

[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096]

14.4 Ejemplo 4.

Transformar grados Celsius en Fahrenheit y viceversa.

Celsius → Fahrenheit

```
c = [0, 22.5, 40, 100]
f = [(9/5)*t + 32 for t in c]
print(f)
```

[32.0, 72.5, 104.0, 212.0]

Fahrenheit → Celsius

```
cn = [(5/9)*(t - 32) for t in f]
print(cn)
```

[0.0, 22.5, 40.0, 100.0]

Observa que en ambos ejemplos la expresión es la fórmula de conversión entre grados.

15 Anidado de listas concisas.

15.1 Ejemplo 5.

Crear la siguiente lista:

$$M = \{\sqrt{x} \mid x \in S \text{ y } x \text{ impar}\}$$

con

$$S = \{x^2 : x \in (0 \dots 9)\} = \{0, 1, 4, \dots, 81\}$$

Este ejemplo se puede realizar como sigue:

```
S = [x**2 for x in range(0,10)]
M = [sqrt(x) for x in S if x%2]
print('S = {}'.format(S))
print('M = {}'.format(M))
```

S = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

M = [1.0, 3.0, 5.0, 7.0, 9.0]

Sin embargo, es posible anidar las listas concisas como sigue:

```
M = [sqrt(x) for x in [x**2 for x in range(0,10)] if x%2]
print('M = {}'.format(M))
```

```
M = [1.0, 3.0, 5.0, 7.0, 9.0]
```

15.2 Ejemplo 6.

Sea una matriz identidad de tamaño $n \times n$:

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

En Python esta matriz se puede representar por la siguiente lista:

```
[[1,0,0, ..., 0],
 [0,1,0, ..., 0],
 [0,0,1, ..., 0],
 .....
 [0,0,0, ..., 1]]
```

- Usando *list comprehensions* anidados se puede obtener dicha lista:

```
n = 8
[[1 if col == row else 0 for col in range(0,n)] for row in range(0,n)]
```

```
[[1, 0, 0, 0, 0, 0, 0, 0],
 [0, 1, 0, 0, 0, 0, 0, 0],
 [0, 0, 1, 0, 0, 0, 0, 0],
 [0, 0, 0, 1, 0, 0, 0, 0],
 [0, 0, 0, 0, 1, 0, 0, 0],
 [0, 0, 0, 0, 0, 1, 0, 0],
 [0, 0, 0, 0, 0, 0, 1, 0],
 [0, 0, 0, 0, 0, 0, 0, 1]]
```

Observa que en este caso la expresión de salida es una lista concisa:

```
[1 if col == row else 0 for col in range(0,n)] .
```

Además, la expresión de salida de esta última lista concisa es el operador ternario: `1 if col == row else 0`

15.3 Ejemplo 7.

Calcular números primos en el rango `[2,50]`.

En este ejercicio se usa el algoritmo conocido como criba de Eratóstenes. Primero se encuentran todos aquellos números que tengan algún múltiplo. En este caso solo vamos a buscar en el intervalo `[2, 50]`.

La siguiente lista concisa calcula los múltiplos de `i` (prueba cambiando el valor de `i` a 2, 3, 4, 5, 6, 7) y observa el resultado.

```
i = 4
[j for j in range(i*2, 50, i)]
```

`[8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48]`

Ahora, para cambiar el valor de `i` a 2, 3, 4, 5, 6, 7 con una lista concisa se puede hacer lo siguiente:

```
[i for i in range(2,8)]
```

`[2, 3, 4, 5, 6, 7]`

Usando las dos listas concisas creadas antes, generamos todos aquellos números en el intervalo `[2, 50]` que tienen al menos un múltiplo (y que por lo tanto no son primos)

```
noprimos = [j for i in range(2,8) for j in range(i*2, 50, i)]
print('NO primos: \n{}'.format(noprimos))
```

NO primos:

`[4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 10, 15, 20, 25, 30, 35, 40, 45, 12, 18, 24, 30, 36, 42, 48, 14, 21, 28, 35, 42, 49]`

Para encontrar los primos usamos una lista concisa verificando los números que faltan en la lista de `noprimos`, esos serán los números primos que estamos buscando:

```
primos = [x for x in range(2,50) if x not in noprimos]
print('Primos: \n{}'.format(primos))
```

Primos:

`[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]`

Juntando todo:

```
[x for x in range(2,50) if x not in [j for i in range(2,8) for j in range(i*2, 50)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

15.4 Listas concisas con elementos no numéricos.

Las listas también pueden contener otro tipo de elementos, no solo números. Por ejemplo:

```
mensaje = 'La vida no es la que uno vivió, sino la que uno recuerda'
```

```
print(mensaje)
```

La vida no es la que uno vivió, sino la que uno recuerda

```
palabras = mensaje.split()
print(palabras,end='')
```

```
['La', 'vida', 'no', 'es', 'la', 'que', 'uno', 'vivió,', 'sino', 'la', 'que', 'uno', 'recuerda']
```

Vamos a crear una lista cuyos elementos contienen cada palabra de la lista anterior en mayúsculas, en forma de título y su longitud, estos tres elementos agregados en una tupla:

```
tabla = [(w.upper(), w.title(), len(w)) for w in palabras]
print(tabla)
```

```
[('LA', 'La', 2), ('VIDA', 'Vida', 4), ('NO', 'No', 2), ('ES', 'Es', 2), ('LA', 'La', 2), ('QUE', 'Que', 3), ('UNO', 'Uno', 3), ('VIVIÓ,', 'Vivió,', 6), ('SINO', 'Sino', 4), ('LA', 'La', 2), ('QUE', 'Que', 3), ('UNO', 'Uno', 3), ('RECUERDA', 'Recuerda', 8)]
```

16 Conjuntos concisos

Al igual que las listas concisas, también es posible crear conjuntos usando los mismos principios, la única diferencia es que la secuencia que resulta es un objeto de tipo **set**.

Definición.

```
{expression(variable) for variable in input_set [predicate][, ...]}
```

1. **expression** : Es una expresión **opcional** de salida que produce los miembros del nuevo conjunto a partir de los miembros del conjunto de entrada que satisfacen el **predicate**.
2. **variable** : Es una variable **requerida** que representa los miembros del conjunto de entrada.

3. `input_set`: Representa la secuencia de entrada. (**requerido**).
4. `predicate`: Expresión **opcional** que actúa como un filtro sobre los miembros del conjunto de entrada.
5. `[, ...]`: Otra *comprehension* anidada **opcional**.

16.1 Ejemplo 8.

Supongamos que deseamos organizar una lista de nombres de tal manera que no haya repeticiones, que los nombres tengan más de un carácter y que su representación sea con la primera letra mayúscula y las demás minúsculas. Por ejemplo, una lista aceptable sería:

```
nombres = ['Luis', 'Juan', 'Angie', 'Pedro', 'María', 'Diana']
```

Leer una lista de nombres del archivo `nombres` y procesarlos para obtener una lista similar a la descrita.

```
# Abrimos el archivo en modo lectura
archivo = open('nombres','r')

# Leemos la lista de nombres y los ponemos en una lista
lista_nombres = archivo.read().split()

# Vemos la lista de nombres
print(lista_nombres)
```

```
['A', 'LuCas', 'Sidronio', 'Michelle', 'a', 'ANGIE', 'Luis', 'lucas', 'MICHelle',
'Pedro', 'PEPE', 'Manu', 'luis', 'diana', 'sidronio', 'pepe', 'a', 'a', 'b']
```

```
# Procesamos las palabras como se requiere
nombres_set = {nombre[0].upper() + nombre[1:].lower()
               for nombre in lista_nombres
               if len(nombre) > 1 }
print(nombres_set)
```

```
{'Pepe', 'Lucas', 'Angie', 'Sidronio', 'Pedro', 'Manu', 'Luis', 'Diana', 'Michelle'}
```

```
# Transformamos el conjunto a una lista
nombres = list(nombres_set)
print(nombres)
```

```
['Pepe', 'Lucas', 'Angie', 'Sidronio', 'Pedro', 'Manu', 'Luis', 'Diana', 'Michelle']
```

16.2 Ejemplo 9.

Observa los siguientes ejemplos de conjuntos concisos y explica su funcionamiento.

```
{s for s in [1, 2, 1, 0]}
```

{0, 1, 2}

```
{s**2 for s in [1, 2, 1, 0]}
```

{0, 1, 4}

```
{s**2 for s in range(10)}
```

{0, 1, 4, 9, 16, 25, 36, 49, 64, 81}

```
{s for s in range(10) if s % 2}
```

{1, 3, 5, 7, 9}

```
{(m, n) for n in range(2) for m in range(3, 5)}
```

{(3, 0), (3, 1), (4, 0), (4, 1)}

17 Dicionarios concisos

- Es un método para transformar un diccionario en otro diccionario.
- Durante esta transformación, los objetos dentro del diccionario original pueden ser incluidos o no en el nuevo diccionario dependiendo de una condición.
- Cada objeto en el nuevo diccionario puede ser transformado como sea requerido.

Definición.

```
{key:value for (key,value) in dictionary.items()}
```


17.1 Ejemplo 9.

Duplicar el valor (*value*) de cada entrada (*item*) de un diccionario:

Recuerda como funcionan los diccionarios:

```
dicc = {'a': 1, 'b': 2, 'c': 3, 'd': 4}
print(dicc.keys()) # Función para obtener las claves
print(dicc.values()) # Función para obtener los valores
print(dicc.items()) # Función para obtener los items
```

```
dict_keys(['a', 'b', 'c', 'd'])
dict_values([1, 2, 3, 4])
dict_items([('a', 1), ('b', 2), ('c', 3), ('d', 4)])
```

Para crear el diccionario del ejemplo hacemos lo siguiente:

```
# Definición del diccionario
dicc = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}

# Duplicación de los valores del diccionario
dicc_doble = {k:v*2 for (k,v) in dicc.items()}

# Mostramos el resultado
print(dicc)
print(dicc_doble)
```

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
{'a': 2, 'b': 4, 'c': 6, 'd': 8, 'e': 10}
```

17.2 Ejemplo 10.

Duplicar la clave (*key*) de cada entrada (*item*) del diccionario:

```
dict1_keys = {k*2:v for (k,v) in dict1.items()}
print(dict1_keys)
```

```
{'aa': 1, 'bb': 2, 'cc': 3, 'dd': 4}
```

17.3 Ejemplo 11.

Crear un diccionario donde la clave sea un número divisible por 2 en un rango de 0 a 10 y sus valores sean el cuadrado de la clave.

```
# La forma tradicional
numeros = range(11)
dicc = {}

for n in numeros:
    if n%2==0:
        dicc[n] = n**2

print(dicc)
```

{0: 0, 2: 4, 4: 16, 6: 36, 8: 64, 10: 100}

```
# Usando dict comprehensions
dicc_smart = {n:n**2 for n in numeros if n%2 == 0}

print(dicc_smart)
```

{0: 0, 2: 4, 4: 16, 6: 36, 8: 64, 10: 100}

17.4 Ejemplo 12.

Intercambiar las claves y los valores en un diccionario.

```
a_dict = {'a': 1, 'b': 2, 'c': 3}
{value:key for key, value in a_dict.items()}
```

{1: 'a', 2: 'b', 3: 'c'}

```
# OJO: No siempre es posible hacer lo anterior.
a_dict = {'a': [1, 2, 3], 'b': 4, 'c': 5}
{value:key for key, value in a_dict.items()}
```

TypeError: unhashable type: 'list'

17.5 Ejemplo 14.

Convertir Fahrenheit a Celsius y viceversa.

```
# Usando map, lambda y diccionarios
[32.0, 72.5, 104.0, 212.0]
fahrenheit_dict = {'t1':32.0, 't2':72.5, 't3':104.0, 't4':212.0}

celsius = list(map(lambda f: (5/9)*(f-32), fahrenheit_dict.values()))

celsius_dict = dict(zip(fahrenheit_dict.keys(), celsius))

print(celsius_dict)
```

```
{'t1': 0.0, 't2': 22.5, 't3': 40.0, 't4': 100.0}
```

```
# Usando dict comprehensions !
celsius_smart = {k:(5/9)*(v-32) for (k,v) in fahrenheit_dict.items()}
print(celsius_smart)
```

```
{'t1': 0.0, 't2': 22.5, 't3': 40.0, 't4': 100.0}
```

17.6 Ejemplo 15.

Dado un diccionario, cuyos valores son enteros, crear un nuevo diccionario cuyos valores sean mayores que 2.

```
a_dict = {'a':1, 'b':2, 'c':3, 'd':4, 'e':5, 'f':6, 'g':7, 'h':8}
print(a_dict)

a_dict_cond = { k:v for (k,v) in a_dict.items() if v > 2 }
print(a_dict_cond)
```

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6, 'g': 7, 'h': 8}
{'c': 3, 'd': 4, 'e': 5, 'f': 6, 'g': 7, 'h': 8}
```

17.7 Ejemplo 16.

Dado un diccionario, cuyos valores son enteros, crear un nuevo diccionario cuyos valores sean mayores que 2 y que además sean pares.

```
a_dict_cond2 = { k:v for (k,v) in a_dict.items() if (v > 2) and (v % 2) == 0}
print(a_dict_cond2)
```

```
{'d': 4, 'f': 6, 'h': 8}
```

17.8 Ejemplo 17.

Dado un diccionario, cuyos valores son enteros, crear un nuevo diccionario cuyos valores sean mayores que 2 y que además sean pares y divisibles por 3.

```
# La forma tradicional
a_dict_cond3_loop = {}

for (k,v) in a_dict.items():
    if (v>=2 and v%2 == 0 and v%3 == 0):
        a_dict_cond3_loop[k] = v

print(a_dict_cond3_loop)
```

```
{'f': 6}
```

```
# Usando dict comprehensions
a_dict_cond3 = {k:v for (k,v) in a_dict.items() if v>2 if v%2 == 0 if v%3 == 0}

print(a_dict_cond3)
```

```
{'f': 6}
```

17.9 Ejemplo 18.

Apartir de un diccionario con valores enteros, identificar los valores pares y los impares, y sustituir los valores por etiquetas 'par' e 'impar' según corresponda.

```
print(a_dict)
a_dict_else = { k:('par' if v%2==0 else 'impar') for (k,v) in a_dict.items() }
```

```
print(a_dict_else)
```

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6, 'g': 7, 'h': 8}
{'a': 'impar', 'b': 'par', 'c': 'impar', 'd': 'par', 'e': 'impar', 'f': 'par', 'g':
'impar', 'h': 'par'}
```

17.10 Ejemplo 19.

Crear un diccionario cuyos valores sean diccionarios.

```
# con dict comprehensions
anidado = {'primero':{'a':1}, 'segundo':{'b':2}, 'tercero':{'c':3}}
pi = 3.1415
float_dict = {e_k:{i_k:i_v*pi for (i_k, i_v) in e_v.items()} for (e_k, e_v) in anidado.items()}

print(float_dict)
```

```
{'primero': {'a': 3.1415}, 'segundo': {'b': 6.283}, 'tercero': {'c': 9.4245}}
```

```
# La forma tradicional sería:
anidado = {'primero':{'a':1}, 'segundo':{'b':2}, 'tercero':{'c':3}}
pi = 3.1415
for (e_k, e_v) in anidado.items():
    for (i_k, i_v) in e_v.items():
        e_v.update({i_k: i_v * pi})

anidado.update({e_k:e_v})

print(anidado)
```

```
{'primero': {'a': 3.1415}, 'segundo': {'b': 6.283}, 'tercero': {'c': 9.4245}}
```

17.11 Ejemplo 20.

Eliminar números duplicados de una lista.

```
numeros = [i for i in range(1,11)] + [i for i in range(1,6)]
numeros
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5]
```

```
# Una manera es:
numeros_unicos = []
for n in numeros:
    if n not in numeros_unicos:
        numeros_unicos.append(n)
numeros_unicos
```

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

```
# Otra forma mas pythonica!
numeros_unicos_easy = list(set(numeros))
numeros_unicos_easy
```

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

17.12 Ejemplo 21.

Eliminar objetos duplicados de una lista de diccionarios.

```
datos = [
    {'id': 10, 'dato': '...'},
    {'id': 11, 'dato': '...'},
    {'id': 12, 'dato': '...'},
    {'id': 10, 'dato': '...'},
    {'id': 11, 'dato': '...'},
]

print(datos)
```

[{'id': 10, 'dato': '...'}, {'id': 11, 'dato': '...'}, {'id': 12, 'dato': '...'}, {'id': 10, 'dato': '...'}, {'id': 11, 'dato': '...'}]

```
# La forma tradicional
objetos_unicos = []
for d in datos:
    dato_existe = False
    for ou in objetos_unicos:
        if ou['id'] == d['id']:
            dato_existe = True
            break
    if not dato_existe:
        objetos_unicos.append(d)
```

```
print(objetos_unicos)
```

```
[{'id': 10, 'dato': '...'}, {'id': 11, 'dato': '...'}, {'id': 12, 'dato': '...'}]
```

```
# Una mejor manera.  
objetos_unicos_easy = { d['id']:d for d in datos }.values()  
  
print(list(objetos_unicos_easy))
```

```
[{'id': 10, 'dato': '...'}, {'id': 11, 'dato': '...'}, {'id': 12, 'dato': '...'}]
```

17.13 Ejemplo 22.

Sea un diccionario que tiene como claves letras minúsculas y mayúsculas, y como valores números enteros:

```
mcase = {'z':23, 'a':30, 'b':21, 'A':78, 'Z':4, 'C':43, 'B':89}
```

- Sumar los valores que corresponden a la misma letra, mayúscula y minúscula.
- Construir un diccionario cuyas claves sean solo letras minúsculas y sus valores sean la suma antes calculada.

```
mcase = {'z':23, 'a':30, 'b':21, 'A':78, 'Z':4, 'C':43, 'B':89}  
mcase_freq = {k.lower() :  
               mcase.get(k.lower(), 0) + mcase.get(k.upper(), 0)  
               for k in mcase.keys()}  
print(mcase_freq)
```

```
{'z': 27, 'a': 108, 'b': 110, 'c': 43}
```

17.14 Ejemplo 23.

Es posible usar las listas y diccionarios concisos para revisar la lista de archivos de un directorio y sus características.

```
import os, glob  
metadata = [(f, os.stat(f)) for f in glob.glob('*.ipynb')]  
metadata
```

```
[('Pensando_como_pythonista_1.ipynb',
  os.stat_result(st_mode=33188, st_ino=2172678637, st_dev=2097240, st_nlink=1,
  st_uid=1000, st_gid=100, st_size=29456, st_atime=1705689214, st_mtime=1705689214,
  st_ctime=1705689214)),
 ('Pythonico_es_mas_bonito_1.ipynb',
  os.stat_result(st_mode=33188, st_ino=2172678648, st_dev=2097240, st_nlink=1,
  st_uid=1000, st_gid=100, st_size=39608, st_atime=1705689214, st_mtime=1705689214,
  st_ctime=1705689214)),
 ('Pythonico_es_mas_bonito_2.ipynb',
  os.stat_result(st_mode=33188, st_ino=2172678651, st_dev=2097240, st_nlink=1,
  st_uid=1000, st_gid=100, st_size=29765, st_atime=1705689214, st_mtime=1705689214,
  st_ctime=1705689214)),
 ('T02_Expr_Decla.ipynb',
  os.stat_result(st_mode=33188, st_ino=2172811400, st_dev=2097240, st_nlink=1,
  st_uid=1000, st_gid=100, st_size=8386, st_atime=1709740171, st_mtime=1709683460,
  st_ctime=1709683460)),
 ('T03_TiposBasico_Operadores.ipynb',
  os.stat_result(st_mode=33188, st_ino=2173059438, st_dev=2097240, st_nlink=1,
  st_uid=1000, st_gid=100, st_size=35254, st_atime=1710044492, st_mtime=1709321664,
  st_ctime=1709321664)),
 ('T04_Cadenas.ipynb',
  os.stat_result(st_mode=33188, st_ino=2174792627, st_dev=2097240, st_nlink=1,
  st_uid=1000, st_gid=100, st_size=18314, st_atime=1709510973, st_mtime=1709323123,
  st_ctime=1709323123)),
 ('T00_Otros.ipynb',
  os.stat_result(st_mode=33188, st_ino=2173586078, st_dev=2097240, st_nlink=1,
  st_uid=1000, st_gid=100, st_size=6513, st_atime=1710044528, st_mtime=1710044528,
  st_ctime=1710044528)),
 ('T01_Etiquetas_y_Palabras_Reservadas.ipynb',
  os.stat_result(st_mode=33188, st_ino=2175783890, st_dev=2097240, st_nlink=1,
  st_uid=1000, st_gid=100, st_size=20284, st_atime=1709740171, st_mtime=1709666919,
  st_ctime=1709666919)),
 ('T05_Estructura_de_Datos.ipynb',
  os.stat_result(st_mode=33188, st_ino=2172811396, st_dev=2097240, st_nlink=1,
  st_uid=1000, st_gid=100, st_size=44066, st_atime=1709567068, st_mtime=1709566559,
  st_ctime=1709566559)),
 ('T06_Control_de_flujo.ipynb',
  os.stat_result(st_mode=33188, st_ino=2172811404, st_dev=2097240, st_nlink=1,
  st_uid=1000, st_gid=100, st_size=27294, st_atime=1709598006, st_mtime=1709598006,
  st_ctime=1709598006)),
 ('zT10_Comprehensions.ipynb',
  os.stat_result(st_mode=33188, st_ino=2172811425, st_dev=2097240, st_nlink=1,
  st_uid=1000, st_gid=100, st_size=49575, st_atime=1710099347, st_mtime=1710099347,
  st_ctime=1710099347)),
 ('zT11_IteradoresGeneradores.ipynb',
  os.stat_result(st_mode=33188, st_ino=2172811428, st_dev=2097240, st_nlink=1,
  st_uid=1000, st_gid=100, st_size=11449, st_atime=1710044551, st_mtime=1710024222,
  st_ctime=1710024222)),
 ('zT12_Decoradores.ipynb',
  os.stat_result(st_mode=33188, st_ino=2172812549, st_dev=2097240, st_nlink=1,
```



```

st_uid=1000, st_gid=100, st_size=13349, st_atime=1709600117, st_mtime=1705689214,
st_ctime=1709596220)),
('zT13_BibliotecaEstandar.ipynb',
 os.stat_result(st_mode=33188, st_ino=2172812551, st_dev=2097240, st_nlink=1,
st_uid=1000, st_gid=100, st_size=15600, st_atime=1709679822, st_mtime=1709679822,
st_ctime=1709679822)),
('T08_Archivos_Gestores_de_contexto.ipynb',
 os.stat_result(st_mode=33188, st_ino=2172811408, st_dev=2097240, st_nlink=1,
st_uid=1000, st_gid=100, st_size=5865, st_atime=1709600072, st_mtime=1709600072,
st_ctime=1709600072)),
('T07_Entrada_salida_estandar.ipynb',
 os.stat_result(st_mode=33188, st_ino=2173013626, st_dev=2097240, st_nlink=1,
st_uid=1000, st_gid=100, st_size=12347, st_atime=1709599786, st_mtime=1709599786,
st_ctime=1709599786)),
('T09_Funciones_y_docstring.ipynb',
 os.stat_result(st_mode=33188, st_ino=2172811410, st_dev=2097240, st_nlink=1,
st_uid=1000, st_gid=100, st_size=33782, st_atime=1709774198, st_mtime=1709668427,
st_ctime=1709668427)),
('T11_Excepciones.ipynb',
 os.stat_result(st_mode=33188, st_ino=2172811414, st_dev=2097240, st_nlink=1,
st_uid=1000, st_gid=100, st_size=34820, st_atime=1709740171, st_mtime=1709680686,
st_ctime=1710038069)),
('T10_LambdaExpressions.ipynb',
 os.stat_result(st_mode=33188, st_ino=2172811418, st_dev=2097240, st_nlink=1,
st_uid=1000, st_gid=100, st_size=15325, st_atime=1710043865, st_mtime=1710043865,
st_ctime=1710043865)),
('T12_IterablesMapFilter.ipynb',
 os.stat_result(st_mode=33188, st_ino=2172811416, st_dev=2097240, st_nlink=1,
st_uid=1000, st_gid=100, st_size=32836, st_atime=1710090534, st_mtime=1710090534,
st_ctime=1710090534)))]

```

```
metadata_dict = {f:os.stat(f) for f in glob.glob('*.ipynb')}
```

```
metadata_dict.keys()
```

```

dict_keys(['Pensando_como_pythonista_1.ipynb', 'Pythonico_es_mas_bonito_1.ipynb',
'Pythonico_es_mas_bonito_2.ipynb', 'T02_Expr_Decla.ipynb',
'T03_TiposBasico_Operadores.ipynb', 'T04_Cadenas.ipynb', 'T00_Otros.ipynb',
'T01_Etiquetas_y_Palabras_Reservadas.ipynb', 'T05_Estructura_de_Datos.ipynb',
'T06_Control_de_flujo.ipynb', 'zT10_Comprehensions.ipynb',
'zT11_IteradoresGeneradores.ipynb', 'zT12_Decoradores.ipynb',
'zT13_BibliotecaEstandar.ipynb', 'T08_Archivos_Gestores_de_contexto.ipynb',
'T07_Entrada_salida_estandar.ipynb', 'T09_Funciones_y_docstring.ipynb',
'T11_Excepciones.ipynb', 'T10_LambdaExpressions.ipynb', 'T12_IterablesMapFilter.ipynb'])

```

```
metadata_dict['T02_Expr_Decla.ipynb'].st_size
```

8386