T10_Comprehensions

January 29, 2021

1 Python de cero a experto

Autor: Luis Miguel de la Cruz Salas

Python de cero a experto by Luis M. de la Cruz Salas is licensed under Attribution-NonCommercial-NoDerivatives 4.0 International

1.1 Pythonico es más bonito: Pensando como pythonista (intermedio)

1.1.1 List comprehensions

- La herramienta de *list comprehensions* (generación corta de listas) puede ser usada para construir listas de una manera muy concisa, natural y fácil, como lo hace un matemático.
- En matemáticas podemos definir un conjunto como sigue:

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

¿Cómo hacemos lo mismo en Python?

[1]:
```
S = [x**2 for x in range(10)]
S
```

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

[2]:
```
type(S)
```

[2]:
```
list
```

Definición.

```
[ out_expression for name in secuence if predicate ]
```

1. Una secuencia de entrada (S) :
   - `M = [math.sqrt(x) for x in` **`S`** `if x%2 == 0]`
2. Un nombre (x) que representa los miembros de la secuencia de entrada:
   - `M = [math.sqrt(x) for` **`x`** `in S if x%2 == 0]`
3. Una expresión de predicado (if x % 2 == 0) opcional:
   - `M = [math.sqrt(x) for x in S` **`if x % 2 == 0`** `]`
4. Una expresión de salida (math.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:

- `M = [math.sqrt(x) for x in S if x % 2 == 0]`

[3]:
```
import math
M = [math.sqrt(x) for x in range(0,10) if x%2 == 0]
print(M)
```

```
[0.0, 1.4142135623730951, 2.0, 2.449489742783178, 2.8284271247461903]
```

Ejercicio 1. 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+')]
```

Ejercicio 2. Crear la siguiente lista:

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

Ejercicio 3. Crear la siguiente lista:

$$M = \{\sqrt{x} \,|\, x \in S \text{ y } x \text{ par } \} = [0, 4, 16, 36, 64]$$

con

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

1.1.2 Anidado de *list comprehensions*

Ejemplo 1. Una matriz identidad de tamaño $n \times n$:

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

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:

[4]:
```
n = 5
[[1 if col == row else 0 for col in range(0,n)] for row in range(0,n)]
```

[4]:
```
[[1, 0, 0, 0, 0],
 [0, 1, 0, 0, 0],
 [0, 0, 1, 0, 0],
 [0, 0, 0, 1, 0],
 [0, 0, 0, 0, 1]]
```

[5]:
```
n = 5
imat = [[1 if col == row else 0 for col in range(0,n)] for row in range(0,n)]
imat
```

[5]:
```
[[1, 0, 0, 0, 0],
 [0, 1, 0, 0, 0],
 [0, 0, 1, 0, 0],
 [0, 0, 0, 1, 0],
 [0, 0, 0, 0, 1]]
```

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

[6]:
```
noprimos = [j for i in range(2,8) for j in range(i*2, 50, i)]
print(noprimos)
```

```
[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]
```

[7]:
```
primos = [x for x in range(2,50) if x not in noprimos]
print(primos)
```

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

[8]:
```
[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, i)]]
```

[8]:
```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

***list comprehension* con elementos diferentes de números** Las listas también pueden contener otro tipo de elementos, no solo números:

[9]:
```
mensaje = 'La vida no es la que uno vivió, sino la que uno recuerda'
```

[10]:
```
print(mensaje)
```

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

[11]:
```
palabras = mensaje.split()
print(palabras,end='')
```

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

[12]:
```
tabla = [(w.upper(), w.title(), len(w)) for w in palabras]
tabla
```

[12]:
```
[('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)]
```

[13]:
```
type(tabla[0])
```

[13]:
```
tuple
```

Ejemplo 3. Transformar grados Celsius en Fahrenheit y viceversa.

[14]:
```
c = [0, 22.5, 40,100]
f = [(9/5)*t + 32 for t in c]
f
```

[14]:
```
[32.0, 72.5, 104.0, 212.0]
```

[15]:
```
cn = [(5/9)*(t - 32) for t in f]
cn
```

[15]:
```
[0.0, 22.5, 40.0, 100.0]
```

1.1.3 Set comprehensions

Permite crear conjuntos usando los mismos principios que las *list comprehensions*, la única diferencia es que la secuencia que resulta es un **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 el conjunto de entrada. (**requerido**).
4. `predicate` : Expresión **opcional** que actúa como un filtro sobre los miembros del conjunto de entrada.
5. `[, …]]` : Otra *comprehension* anidada **opcional**.

Ejemplo 4. Supongamos que deseamos organizar una lista de nombres de tal manera que no haya repeticiones, que los nombres tengan más de un caracter 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']
```

[16]:
```
archivo = open('nombres','r')
lista_nombres = archivo.read().split()
print(lista_nombres)
```

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

[17]:
```
nombres_ordenados = {nombre[0].upper() + nombre[1:].lower()
                     for nombre in lista_nombres if len(nombre) > 1 }
print(nombres_ordenados)
```

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

Ejercicio 4: Explicar los siguientes ejemplos.

[18]:
```
{s for s in [1, 2, 1, 0]}
```

[18]:
```
{0, 1, 2}
```

[19]:
```
{s**2 for s in [1, 2, 1, 0]}
```

[19]:
```
{0, 1, 4}
```

[20]:
```
{s**2 for s in range(10)}
```

[20]:
```
{0, 1, 4, 9, 16, 25, 36, 49, 64, 81}
```

[21]:
```
{s for s in range(10) if s % 2}
```

[21]:
```
{1, 3, 5, 7, 9}
```

[22]:
```
{(m, n) for n in range(2) for m in range(3, 5)}
```

[22]: {(3, 0), (3, 1), (4, 0), (4, 1)}

Dictionary Comprehensions

- 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.

1.1.4 Repaso de diccionarios:

[23]:
```
# Los diccionarios tienes claves (key).
dict1 = {'a': 1, 'b': 2, 'c': 3, 'd': 4}
dict1.keys() # Función para obtener las claves
```

[23]: dict_keys(['a', 'b', 'c', 'd'])

[24]:
```
# Los diccionarios tienen un valor por cada clave.
dict1.values() # Función para obtener los valores
```

[24]: dict_values([1, 2, 3, 4])

[25]:
```
# Los diccionarios tienen entradas (items) que consisten
# de una key y un valor.
dict1.items() # Función para obtener los items
```

[25]: dict_items([('a', 1), ('b', 2), ('c', 3), ('d', 4)])

Definición.

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

Ejemplo 5: Duplicar el valor (*value*) de cada entrada (*item*) de un diccionario:

[26]:
```
dict1 = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
dict1_values = {k:v*2 for (k,v) in dict1.items()}
print(dict1_values)
```

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

Ejemplo 6. Duplicar la clave (*key*) de cada entrada (*item*) del diccionario:

[27]:
```
dict1_keys = {k*2:v for (k,v) in dict1.items()}
print(dict1_keys)
```

```
{'aa': 1, 'bb': 2, 'cc': 3, 'dd': 4, 'ee': 5}
```

Ejemplo 7. 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.

[28]:
```
# 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}
```

[29]:
```
# 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}
```

Ejemplo 8. Intercambiar las claves y los valores en un diccionario.

[30]:
```
a_dict = {'a': 1, 'b': 2, 'c': 3}
{value:key for key, value in a_dict.items()}
```

[30]:
```
{1: 'a', 2: 'b', 3: 'c'}
```

[31]:
```
# 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                                 Traceback (most recent call last)
<ipython-input-31-7a0ef59d6f0f> in <module>
      1 # OJO: No siempre es posible hacer lo anterior.
      2 a_dict = {'a': [1, 2, 3], 'b': 4, 'c': 5}
----> 3 {value:key for key, value in a_dict.items()}

<ipython-input-31-7a0ef59d6f0f> in <dictcomp>(.0)
      1 # OJO: No siempre es posible hacer lo anterior.
      2 a_dict = {'a': [1, 2, 3], 'b': 4, 'c': 5}
----> 3 {value:key for key, value in a_dict.items()}

TypeError: unhashable type: 'list'
```

Ejemplo 9. Convertir Fahrenheit a Celsius y viceversa (again!!!)

[32]:
```
# 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}
```

[33]:
```
# 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}
```

Ejemplo 10. Dado un diccionario, cuyos valores son enteros, crear un nuevo diccionario cuyos valores sean mayores que 2.

[34]:
```
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}
```

Ejemplo 11. Dado un diccionario, cuyos valores son enteros, crear un nuevo diccionario cuyos valores sean mayores que 2 y que además sean pares.

[35]:
```
a_dict_cond2 = { k:v for (k,v) in a_dict.items() if v > 2 if v % 2 == 0}
print(a_dict_cond2)
```

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

Ejemplo 12. 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.

[36]:
```
# 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}
```

[37]:
```
# 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}
```

Ejemplo 13. Apartir de un diccionario con valores enteros, identificar los valores pares y los impares, y sustituir los valores por etiquetas 'par' e 'impar' segun corresponda.

[38]:
```
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'}
```

Ejemplo 14. Dict comprehensions anidados

[39]:
```
# 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}}
```

[40]:
```
# 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}}
```

Ejemplo 15. Eliminar números duplicados de una lista.

[41]:
```
numeros = [i for i in range(1,11)] + [i for i in range(1,6)]
numeros
```

[41]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5]
```

[42]:
```
# Una manera es:
numeros_unicos = []
for n in numeros:
    if n not in numeros_unicos:
        numeros_unicos.append(n)
numeros_unicos
```

[42]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

[43]:
```
# Otra forma mas pythonica!
numeros_unicos_easy = list(set(numeros))
numeros_unicos_easy
```

[43]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Ejemplo 16. Eliminar objetos duplicados de una lista de diccionarios.

[44]:
```
datos = [
  {'id': 10, 'dato': '...'},
  {'id': 11, 'dato': '...'},
  {'id': 12, 'dato': '...'},
  {'id': 10, 'dato': '...'},
  {'id': 11, 'dato': '...'},
]

datos
```

[44]:
```
[{'id': 10, 'dato': '…'},
 {'id': 11, 'dato': '…'},
 {'id': 12, 'dato': '…'},
 {'id': 10, 'dato': '…'},
 {'id': 11, 'dato': '…'}]
```

[45]:
```
# 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)

objetos_unicos
```

[45]:
```
[{'id': 10, 'dato': '…'},
 {'id': 11, 'dato': '…'},
 {'id': 12, 'dato': '…'}]
```

[46]:
```
# The very best!!
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':
'…'}]
```

Ejemplo 17. 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.

[47]:
```
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}
```

Ejemplo 18.

[48]:
```
import os, glob
metadata = [(f, os.stat(f)) for f in glob.glob('*.ipynb')]
metadata
```

[48]:
```
[('T09_LambdaExpressions_Reduce.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459373, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=17198, st_atime=1611952490,
st_mtime=1611952490, st_ctime=1611952490)),
 ('T13_BibliotecaEstandar.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459377, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=16655, st_atime=1611787644,
st_mtime=1611787644)),
```

```
 ('T08_IterablesMapFilter.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459372, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=10100, st_atime=1611952369,
st_mtime=1611952369, st_ctime=1611952369)),
 ('T07_Excepciones.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459371, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=11368, st_atime=1611952319,
st_mtime=1611952319, st_ctime=1611952319)),
 ('T10_Comprehensions.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459374, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=40858, st_atime=1611952494,
st_mtime=1611787649, st_ctime=1611787649)),
 ('T01_Etiquetas_y_Palabras_Reservadas.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459357, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=7269, st_atime=1611949503,
st_mtime=1611949503, st_ctime=1611949503)),
 ('AtractorDeLorenz.ipynb',
  os.stat_result(st_mode=33204, st_ino=1968285, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=84714, st_atime=1611950997,
st_mtime=1611942389, st_ctime=1611942389)),
 ('T02_Expr_Decla_Tipos_Oper.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459358, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=29065, st_atime=1611949512,
st_mtime=1611949512, st_ctime=1611949512)),
 ('T14_Numpy.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459427, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=26171, st_atime=1611856934,
st_mtime=1611856625, st_ctime=1611856625)),
 ('T12_Decoradores.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459376, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=15368, st_atime=1611787646,
st_mtime=1611787646, st_ctime=1611787646)),
 ('01_Pensando_como_pythonista_1.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459165, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=14699, st_atime=1611948889,
st_mtime=1611946215, st_ctime=1611946215)),
 ('04_ComputoCientifico.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459368, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=80492, st_atime=1611942139,
st_mtime=1611942139, st_ctime=1611942139)),
 ('T03_Estructura_de_Datos.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459359, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=20255, st_atime=1611950028,
st_mtime=1611950028, st_ctime=1611950028)),
 ('02_Pythonico_es_mas_bonito_1.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459340, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=18342, st_atime=1611950966,
```

```
st_mtime=1611950966, st_ctime=1611950966)),
 ('T11_IteradoresGeneradores.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459375, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=15803, st_atime=1611787666,
st_mtime=1611787666, st_ctime=1611787666)),
 ('T06_Funciones_y_Documentacion.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459370, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=15895, st_atime=1611952273,
st_mtime=1611952273, st_ctime=1611952273)),
 ('T04_Control_de_flujo.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459360, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=9473, st_atime=1611950186,
st_mtime=1611950186, st_ctime=1611950186)),
 ('T05_Entrada_Salida_Archivos.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459369, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=9589, st_atime=1611950501,
st_mtime=1611950500, st_ctime=1611950500)),
 ('03_Pythonico_es_mas_bonito_2.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459400, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=3691, st_atime=1611950981,
st_mtime=1611787640, st_ctime=1611787640))]
```

[49]:
```
metadata_dict = {f:os.stat(f) for f in glob.glob('*.ipynb')}
```

[50]:
```
metadata_dict.keys()
```

[50]:
```
dict_keys(['T09_LambdaExpressions_Reduce.ipynb', 'T13_BibliotecaEstandar.ipynb',
'T08_IterablesMapFilter.ipynb', 'T07_Excepciones.ipynb',
'T10_Comprehensions.ipynb', 'T01_Etiquetas_y_Palabras_Reservadas.ipynb',
'AtractorDeLorenz.ipynb', 'T02_Expr_Decla_Tipos_Oper.ipynb', 'T14_Numpy.ipynb',
'T12_Decoradores.ipynb', '01_Pensando_como_pythonista_1.ipynb',
'04_ComputoCientifico.ipynb', 'T03_Estructura_de_Datos.ipynb',
'02_Pythonico_es_mas_bonito_1.ipynb', 'T11_IteradoresGeneradores.ipynb',
'T06_Funciones_y_Documentacion.ipynb', 'T04_Control_de_flujo.ipynb',
'T05_Entrada_Salida_Archivos.ipynb', '03_Pythonico_es_mas_bonito_2.ipynb'])
```

[51]:
```
metadata_dict['01_Pensando_como_pythonista_1.ipynb'].st_size
```

[51]:
```
14699
```

[ ]:
```

```