02_Pythonico_es_mas_bonito_1

January 29, 2021

1 Python de cero a experto

Autor: Luis Miguel de la Cruz Salas

Objetivos. Revisar los conceptos de etiquetas, palabras reservadas, expresiones, declaraciones, tipos y operadores, estructura de datos, control de flujo, entrada y salida estándar.

1.1 El huevo cocido perfecto

¿Cuál es el tiempo ideal para cocinar un huevo?

Para lograr un **huevo cocido** suave, la clara debe haberse calentado el tiempo suficiente para coagular a una temperatura superior a 63 ^{o}C, pero la yema no debe calentarse por encima de 70 ^{o}C.

Para lograr un **huevo duro**, el centro de la yema debe de alcanzar los 70 ^{o}C.

1.1.1 Fórmula para calcular el tiempo de cocción.

La siguiente fórmula expresa el tiempo t, en segundos, que le toma a la yema alcanzar la temperatura T_y, en grados Celsius.

$$t = \frac{M^{2/3} c \rho^{1/3}}{K \pi^2 (4\pi/3)^{2/3}} \ln \left[0.76 \frac{T_o - T_w}{T_y - T_w} \right]$$

donde las propiedades son:

- M masa;
- ρ densidad;
- c capacidad calorífica específica;
- K conductividad térmica;
- T_w es la temperatura de ebullición del agua;
- T_o es la temperatura original del huevo antes de meterlo al agua;
- T_y es la temperatura que debe alcanzar la yema.

1.1.2 Ejercicio 1.

Calcular el tiempo de cocción necesario para un huevo duro pequeño de 47 g (la masa de un huevo grande es 67 g), para cuando la temperatura inicial del huevo es: 1. Temperatura ambiente: $T_o = 20\ ^oC$. 2. Temperatura en el refrigerador: $T_o = 4\ ^oC$.

Datos: - $M = 47$ g - $\rho = 1.038$ g / cm^3 - $c = 3.7$ J / g K - $K = 5.4 \times 10^{-3}$ W / cm K - $T_w = 100\ ^oC$ - $T_y = 70\ ^oC$

SOLUCIÓN. Primera parte de la fórmula: $M^{2/3}c\rho^{1/3}$

[1]:
```
# Definir los datos iniciales
M   = 47    # Un entero
rho = 1.038 # Un flotante
c   = 3.7   # Otro flotante

print(type(M), type(rho), type(c))
print(id(M), id(rho), id(c))
print(M, rho, c)
```

```
<class 'int'> <class 'float'> <class 'float'>
94298477671168 140029636550864 140029586043600
47 1.038 3.7
```

[2]:
```
numerador = M**(2/3) * c * rho**(1/3)
numerador
```

[2]:
```
48.790216719661984
```

Revisar: Etiquetas y palabras reservadas

Podríamos usar caracteres matemáticos:

[3]:
```
print(chr(0x1D70C))
```

[4]:
```
  = 1.038
```

[5]:
```
M**(2/3) * c *  **(1/3)
```

[5]:
```
48.790216719661984
```

Revisar: Expresiones, Declaraciones, Tipos y Operadores

Segunda parte de la fórmula: $K\pi^2(4\pi/3)^{2/3}$

[6]:
```
# Denominador, necesitamos el valor de Pi.
import math
# Se puede importar la biblioteca math y ponerle otro nombre
#import math as m
```

```
# Se puede importat solo una función de la biblioteca y ponerle un nombre
#from math import sinh as senoHiperbolico
```

[7]:
```
K = 5.4e-3
denominador = K * math.pi**2 * (4 * math.pi / 3)**(2/3)
```

[8]:
```
print(denominador)
```

```
0.13849026450902358
```

[9]:
```
chr(0x03C0)
```

[9]:
```
' '
```

[10]:
```
  = math.pi
K *  **2 * (4 *   / 3)**(2/3)
```

[10]:
```
0.13849026450902358
```

Tercera parte de la fórmula: $\ln\left[0.76\dfrac{T_o - T_w}{T_y - T_w}\right]$

Caso 1. $T_o = 20\ ^o\text{C}$.

[11]:
```
To = 20  # Temp. ambiente
Tw = 100
Ty = 70
```

[12]:
```
logaritmo = math.log(0.76 * (To - Tw) / (Ty - Tw))
```

Fórmula completa:

[13]:
```
t = numerador / denominador * logaritmo
print(t)
```

```
248.86253747844736
```

Caso 2. $T_o = 4\ ^o\text{C}$.

[14]:
```
To = 4  # Temp. ambiente
Tw = 100
Ty = 70
```

[15]:
```
logaritmo = math.log(0.76 * (To - Tw) / (Ty - Tw))
```

[16]:
```
t = numerador / denominador * logaritmo
print(t)
```

```
313.09454902221637
```

1.1.3 Ejercicio 2.

Imprimir el tiempo de los casos del ejercicio 1 en el siguiente formato usando *cadenas*: `El tiempo de cocción óptimo es: 313.1 [s] (5.2 [m])`

[17]:
```
To = 4
logaritmo = math.log(0.76 * (To - Tw) / (Ty - Tw))
t1 = numerador /denominador * logaritmo

To = 20
logaritmo = math.log(0.76 * (To - Tw) / (Ty - Tw))
t2 = numerador /denominador * logaritmo
```

[18]:
```
# Solución 1:
print('El tiempo de cocción óptimo es:', t1,' [s] (', t1/60, ' [m])')
print('El tiempo de cocción óptimo es:', t2,' [s] (', t2/60, ' [m])')
```

```
El tiempo de cocción óptimo es: 313.09454902221637  [s] ( 5.218242483703606
[m])
El tiempo de cocción óptimo es: 248.86253747844736  [s] ( 4.147708957974123
[m])
```

[19]:
```
# Solución 2:
cadena1 = 'El tiempo de cocción óptimo es: '
cadena2 = str(t1) + ' [s] (' + str(t1) + ' [m])'
cadena3 = str(t2) + ' [s] (' + str(t2) + ' [m])'
print(cadena1 + cadena2)
print(cadena1 + cadena3)
```

```
El tiempo de cocción óptimo es: 313.09454902221637 [s] (313.09454902221637 [m])
El tiempo de cocción óptimo es: 248.86253747844736 [s] (248.86253747844736 [m])
```

[20]:
```
# Solución 3:
cadena2 = '{} [s] ( {} [m])'.format(t1, t1/60)
cadena3 = '{} [s] ( {} [m])'.format(t2, t2/60)
print(cadena1 + cadena2)
print(cadena1 + cadena3)
```

```
El tiempo de cocción óptimo es: 313.09454902221637 [s] ( 5.218242483703606 [m])
El tiempo de cocción óptimo es: 248.86253747844736 [s] ( 4.147708957974123 [m])
```

[21]:
```
# Solución 4:
cadena2 = '{:.1f} [s] ( {:.1f} [m])'.format(t1, t1/60)
cadena3 = '{:.1f} [s] ( {:.1f} [m])'.format(t2, t2/60)
print(cadena1 + cadena2)
print(cadena1 + cadena3)
```

```
El tiempo de cocción óptimo es: 313.1 [s] ( 5.2 [m])
El tiempo de cocción óptimo es: 248.9 [s] ( 4.1 [m])
```

[22]:
```
# Solución 5:
print('El tiempo de cocción óptimo es: {:0.1f} [s] ({:0.1f} [m])'.format(t1, t1/
 ↪60))
print('El tiempo de cocción óptimo es: {:0.1f} [s] ({:0.1f} [m])'.format(t2, t2/
 ↪60))
```

```
El tiempo de cocción óptimo es: 313.1 [s] (5.2 [m])
El tiempo de cocción óptimo es: 248.9 [s] (4.1 [m])
```

1.1.4 Ejercicio 3.

Hacer una lista de tiempos de cocción para temperaturas de huevos, desde la que se tiene en el refrigerador hasta temperatura ambiente, en pasos de 1^oC. 1. Imprimir dos columnas: temperatura y tiempo. 2. Imprimir los tiempos en el formato: min:seg (por ejemplo 5:30). 3. Imprimir un encabezado para identificar las columnas. 4. Realizar lo mismo que en 1, 2 y 3, pero con dos listas `list`: una para las temperaturas y otra para los tiempos. 5. Realizar lo mismo que en 1, 2 y 3, pero con un diccionario `dict`.

Estructura de datos

Control de flujo

SOLUCIÓN. Recordemos la fórmula

$$t = \frac{M^{2/3} c \rho^{1/3}}{K \pi^2 (4\pi/3)^{2/3}} \ln \left[0.76 \frac{T_o - T_w}{T_y - T_w} \right]$$

`numerador` $= M^{2/3} c \rho^{1/3}$

`denominador` $= K \pi^2 (4\pi/3)^{2/3}$

`logaritmo` $= ln \left[0.76 \frac{T_o - T_w}{T_y - T_w} \right]$

Observamos que lo único que cambia es `logaritmo`.

[23]:
```
for To in range(4, 20):
    print(To)
```

```
4
5
6
7
8
9
10
11
12
13
14
15
```

```
16
17
18
19
```

[24]:
```
for To in range(4, 21):
    logaritmo = math.log(0.76 * (To - Tw) / (Ty - Tw))
    t = numerador / denominador * logaritmo
    print(t)
```

```
313.09454902221637
309.4055027800624
305.67741828677
301.9094604759048
298.100767196758
294.2504480302776
290.3575830395756
286.42122145062837
282.44038025843554
278.4140427535251
274.34115696328047
270.22063400211084
266.05134632399177
261.83212587036
257.5617621057524
253.23899993292162
248.86253747844736
```

[25]:
```
for To in range(4, 21):
    logaritmo = math.log(0.76 * (To - Tw) / (Ty - Tw))
    t = numerador / denominador * logaritmo
    print(To, t/60)
```

```
4 5.218242483703606
5 5.156758379667707
6 5.094623638112833
7 5.03182434126508
8 4.968346119945966
9 4.90417413383796
10 4.8392930506595935
11 4.773687024177139
12 4.7073396709739255
13 4.640234045892085
14 4.572352616054674
15 4.503677233368514
16 4.434189105399863
17 4.363868764506
18 4.292696035095873
```

```
19 4.220649998882027
20 4.147708957974123
```

[26]:
```
print('Temperatura \t Tiempo')
for To in range(4, 21):
    logaritmo = math.log(0.76 * (To - Tw) / (Ty - Tw))
    t = numerador / denominador * logaritmo
    t_min = int(t / 60)
    t_seg = int(t - t_min * 60)
    print(' {} \t\t {}:{}'.format(To, t_min, t_seg))
```

```
Temperatura      Tiempo
 4               5:13
 5               5:9
 6               5:5
 7               5:1
 8               4:58
 9               4:54
 10              4:50
 11              4:46
 12              4:42
 13              4:38
 14              4:34
 15              4:30
 16              4:26
 17              4:21
 18              4:17
 19              4:13
 20              4:8
```

[27]:
```
#4. Realizar lo mismo que en 1, 2 y 3, pero con dos listas `list`: una para las␣
↪temperaturas y otra para los tiempos.
print('Temperatura \t Tiempo')
Ts = [] # Lista de temperaturas
ts = [] # Lista de tiempos
for To in range(4,21):
    Ts.append(To)
    logaritmo = math.log(0.76 * (To - Tw) / (Ty - Tw))
    t = numerador / denominador * logaritmo
    t_min = int(t / 60)
    t_seg = int(t - t_min * 60)
    ts.append(str(t_min) + ':' + str(t_seg))

for T, t in zip(Ts, ts):
    print(' {} \t\t {}'.format(T, t))
```

```
Temperatura      Tiempo
 4               5:13
```

```
5               5:9
6               5:5
7               5:1
8               4:58
9               4:54
10              4:50
11              4:46
12              4:42
13              4:38
14              4:34
15              4:30
16              4:26
17              4:21
18              4:17
19              4:13
20              4:8
```

[28]:
```python
#5. Realizar lo mismo que en 1, 2 y 3, pero con un diccionario `dict`.
 ↪print('Temperatura \t Tiempo')
print('Temperatura \t Tiempo')
tiempos_huevo = {} # diccionario vacío
for To in range(4,21):
    logaritmo = math.log(0.76 * (To - Tw) / (Ty - Tw))
    t = numerador / denominador * logaritmo
    t_min = int(t / 60)
    t_seg = int(t - t_min * 60)
    tiempos_huevo[To] = str(t_min) + ':' + str(t_seg)

for key in tiempos_huevo:
    print(' {} \t\t {}'.format(key, tiempos_huevo[key]))
```

```
Temperatura     Tiempo
 4              5:13
 5              5:9
 6              5:5
 7              5:1
 8              4:58
 9              4:54
 10             4:50
 11             4:46
 12             4:42
 13             4:38
 14             4:34
 15             4:30
 16             4:26
 17             4:21
 18             4:17
 19             4:13
```

```
20              4:8
```

Ejemplos de como recorrer diccionarios

[29]:
```
d_items = tiempos_huevo.items()
d_items
```

[29]:
```
dict_items([(4, '5:13'), (5, '5:9'), (6, '5:5'), (7, '5:1'), (8, '4:58'), (9,
'4:54'), (10, '4:50'), (11, '4:46'), (12, '4:42'), (13, '4:38'), (14, '4:34'),
(15, '4:30'), (16, '4:26'), (17, '4:21'), (18, '4:17'), (19, '4:13'), (20,
'4:8')])
```

[30]:
```
for item in tiempos_huevo.items():
    print(item)
```

```
(4, '5:13')
(5, '5:9')
(6, '5:5')
(7, '5:1')
(8, '4:58')
(9, '4:54')
(10, '4:50')
(11, '4:46')
(12, '4:42')
(13, '4:38')
(14, '4:34')
(15, '4:30')
(16, '4:26')
(17, '4:21')
(18, '4:17')
(19, '4:13')
(20, '4:8')
```

[31]:
```
for k, i in tiempos_huevo.items():
    print(k, i)
```

```
4 5:13
5 5:9
6 5:5
7 5:1
8 4:58
9 4:54
10 4:50
11 4:46
12 4:42
13 4:38
14 4:34
15 4:30
16 4:26
```

```
17 4:21
18 4:17
19 4:13
20 4:8
```

[32]:
```
for key in tiempos_huevo.keys():
    print(key)
```

```
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

[33]:
```
for val in tiempos_huevo.values():
    print(val)
```

```
5:13
5:9
5:5
5:1
4:58
4:54
4:50
4:46
4:42
4:38
4:34
4:30
4:26
4:21
4:17
4:13
4:8
```

1.1.5 Ejercicio 4.

Utilice la versión que más le agrade para escribir las temperaturas y los tiempos de cocción de un huevo duro, y modifíquela para que haga lo siguiente: 1. Al principio de la ejecución solicite al usuario: - el nombre de un archivo donde va a guardar la tabla de resultados - el peso del huevo 2. Imprima la tabla de resultados en pantalla. 3. Guarde la tabla en el archivo. 4. Muestre un mensaje al usuario diciendo el nombre del archivo donde se guardó el resultado.

Entrada, Salida y Archivos

[34]:
```
nombre_archivo = input('Nombre del archivo: ')
M = float(input('Peso del huevo = '))
numerador = M**(2/3) * c * rho**(1/3)

print('Temperatura \t Tiempo')
tiempos_huevo = {} # diccionario vacío
for To in range(4,21):
    logaritmo = math.log(0.76 * (To - Tw) / (Ty - Tw))
    t = numerador / denominador * logaritmo
    t_min = int(t / 60)
    t_seg = int(t - t_min * 60)
    tiempos_huevo[To] = str(t_min) + ':' + str(t_seg)

with open(nombre_archivo, 'w') as archivo_abierto:
    archivo_abierto.write('Temperatura \t Tiempo\n')
    for key in tiempos_huevo:
        print(' {} \t\t {}'.format(key, tiempos_huevo[key]))
        archivo_abierto.write(' {} \t\t {}\n'.format(key, tiempos_huevo[key]))

print('La tabla de tiempos de cocción se guardó en el archivo: "{}"'.
 ↪format(nombre_archivo))
```

```
Nombre del archivo:  tabla
Peso del huevo =  67

Temperatura      Tiempo
 4               6:36
 5               6:31
 6               6:27
 7               6:22
 8               6:17
 9               6:12
 10              6:7
 11              6:2
 12              5:57
 13              5:52
 14              5:47
 15              5:42
 16              5:36
```

```
 17              5:31
 18              5:26
 19              5:20
 20              5:15
La tabla de tiempos de cocción se guardó en el archivo: "tabla"
```

1.1.6 Ejercicio 5.

Modificar el código del ejercicio 4 para que el nombre del archivo contenga el peso del huevo. Por ejemplo, si el usuario teclea `tiempo_coccion`, el resultado final se almacenará en un archivo de nombre `tiempo_coccion_67`.

1.1.7 Ejercicio 6.

Calcule la tabla de tiempos para un huevo de 67 gramos. Luego determine la temperatura a la cuál debería estar el huevo originalmente para que el tiempo de cocción sea exactamente de 6 minutos.

1.1.8 Miniproyecto 1.

Calcule las tablas de tiempos para huevos con peso mínimo de 47 y máximo de 67 gramos, en pasos de 1 gramo. Imprima al principio de cada tabla la leyenda: `Peso del huevo = 67 [g]`. Almacenar estas tablas en una lista de diccionarios. Guardar las tablas de manera secuencial en un archivo cuyo nombre debe ser proporcionado por el usuario. Finalmente preguntar al usuario la tabla de tiempos que desea ver (con base en el peso del huevo) en pantalla.