

LA RECETA EN PYTHON

Cátedra Programación II

Agosto 2023

1. Construcción de Programas

Como vimos en **Programación I**, la receta que aplicamos consta de los siguientes pasos:

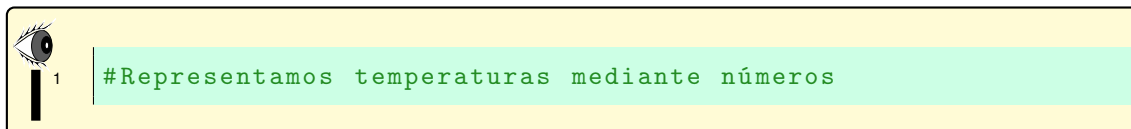
1. diseño de datos;
2. signatura y declaración de proposito;
3. ejemplos;
4. definición de la función;
5. evaluación del código en los ejemplos;
6. realización de modificaciones en caso de que el paso anterior genere errores.

Vamos a ver esto en Python, aplicado al mismo ejemplo visto en Racket:

Escriba un programa que convierta una temperatura medida en un termómetro Fahrenheit a una temperatura en Celsius.

1.1. Item 1: Diseño de datos

¿Cómo representamos la información?




1.2. Item 2: Signatura y declaración de propósito

La signatura de una función indica qué parámetros recibe (cuántos y de qué tipo) y qué datos retorna. La declaración de propósito indica una breve descripción del comportamiento de la función (¿qué hace?).

En el caso de los problemas sencillos que abordaremos, deberemos decidir:

- a) cuáles son los datos de entrada que se nos proveen,
- b) cuáles son las salidas que debemos producir
- c) y cuál es la relación entre todos ellos.


Como veremos, puede pasar que una función en Python no tome argumentos o no retorne valores. Más adelante veremos cómo representar esto.



```
1 #Representamos temperaturas mediante números enteros
2 #farCel: Int -> Int
3 #El parámetro representa una temperatura en Fahrenheit y,
4 # se retorna su equivalente en Celsius.
```

1.3. Item 3: Ejemplos

Luego de los pasos anteriores, podemos saber el resultado de la función para algunos valores de entrada.




```
1 #Representamos temperaturas mediante números enteros
2 #farCel: Int -> Int
3 #El parámetro representa una temperatura en Fahrenheit y,
4 # se retorna su equivalente en Celsius.
5 # entrada: 32, salida: 0
6 # entrada: 212, salida: 100
7 # entrada: -40, salida: -40
```

1.4. Item 4: Definición del programa

¡Escribimos código!

Traducir a un lenguaje de programación (en nuestro caso, y por el momento, Python) el diseño que describimos en el punto anterior.



```
1 #Representamos temperaturas mediante números enteros
2 #farCel: Int -> Int
3 #El parámetro representa una temperatura en Fahrenheit y,
4 # se retorna su equivalente en Celsius.
5 # entrada: 32, salida: 0
6 # entrada: 212, salida: 100
7 # entrada: -40, salida: -40
8 def farCel(f):
9     return (f-32)*5/9
```

1.5. Item 5: Evaluar el código en los ejemplos

¿Qué significa?

Aplicar el código generado en los ejemplos que se diseñaron y verificar que los resultados obtenidos coincidan con lo esperado.



```
>>> farCel(32)
0.0

>>> farCel(212)
100.0

>>> farCel(-40)
-40.0
```

1.6. Item 6: Realizar modificaciones en caso de error

¿Qué significa?

Encontrar los problemas que se hubieran detectado y solucionarlos.


2. Un Ejemplo: Conversión de Unidades Métricas

PROBLEMA 1. Al leer un artículo en una revista que contiene información de longitudes expresadas en millas, pies y pulgadas, correspondientes al sistema anglosajón de unidades no métricas, necesitamos convertir esas distancias a metros. Para ello decidimos escribir un programa que convierta las longitudes del **sistema anglosajón** al **sistema métrico decimal**.

Diseño de datos

En este caso el problema es sencillo: representamos millas, pies, pulgadas y metros como números flotantes (con coma).

Signatura y declaración de propósito




```
1 #Representamos millas, pies, pulgadas y metros como números
2 #convierteAMetros: Float Float Float -> Float
3 #El primer parámetro representa una longitud en millas,
4 #el segundo en pies, el tercero en pulgadas y el valor de
5 #retorno representa su equivalente en metros.
```

Ejemplos

Para construir los ejemplos, tenemos que encontrar la relación existente entre las unidades que tomamos de entrada y la salida que vamos a generar. Para esto, buscando en Internet, encontramos la siguiente tabla:

- 1 milla = 1609.344 m
- 1 pie = 0.3048 m
- 1 pulgada = 0.0254 m

Por lo que podemos decir que



```


1 #Representamos millas, pies, pulgadas y metros como números
2 #convierteAMetros: Float Float Float -> Float
3 #El primer parámetro representa una longitud en millas,
4 #el segundo en pies, el tercero en pulgadas y el valor de
5 #retorno representa su equivalente en metros.
6 #entrada: 1, 0, 0; salida: 1609.344
7 #entrada: 0, 1, 0; salida: 0.3048
8 #entrada: 0, 0, 1; salida: 0.0254
9 #entrada: 1, 1, 1; salida: 1609.6742

```

Definición del programa

Ahora, vamos a escribir el código de la función en Python entendiendo que si una longitud se expresa como L millas, F pies y P pulgadas, su conversión a metros se calculará como:

$$M = 1609.344 \times L + 0.3048 \times F + 0.0254 \times P$$



```

1 #Representamos millas, pies, pulgadas y metros como números
2 #convierteAMetros: Float Float Float -> Float
3 #El primer parámetro representa una longitud en millas,
4 #el segundo en pies, el tercero en pulgadas y el valor de
5 #retorno representa su equivalente en metros.
6 #entrada: 1, 0, 0; salida: 1609.344
7 #entrada: 0, 1, 0; salida: 0.3048
8 #entrada: 0, 0, 1; salida: 0.0254
9 #entrada: 1, 1, 1; salida: 1609.6742
10 def convierteAMetros(numeroMillas, numeroPie, ↵
    numeroPulgadas):
11     metros = 1609.34 * numeroMillas + 0.3048 * numeroPie +↵
        0.0254 * numeroPulgadas
12     return metros

```

Evaluar el código en los ejemplos

Aquí tendríamos que verificar que:



```
>>> convierteAMetros(1,0,0)
1609.344

>>> convierteAMetros(0,1,0)
0.3048

>>> convierteAMetros(0,0,1)
0.0254

>>> convierteAMetros(1,1,1)
1609.6742
```

Con estos ejemplos podemos ver cómo aplicaríamos la receta a cada función que forme parte del programa.