

## Auxiliar 1: Introducción a Funciones

Todos los problemas deben ser resueltos en *Python*, utilizando estrictamente la Receta de Diseño entregada a lo largo del curso. Use nombres apropiados para funciones y variables, y testee cada vez que sea posible.

### 1. Pregunta 1

- Defina una función llamada `invertir3` que, dado un número de tres dígitos, retorne el mismo número dado vuelta. Ejemplo: `invertir(158)` retorna 851
- Ahora defina una función llamada `invertir4` que, dado un número de cuatro dígitos, retorne el mismo número dado vuelta. Ejemplo: `invertir4(1584)` retorna 4851. Fíjese que existen al menos dos formas de implementar esta función: una que hace todo de nuevo, y otra que aprovecha que ya tenemos lista la función `invertir3`.

### 2. Pregunta 2

Una forma común de representar un instante del día es utilizar 4 dígitos, donde los dos primeros dígitos corresponden a la hora y los dos siguientes a los minutos (ejemplo: 1935 corresponde a las 19 horas con 35 minutos, 0314 corresponde a las 3 horas con 14 minutos). Escriba una función que tome dos instantes en formato “HHMM” y retorne la diferencia en formato “HHMM”.

*Indicación 1:* escriba una función que transforme un número en formato “HHMM” a cantidad de minutos, y otra función que transforme minutos a formato “HHMM”.

*Indicación 2:* el resultado debe ser siempre positivo. Para ello, dados los instantes  $i_1$  e  $i_2$ , revise cuál es el mayor y cuál es el menor, y recién entonces proceda a calcular a la diferencia.

*Indicación 3:* En Python, escribir un número con ceros a la izquierda suele traer problemas (utilizar ceros a la izquierda significa utilizar base 8 en vez de base 10, no se preocupe por entender esto). Por ello, en caso de que las horas sean menores a 10, el formato a utilizar será “HMM” (ejemplo: para representar las 9 horas con 42 minutos, escribiremos 942, NO 0942)

### 3. Pregunta 3

Para esta pregunta, usted supondrá que es un/a cajero/a de un banco (o simplemente una persona muy ordenada que quiere andar con la menor cantidad de billetes posible):

1. Se le pide programar una función llamada `billetes20k`, que dada una cantidad de dinero, le dice cuál es la cantidad máxima de billetes de \$20000 que puede tener. Ejemplo: `billetes20k(95000)` retorna 4, ya que puede tener \$80000 en billetes de \$20000, pero el resto deberá tenerlo en billetes de otro valor.
2. Ahora, le solicitan realizar lo mismo, pero con billetes de \$10000 (es decir, una función que se llame `billetes10k`).

3. Nuevamente le piden lo mismo, pero para billetes de \$5000. Usted, ya cansado/a de escribir la misma función para distintos valores de billetes, decide que es momento de crear una función genérica. Llame a esta función `nBilletes(cantidad, valorBillete)`. Note que esta función tomará 2 parámetros: la cantidad de dinero del que dispone, y la nominación del billete.
4. Finalmente, utilizando la función `nBilletes`, programe una función llamada `organizarDinero`, que dada una cantidad de dinero, imprima en pantalla la mínima cantidad de billetes que necesita para tener ese dinero junto a la nominación correspondiente. Ejemplo: `organizarDinero(98000)` imprimirá algo como:
  - 4 billetes de 20000
  - 1 billetes de 10000
  - 1 billetes de 5000
  - 1 billetes de 2000
  - 1 billetes de 1000

Varias cosas que debe notar:

- No es necesario cambiar de singular a plural cuando se tiene 1 o más billetes
- Note que cuando ya calculó la cantidad de billetes de \$20000 necesarios, sólo debe trabajar con el resto (lo mismo cuando ya terminó de trabajar con los billetes de \$10000, etc)
- Asuma que sólo existen billetes de 20000, 10000, 5000, 2000 y 1000 pesos (no se preocupe por las monedas -aunque obviamente, estas son las más apañadoras -)
- Finalmente, lo más importante: note que esta función NO RETORNA NADA, pero aún así obtenemos resultados :D