


Open · Opened 4 days ago by  **Rubén Montero**

Te devuelvo una manzana o una pera, según me cuadre

Resumen

- Comprenderemos que el tipado dinámico hace el código más flexible también a la hora de devolver valores con `return`
- Añadiremos un nuevo fichero de código fuente `example3.py` en `intro/` con una función de ejemplo
- Nos enfrentaremos a problemas *concatenando* valores `int` con `+`
- Usaremos la utilidad propia de Python `str(...)` para solventarlos

Descripción

Hemos visto en la práctica que el **tipado dinámico** es responsable de que una función **admita 'cualquier' cosa** en sus parámetros de entrada.

- Por ejemplo, la función de `maths.py` que suma (`+`) dos valores, puede recibir `int` ó `str` indistintamente, produciendo así *resultados* distintos (**sumar** vs **concatenar** vs **romper**).

Pues bien, el **tipado dinámico**... ¡También hace posible **devolver 'cualquier' cosa** con `return` !

Sentencia `if`

En Python, `if` **no** lleva **paréntesis**.

Emplea **dos puntos** (`:`) e **indentación**.

Las palabras reservadas son `if`, ó `if` y `else`:

```
if valor == "Mi cadena de texto": # Los strings se comparan sin usar .equals
    print("Se ha ejecutado el IF")

# Otro ejemplo
if (valor == "Mi cadena de texto") and (otroValor != 5): # Los operadores booleanos 'and' y 'or' se escriben así directamente
    print("Se ha ejecutado el IF")
else:
    print("Se ha ejecutado el ELSE")
```

¡Bien!

¡Sí!

Vamos allá

Añadamos un nuevo fichero `example3.py` que contenga una función llamada `dental_insurance_cost`.

- Recibirá: Un `string` con el nombre de una compañía de seguros dental
- Devolverá: El coste mensual que oferta dicha compañía

Es decir, dicha función *sabrá* cuánto cuestan cada compañía. Por ejemplo, supongamos que la compañía ficticia [Ratoncito Pérez S.L.](#) cuesta **40** euros al mes:

```
def dental_insurance_cost(company_name):
    if company_name == "Ratoncito Pérez S.L.":
        return "40"
```

Si te fijas, devolvemos el valor `"40"` **como** `string`, fingiendo que nos hemos *equivocado*, por razones de aprendizaje:

Ahora vamos a *suponer que cualquier otra compañía* tiene un precio de **30** euros al mes.

```
def dental_insurance_cost(company_name):
    if company_name == "Ratoncito Pérez S.L.":
        return "40"
    else:
        return 30
```

¿Ves que podemos devolver dos **tipos de dato** distintos?

Ahora, un **main**

Debajo de dicha función, escribamos la sentencia **if** asociada al cuerpo principal del fichero.

Declaremos una variable nueva:

```
if __name__ == '__main__':
    company = "Ratoncito Pérez S.L."
```

...invocamos la función recién creada, para recuperar el valor:

```
if __name__ == '__main__':
    company = "Ratoncito Pérez S.L."
    cost = dental_insurance_cost(company)
```

...y hacemos un *print* que enseña al usuario la información:

```
if __name__ == '__main__':
    company = "Ratoncito Pérez S.L."
    cost = dental_insurance_cost(company)
    print("Tu seguro dental con la compañía " + company + " cuesta " + cost + " euros al mes")
```

Probémoslo

Si **ejecutamos** `example3.py` deberíamos ver por consola algo como:

```
Tu seguro dental con la compañía Ratoncito Pérez S.L. cuesta 40 euros al mes

Process finished with exit code 0
```

¡Bien! Funciona como es esperado.


Ahora, otra compañía

Vamos a **sustituir** el valor de `company` que acabamos de usar, por otro que *predicablemente* costará **30** euros al mes.

Así:

```
if __name__ == '__main__':
    company = "DentaPlus S.L."
    cost = dental_insurance_cost(company)
    print("Tu seguro dental con la compañía " + company + " cuesta " + cost + " euros al mes")
```

Si ejecutamos `example3.py` ...

¿Qué pasa? 

```
Traceback (most recent call last):
  File "C:\Users\Developer\...\python-introduction\intro\example3.py", line 10, in <module>
    print("Tu seguro dental con la compañía " + company + " cuesta " + cost + " euros al mes")
TypeError: can only concatenate str (not "int") to str
```

El **mismo código** que antes funcionaba, ahora **ya no**.

Conclusión

MUST

- Se **debe** verificar y conocer **en profundidad** el código que escribimos
- Cada pocas líneas, debemos **ejecutarlo y probarlo**
- Debido al tipado dinámico, los **errores inesperados** surgen con **frecuencia**. Escribir mucho **código sin probarlo** es sinónimo de **catástrofe**

Ahora, arreglemos esto

Podríamos simplemente modificar el valor de retorno de `dental_insurance_cost` :

```
- return 30
+ return "30"
```

Pero vamos a elegir **otra opción**

`str()`

En Python, podemos usar `str(cualquierCosa)` para convertir `cualquierCosa` a una variable que es un `string` .

En el cuerpo **principal** de `example3.py` , podemos sustituir:

```
- print("Tu seguro dental con la compañía " + company + " cuesta " + cost + " euros al mes")
+ print("Tu seguro dental con la compañía " + company + " cuesta " + str(cost) + " euros al mes")
```

¡Listo!

Por último

Verifica que tu código pasará el `test` asociado a la tarea correctamente.

Haz `commit` y `push` para subir los cambios al repositorio.



[Rubén Montero @ruben.montero](#) changed milestone to [%Sprint 3](#) 4 days ago