Tipos de datos. Ejercicios y soluciones

Ejercicio 1. Cambio de moneda

a. Un país tiene monedas de 1 U, 5 U y 25 U. Si tenemos una cierta cantidad de dinero en una variable, diseña fórmulas para dar el cambio óptimo en las monedas descritas.

```
>>> dinero = 537
>>> mon_1 = dinero % 5
...
```

In [1]:

```
dinero = 537
print("Dinero = ", dinero)

mon_25 = dinero // 25
resto_25 = dinero % 25
mon_5 = resto_25 // 5
mon_1 = resto_25 % 5

print(mon_25, mon_5, mon_1)
```

```
Dinero =  537
21 2 2
```

In [2]:

```
# Comprobación:

print(dinero, " = ", mon_25  * 25 +  mon_5  * 5 +  mon_1 * 1)
```

```
537  =  537
```

b. Completa un programa que pida una cierta cantidad de dinero y dé una salida adecuada:

```
Dame la cantidad de dinero que deseas cambiar: 537

    Desglose de 537 U:
    21 monedas de 25 U
    2 monedas de 5 U
    1 monedas de 1 U
```

In [3]:

```
# Con una salida más clara:

entrada = input("Dame la cantidad de dinero que deseas cambiar: ")
dinero = int(entrada)

mon_25 = dinero // 25
resto_25 = dinero % 25
mon_5 = resto_25 // 5
mon_1 = resto_25 % 5

print("desglose de " + entrada + " U:")
print(mon_25, "monedas de 25 U")
print(mon_5, "monedas de 5 U")
print(mon_1, "monedas de 1 U")
```

```
Dame la cantidad de dinero que deseas cambiar: 459
desglose de 459 U:
18 monedas de 25 U
1 monedas de 5 U
4 monedas de 1 U
```

Ejercicio 2. Hola Edmundo

a. En casi todas partes, es clásico el programa “Hola Mundo”, que simplemente escribe esta frase en la pantalla:

```
print("Hola, mundo")
```

Hazlo en tu consola de Spyder, y haz luego dos versiones o tres más: una, que pregunta el nombre del usuario y lo saluda con dicho nombre:

```
¿Cómo te llamas? Edmundo
Hola, Edmundo
```

In [4]:

```
nombre = input("¿Cómo te llamas? ")
print("Hola, ", nombre)
```

```
¿Cómo te llamas? Cristóbal
Hola,  Cristóbal
```

b. Diseña ahora una segunda versión, que pregunta también la edad y le contesta con su nombre, la edad y los años que cumplirá la próxima vez. (Esto es simplemente para que convierta la edad en un número…)

In [5]:

```
nombre = input("¿Cómo te llamas? ")
edad = input("Dime tu edad: ")

print("Hola, ", nombre, ". En tu próximo cumpleaños, cumplirás ", int(edad) + 1, " años.")
```

```
¿Cómo te llamas? Fernando
Dime tu edad: 21
Hola,  Fernando . En tu próximo cumpleaños, cumplirás  22  años.
```

c. La última versión que te pido, funciona desde la consola del sistema operativo.

```
if __name__ == "__main__":
    ...
```

Esta solución se ha de dar fuera de Jupyter...

Algunas funciones con cadenas de caracteres

Practica con las funciones siguientes, hasta entender bien su funcionamiento:

In [6]:

```
cadena = "esta frase es para practicar"
print(cadena[0], cadena[1], len(cadena))
print(cadena[len(cadena)-1], cadena[-1])
print("hola".upper(), "Hola".lower())
print(cadena[:10], "-", cadena[10:])
frase = "Hola " + "amigo, " + "espero que estés bien."
print(frase)
```

```
e s 28
r r
HOLA hola
esta frase -  es para practicar
Hola amigo, espero que estés bien.
```

Ejercicio 3. Nombres propios con mayúscula

Un nombre propio ha de escribirse con la primera letra mayúscula. Diseña instrucciones para que, en el caso de que se dé con minúscula, lo arregle:

```
nombre_propio = “blacky”
...
print(nombre_propio)
Blacky
```

In [7]:

```
nombre_propio = "blacky"
inicial = nombre_propio[0]
resto = nombre_propio[1:]
nombre_propio = inicial.upper() + resto
print(nombre_propio)
```

```
Blacky
```

Ecuación de segundo grado con raíces reales

Si tenemos los coeficientes (a, b, c) de una ecuación de segundo grado y sabemos que tiene dos raíces reales, expresa instrucciones para calcular dichas fórmulas.

In [8]:

```
import math
a, b, c = 2, -10, 12
discriminante = b**2 - 4*a*c
sol1 = (-b + math.sqrt(discriminante)) / (2*a)
sol2 = (-b - math.sqrt(discriminante)) / (2*a)
print(sol1, sol2)
```

```
3.0 2.0
```

Ecuación de segundo grado con raíces complejas

Ídem., si sabemos que tiene dos raíces complejas.

In [9]:

```
#solución 1:

import math
a, b, c = 1, 2, 5
discriminante = b**2 - 4*a*c
parte_real = -b/(2*a)
parte_imag = math.sqrt(-discriminante) / (2*a)
sol1 = "(" + str(parte_real) + ", " + str(parte_imag) + " i)"
sol2 = "(" + str(parte_real) + ", " + str(-parte_imag) + " i)"
print(sol1, sol2)

# solución 2:

import cmath

a, b, c = 1, 2, 5
discriminante = b**2 - 4*a*c
parte_real = -b/(2*a)
parte_imag = cmath.sqrt(-discriminante) / (2*a)
z1 = complex(parte_real, parte_imag)
z2 = complex(parte_real, -parte_imag)
print(z1, z2)

# solución 3:

sol1 = (-b + cmath.sqrt(discriminante)) / (2*a)
sol2 = (-b - cmath.sqrt(discriminante)) / (2*a)
print(sol1, sol2)
```

```
(-1.0, 2.0 i) (-1.0, -2.0 i)
(-1+2j) (-1-2j)
(-1+2j) (-1-2j)
```