Cátedra 20 -- PRÁCTICA PARA LA SOLEMNE

Problema 0

¿Cuál es la diferencia entre los métodos **upper** e **isupper**?

Ambos son métodos de string y se refieren a mayúsculas (*uppercase*). Sin embargo, el método **upper** transforma un string en mayúsculas (ej. **'hola'.upper()** vale **'HOLA'**), mientras que **isupper** verifica si un string está en mayúsculas (ej. **'hola'.isupper()** vale **False**, mientras que **'HOLA'.isupper()** vale **True**).

¿Para qué sirve **len**?

La función **len** sirve para medir el largo de un string (y de *colecciones* que contienen varios datos).

¿Qué números genera **range(5,19,3)**?

Genera los números **5**, **8**, **11**, **14** y **17**. El siguiente valor de la sucesión, 20, no sería generado, pues 20 ≥ 19 (el segundo argumento de range).

¿Qué repetición o bucle (**for** o **while**) debe usar cuando le dicen *su programa recibe un número entero con el número de inputs adicionales que debe solicitar*?

Conviene usar **for**, puesto que ya sabemos cuántos inputs vendrán.

Problema 1

Contexto. Queremos automatizar el proceso de clasificar las guindas que están siendo cosechadas. Para esto, las guindas pasarán por una máquina que las clasificará usando información de sensores: diámetro, firmeza y calidad de la piel.

Cómo abordar el desafío. Escriba un programa que reciba tres números enteros: el diámetro (en milímetros), la firmeza (puntaje en 0-5) y la calidad de la piel (puntaje en 0-5). Su programa deberá clasificar la cereza como: **export premium**, **export normal**, **domestic** (consumo doméstico) y **rejected** (rechazada).

Lógica para la clasificación:

1. Si el diámetro es mayor o igual a 30 mm, su firmeza 4-5 y su calidad de piel es 5, entonces se etiqueta como **export premium**
2. Si no, si el diámetro es mayor o igual a 25 mm, su firmeza 4-5 y su calidad de piel es 4-5, entonces se etiqueta como **export**
3. Si no, pero si el diámetro es mayor o igual a 20 mm, su firmeza 3-5 y su calidad de piel es 3-5, entonces se etiqueta como **domestic**
4. Si no cumple con las anteriores, se debe etiquetar como **rejected**.

Solución

Este problema es sencillo: las salidas son simples **print** que se ejecutan en condiciones específicas, así que no es necesario pensar en la *variable de respuesta*. Así que nos centramos en el algoritmo.

Primero, obtenemos las variables de entrada

```
diam = int(input())
firm = int(input())
piel = int(input())
```

Luego, pasamos a traducir las condiciones a estructura if-elif-else. El primer **if** quedaría como sigue:

```
diam = int(input())
firm = int(input())
piel = int(input())

if diam >= 30 and 4 <= firm <= 5 and piel == 5:
      print("export premium")
```

Podemos completar los elif y el else siguiendo este mismo esquema:

```
diam = int(input())
firm = int(input())
piel = int(input())

if diam >= 30 and 4 <= firm <= 5 and piel == 5:
      print("export premium")
elif diam >= 25 and 4 <= firm <= 5 and 4 <= piel <= 5:
      print("export")
elif diam >= 20 and 3 <= firm <= 5 and 3 <= piel <= 5:
      print("domestic")
else:
      print("rejected")
```

Con esto, el problema se considera resuelto.

Hay también otras maneras de escribir las guardas (condiciones) de los if y elif. Por ejemplo, si dejamos de comprobar que el puntaje de firmeza y textura de la piel llega hasta 5, entonces podemos escribir:

```
diam = int(input())
firm = int(input())
piel = int(input())

if diam >= 30 and firm >= 4 and piel == 5:
      print("export premium")
elif diam >= 25 and firm >= 4 and piel >= 4:
      print("export")
elif diam >= 20 and firm >= 3 and piel >= 3:
      print("domestic")
else:
      print("rejected")
```

Esta respuesta también está correcta.

(En clases vimos incluso más formas correctas de contestar este problema.)

Problema 2

Contexto. Ha llegado un barco a un puerto chileno y debemos ir a inspeccionarlo. Revisaremos cada contenedor y contaremos cuántas cajas vienen y cuánto pesan.

Cómo abordar el desafío. Su programa debe primero preguntar cuántos contenedores hay en el barco y luego, por cada contenedor, debe primero decir **contenedor nuevo**, luego debe preguntar **cuántas cajas vienen** y, finalmente, debe preguntar el **peso de cada caja** (kg) del contenedor. Después de todo este proceso, el programa debe **imprimir el número de cajas** contabilizadas **y el peso total** de las cajas.

Solución

Primero identificamos los datos que queremos calcular; estos son el **número de cajas** y el **peso total**. Debemos definir variables para ambas, que serán números enteros e iniciarán con cero:

```
total_cajas = 0
total_peso = 0

print(total_cajas)
print(total_peso)
```

Luego pasamos a implementar el algoritmo que nos indican. Primero debemos identificar el número de contenedores:

```
total_cajas = 0
total_peso = 0

contenedores = int(input())

print(total_cajas)
print(total_peso)
```

Luego, como el enunciado dice *por cada contenedor*, incluiremos un for-range:

```
total_cajas = 0
total_peso = 0

contenedores = int(input())
for c in range(contenedores):

print(total_cajas)
print(total_peso)
```

Como por cada contenedor hay que imprimir *contenedor* nuevo y, además, preguntar cuántas cajas hay en el contenedor:

```
total_cajas = 0
total_peso = 0

contenedores = int(input())
for c in range(contenedores):
    print("nuevo contenedor")
    cajas = int(input())

print(total_cajas)
print(total_peso)
```

Ahora estamos en condiciones de sumar las cajas al total de cajas:

```
total_cajas = 0
total_peso = 0

contenedores = int(input())
for c in range(contenedores):
    print("nuevo contenedor")
    cajas = int(input())
    total_cajas += cajas

print(total_cajas)
print(total_peso)
```

Luego, debemos preguntar el peso de *cada caja*, o sea, debemos hacer otro **for** y además pedir un input en cada paso de este nuevo for:

```
total_cajas = 0
total_peso = 0

contenedores = int(input())
for co in range(contenedores):
    print("nuevo contenedor")
    cajas = int(input())
    total_cajas += cajas
    for ca in range(cajas):
        peso = int(input())

print(total_cajas)
print(total_peso)
```

Finalmente, debemos sumar este peso al peso total:

```
total_cajas = 0
total_peso = 0

contenedores = int(input())
for co in range(contenedores):
    print("nuevo contenedor")
    cajas = int(input())
    total_cajas += cajas
    for ca in range(cajas):
        peso = int(input())
        total_peso += peso

print(total_cajas)
print(total_peso)
```

Y con esto, el problema está contestado.
Por cierto, hay más de una solución para este problema. Otra solución sería:

```
total_cajas = 0
total_peso = 0

contenedores = int(input())
for co in range(contenedores):
    print("nuevo contenedor")
    cajas = int(input())
    for ca in range(cajas):
        peso = int(input())
        total_peso += peso
        total_cajas += 1

print(total_cajas)
print(total_peso)
```

Otra solución, aunque **no la recomendamos**, sería:

```
total_cajas = 0
total_peso = 0

for co in range( int(input()) ):
    print("nuevo contenedor")
    for ca in range( int(input()) ):
        total_cajas += 1
        total_peso += int(input())

print(total_cajas)
print(total_peso)
```

Problema 3

Contexto. Necesita descifrar un mensaje encriptado. Los mensajes encriptados son strings con signos **+** y **-** intercalados con otros caracteres, que indican si un caracter se debe incluir en la respuesta. Por ejemplo, el string **+h-o+e+l-a+l+o** se descifra como **hello**, puesto que los **+** anteceden las letras h, e, l, l y o.

Cómo resolver el desafío. Escriba un programa que reciba tales strings y que imprima el mensaje descifrado.

Solución

Primero identificamos el tipo de respuesta que queremos. Vemos que el tipo es un string (el mensaje decodificado). Entonces:

```
deco = ""

print(deco)
```

Ahora pasamos a escribir el algoritmo entre estas líneas. Primero pedimos el mensaje codificado:

```
deco = ""

codif = input()

print(deco)
```

Nuestra estrategia será avanzar por los índices pares del string codificado (**codif**), o sea, los índices 0, 2, 4, 6, 8, etc, puesto que los caracteres en esos índices son + y -:

```
deco = ""

codif = input()
for i in range( 0, len(codif), 2 ):

print(deco)
```

Ahora, debemos chequear el caracter en la posición actual (según el índice **i**):

```
deco = ""

codif = input()
for i in range( 0, len(codif), 2 ):
    caracter = codif[i]

print(deco)
```

Luego, si ese carácter es +, entonces debemos agregar el *siguiente caracter* a la respuesta. El siguiente caracter está en **i+1**. Entonces:

```
deco = ""

codif = input()
for i in range( 0, len(codif), 2 ):
      caracter = codif[i]
      if caracter == "+":
            siguiente = codif[i+1]
            deco = deco + siguiente

print(deco)
```

Y, con esto, el problema queda resuelto.

Otra solución, que no ocupa índices, puede ser así:

- Recorremos el string de izquierda a derecha (esto sería un **for**)
- Recordamos el último caracter visto
- Si el último caracter era +, agregamso el caracter actual a la respuesta

La estrategia (algoritmo) anterior se puede implementar así:

```
deco = ""

codif = input()
anterior = ""
for actual in codif:
      if anterior == "+":
            deco = deco + actual
      anterior = actual

print(deco)
```

En este código, es especialmente importante incluir la línea **anterior = actual** (destacada en amarillo), pues nos asegura que recordaremos el caracter actual en el siguiente turno.