# Solucionario Ejercicios Propuestos - Control de flujo y ciclos

# Lectura - Control de Flujo

## Ejercicio propuesto 1: Clasificar Password

```
password = input('Ingrese un Password: ')
# también es posible resolver esto con getpass como sigue
# import getpass
# password = getpass.getpass('Ingrese un Password: ')

if len(password) < 6:
      print('El password ingresado es demasiado corto')
```

## Ejercicio propuesto 2: Password Incorrecto

```
password = input('Ingrese un Password: ')
# también es posible resolver esto con getpass como sigue
# import getpass
# password = getpass.getpass('Ingrese un Password: ')

if password == '12345':
      print('El password es incorrecto')
```

## Ejercicio propuesto 3: Seguridad

El código a) es el correcto, ya que utilizando `elif` es posible determinar de manera correcta a qué categoría de largo pertenece. En el caso del código b), si se tiene un código de menos de 4 letras será categorizado como Pequeña y Mediana ya que ambas condiciones se cumplen y están siendo consultadas de manera independiente, no secuencial.

Para agregar una categoría extra basta con agregar una condición más de la siguiente forma:

```
import getpass
password = getpass.getpass("Ingrese una Contraseña: ")
largo = len(password)

if largo <= 4:
      print("Pequeña")
```

```
elif largo < 10:
      print("Mediana")
elif largo < 15:
      print("Larga")
else:
      print("Muy Larga")
```

## Lectura - El Ciclo While

### Ejercicio propuesto 1: Sumando de 1 a 100

```
i = 1 # primer valor a sumar
suma = 0

while i <= 100:
    suma += i # acumulamos para la suma
    i += 1 # incrementamos para sumar el siguiente valor

print(f"El resultado final es {suma}")
```

Diagrama de flujo para reforzar lo aprendido:

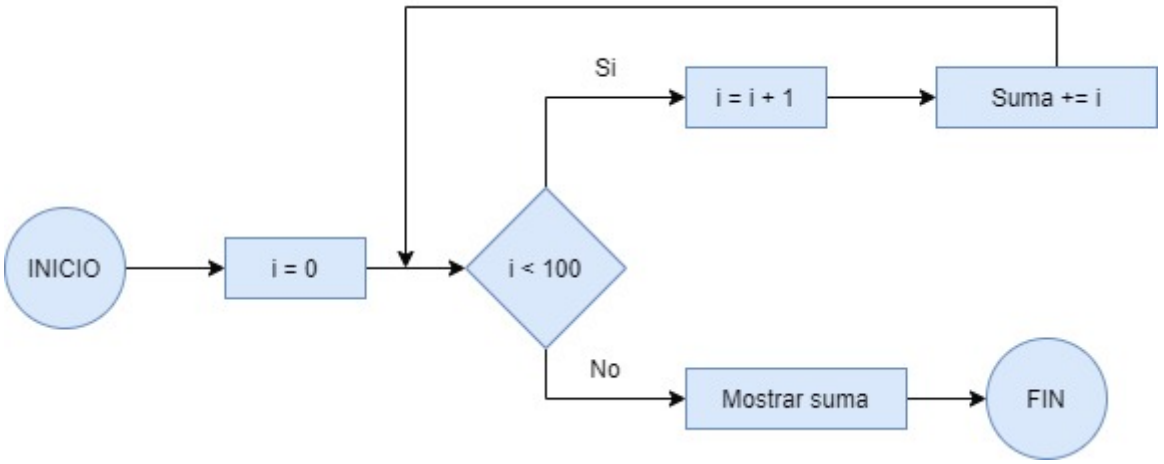


Imagen xx. Diagrama de una sumatoria de 100 números..
Fuente: Desafío Latam

La instrucción `suma += i` es la encargada de aumentar el valor de la variable suma en cada iteración.

### Ejercicio propuesto 2: Generar una lista en HTML

```
import sys

# inicio del html
html = "<ul>\n"
items = int(sys.argv[1])
i = 1 # valor inicial

while i <= items:
    html += f"\t<li> {i} </li>\n"
    i += 1

# final del html
html += "</ul>"
print(html)
```

## Lectura - Python Comprehensions

### Ejercicio propuesto 1: Transformar a Mayúsculas

```
mascotas_mayusculas = [mascota.upper() for mascota in mascotas]
print(mascotas_mayusculas)
```

#### Elevar al cuadrado

```
lista_square = [valor**2 for valor in valores]
print(lista_square)
```

### Ejercicio propuesto 2: Mayúscula sólo Gatos

```
mascotas_mayusculas = [mascota.upper() if mascota == 'gato' else mascota for
mascota in mascotas]
print(mascotas_mayusculas)
```

## Ejercicio propuesto 3: Filtrado

```
print([valor for valor in a if valor >= 1000])
```

## Ejercicio propuesto 4: Adictos a redes

```
tiempos = [120, 50, 600, 30, 90, 10, 200, 0, 500]
print(['bien' if minutos < 90 else 'mal' for minutos in tiempos])
```

## Ejercicio propuesto 5: Adictos v2

```
tiempos = [120, 50, 600, 30, 90, 10, 200, 0, 500]
print(['bien' if minutos < 90 else 'mejorable' if minutos < 180 else
'mal' for minutos in tiempos])
```

Si bien es posible responder a este problema utilizando List Comprehensions notamos que la solución es bastante difícil de leer, ya que ell objetivo de los List Comprehensions es siempre simplificar la respuesta e idealmente entregar algo más sencillo de entender, y en este caso, dado que hay varios intervalos que definir lo ideal es utilizar la instrucción `elif`, la cual no está disponible y por eso termina siendo más complejo. Para este tipo de problema una solución más óptima es utilizar `for`.

```
resultados = []

for minutos in tiempos:
    if minutos < 90:
        resultados.append('bien')
    elif minutos < 180:
        resultados.append('mejorable')
    else:
        resultados.append('mal')
print(resultados)
```

Si bien el código es más largo, es más importante que sea entendible.

## Ejercicio propuesto 6: Transformando segundos a minutos

```
seconds = [100, 50, 1000, 5000, 1000, 500]
# Se utiliza la división entera para desechar la parte decimal
# No es equivalente a redondear
print([tiempos // 60 for tiempos in seconds])
```

## Ejercicio propuesto 7: Países

```
usuarios = {'México': 70, 'Chile': 50, 'Argentina': 55}

print({k: v for k,v in usuarios.items()})
```

## Ejercicio propuesto 8: Ventas

```
ventas = {'Octubre': 65000,
          'Noviembre': 68000,
          'Diciembre': 72000}
# 10% más
print({k: 1.1*v for k,v in ventas.items()})
# 20% menos
print({k: 0.8*v for k,v in ventas.items()})
```

## Ejercicio propuesto 9: Artículos de tienda

```
n_articulos = len(articulos)

# todos los indices pares son claves
claves = [articulos[i] for i in range(0,n_articulos,2)]
# todos los índices impares son valores
valores = [articulos[i] for i in range(1,n_articulos + 1, 2)]

# Notar que los valores son strings.
# además notar que la clave se define sólo una vez
{k: int(v)*0.9 if int(v) > 80000 else int(v) for k,v in zip(claves, valores)}
```

## Ejercicio propuesto 10: Invertir diccionario

```
colores = {
      "red": "#cc0000",
      "green": "#00cc00",
      "blue": "#0000cc",
}

colores_inv = {v:k for k,v in colores.items()}
print(colores_inv["#cc0000"])
```

## Ejercicio propuesto 11: Búsqueda de colores

```
import sys

busqueda = sys.argv[1]
colores = {
  "aliceblue": "#f0f8ff",
  "antiquewhite": "#faebd7",
  "aqua": "#00ffff",
  "aquamarine": "#7fffd4",
  "azure": "#f0ffff",
  "darkorchid": "#9932cc",
  "darkred": "#8b0000",
  "darksalmon": "#e9967a",
  "navajowhite": "#ffdead",
  "navy": "#000080",
  "orchid": "#da70d6",
  "palegoldenrod": "#eee8aa",
  "peachpuff": "#ffdab9",
  "peru": "#cd853f",
  "pink": "#ffc0cb",
  "purple": "#800080",
  "rebeccapurple": "#663399",
  "red": "#ff0000",
  "saddlebrown": "#8b4513",
  "seashell": "#fff5ee",
  "sienna": "#a0522d",
  "silver": "#c0c0c0",
  "skyblue": "#87ceeb",
  "slateblue": "#6a5acd",
  "teal": "#008080",
  "thistle": "#d8bfd8",
```

```
  "tomato": "#ff6347",
  "turquoise": "#40e0d0",
  "violet": "#ee82ee",
  "wheat": "#f5deb3",
  "white": "#ffffff",
  "whitesmoke": "#f5f5f5",
  "yellow": "#ffff00",
  "yellowgreen": "#9acd32",
}
resultado = {v:k for k,v in colores.items()}.get(busqueda,'No se encontró el
color')
print(resultado)
```