

Introducción a Python

Clase 2

Mateo Suster

mateosuster@gmail.com

Matemática para Economistas III

Instituto de Industria

Universidad Nacional de General Sarmiento

6 de mayo de 2022

¿Qué hace el siguiente programa en Python?

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = 34.122 * 1.6
6
7 #imprimo por pantalla el resultado
8 print(mills, "millas son", km, "kilometros")
9
10 #la salida queda
11 > 34.122 millas son 54.595200000000006 kilometros
```

¿Qué hace el siguiente programa en Python?

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = 34.122 * 1.6
6
7 #imprimo por pantalla el resultado
8 print(mills, "millas son", km, "kilometros")
9
10 #la salida queda
11 > 34.122 millas son 54.595200000000006 kilometros
```

¿Qué problemas tiene?

¿Qué hace el siguiente programa en Python?

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = 34.122 * 1.6
6
7 #imprimo por pantalla el resultado
8 print(mills, "millas son", km, "kilometros")
9
10 #la salida queda
11 > 34.122 millas son 54.595200000000006 kilometros
```

¿Qué problemas tiene?

- Duplicación de información.

¿Qué hace el siguiente programa en Python?

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = 34.122 * 1.6
6
7 #imprimo por pantalla el resultado
8 print(mills, "millas son", km, "kilometros")
9
10 #la salida queda
11 > 34.122 millas son 54.595200000000006 kilometros
```

¿Qué problemas tiene?

- **Duplicación de información.** 34.122 lo estoy guardando en mills, pero después vuelvo a poner 34.112 en la conversión a kilómetros

¿Qué hace el siguiente programa en Python?

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = 34.122 * 1.6
6
7 #imprimo por pantalla el resultado
8 print(mills, "millas son", km, "kilometros")
9
10 #la salida queda
11 > 34.122 millas son 54.595200000000006 kilometros
```

¿Qué problemas tiene?

- ▶ **Duplicación de información.** 34.122 lo estoy guardando en mills, pero después vuelvo a poner 34.112 en la conversión a kilómetros
- ▶ ¿Cómo se podría arreglar?

¿Solución?

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 #imprimo por pantalla el resultado
8 print(mills, "millas son", km, "kilometros")
9
10 #la salida queda
11 > 34.122 millas son 54.595200000000006 kilometros
```

¿Solución?

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 #imprimo por pantalla el resultado
8 print(mills, "millas son", km, "kilometros")
9
10 #la salida queda
11 > 34.122 millas son 54.595200000000006 kilometros
```

► Duplicación de información corregida

¿Solución?

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 #imprimo por pantalla el resultado
8 print(mills, "millas son", km, "kilometros")
9
10 #la salida queda
11 > 34.122 millas son 54.595200000000006 kilometros
```

- **Duplicación de información corregida:** 34.122 lo estoy guardando en mills, y después uso la variable mills y no vuelvo a poner 34.112 en la conversión

¿Solución?

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 #imprimo por pantalla el resultado
8 print(mills, "millas son", km, "kilometros")
9
10 #la salida queda
11 > 34.122 millas son 54.595200000000006 kilometros
```

- ▶ **Duplicación de información corregida:** 34.122 lo estoy guardando en mills, y después uso la variable mills y no vuelvo a poner 34.112 en la conversión
- ▶ Sin embargo, la salida no es del todo linda...

¿Qué hace el siguiente programa en Python? (II)

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 round(54.595200000000006, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

¿Qué hace el siguiente programa en Python? (II)

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 round(54.595200000000006, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

► ¿Qué pasó con la salida?

¿Qué hace el siguiente programa en Python? (II)

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 round(54.595200000000006, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

► ¿Qué pasó con la salida? ¿El programa es el mismo?

¿Qué hace el siguiente programa en Python? (II)

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 round(54.595200000000006, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

- ▶ ¿Qué pasó con la salida? ¿El programa es el mismo?
- ▶ ¿Qué problema tiene?

¿Qué hace el siguiente programa en Python? (II)

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 round(54.595200000000006, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

- ▶ ¿Qué pasó con la salida? ¿El programa es el mismo?
- ▶ ¿Qué problema tiene? El resultado de `round(54.595200000000006, 2)` **no se guarda en ningún lado.**

¿Qué hace el siguiente programa en Python? (II)

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 round(54.595200000000006, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

- ▶ ¿Qué pasó con la salida? ¿El programa es el mismo?
- ▶ ¿Qué problema tiene? El resultado de `round(54.595200000000006, 2)` **no se guarda en ningún lado.** ¿Por qué?

¿Qué hace el siguiente programa en Python? (II)

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 round(54.595200000000006, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

- ▶ ¿Qué pasó con la salida? ¿El programa es el mismo?
- ▶ ¿Qué problema tiene? El resultado de `round(54.595200000000006, 2)` **no se guarda en ningún lado**. ¿Por qué? ¿Cómo podría hacerlo?

¿Qué hace el siguiente programa en Python? (II)

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 round(54.595200000000006, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

- ▶ ¿Qué pasó con la salida? ¿El programa es el mismo?
- ▶ ¿Qué problema tiene? El resultado de `round(54.595200000000006, 2)` **no se guarda en ningún lado**. ¿Por qué? ¿Cómo podría hacerlo?
- ▶ ¿Qué otro problema hay?

¿Qué hace el siguiente programa en Python? (II)

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 round(54.595200000000006, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

- ▶ ¿Qué pasó con la salida? ¿El programa es el mismo?
- ▶ ¿Qué problema tiene? El resultado de `round(54.595200000000006, 2)` **no se guarda en ningún lado**. ¿Por qué? ¿Cómo podría hacerlo?
- ▶ ¿Qué otro problema hay? Nuevamente hay duplicación de información: en vez de 54.595200000000006 podemos usar `km` que es la variable que guarda ese valor

Guardo el resultado de redondear y reutilizo su valor en el programa

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 km_redondo = round(km, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

Guardo el resultado de redondear y reutilizo su valor en el programa

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 km_redondo = round(km, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

► ¿Qué hace ahora el programa?

Guardo el resultado de redondear y reutilizo su valor en el programa

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 km_redondo = round(km, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

- ▶ ¿Qué hace ahora el programa? En esencia, ¿es distinto que el programa anterior?

Guardo el resultado de redondear y reutilizo su valor en el programa

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 km_redondo = round(km, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

- ▶ ¿Qué hace ahora el programa? En esencia, ¿es distinto que el programa anterior?
- ▶ ¿Solucionó el problema identificado?

Guardo el resultado de redondear y reutilizo su valor en el programa

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 km_redondo = round(km, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.595200000000006 kilometros
```

- ▶ ¿Qué hace ahora el programa? En esencia, ¿es distinto que el programa anterior?
- ▶ ¿Solucionó el problema identificado? ¿Se utilizó en algún lado la variable `km_redondo`? ¿Cómo podría utilizarlo?

Ahora sí?

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 km_redondo = round(km, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km_redondo, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.59 kilometros
```

Ahora sí?

```
1 #ingreso las millas
2 mills = 34.122
3
4 #hago la conversion a kilometros
5 km = mills * 1.6
6
7 km_redondo = round(km, 2)
8
9 #imprimo por pantalla el resultado
10 print(mills, "millas son", km_redondo, "kilometros")
11
12 #la salida queda
13 > 34.122 millas son 54.59 kilometros
```

- ▶ Este programa, en vez de mostrar (imprimir por pantalla) el valor de `km`, muestra el valor redondeado asignado a la variable `km_redondeado`

Utilizando Listas

```
1 #ingreso las millas
2 mills = [34.122, 17.588, 3.187]
3
4 # Hago la conversion a kilometros
5 km = [mills[0]*1.6, mills[1]*1.6, mills[2]*1.6]
6
7 # Imprimo por pantalla el resultado
8 print(mills[0], "millas son", km[0], "kilometros")
9 print(mills[1], "millas son", km[1], "kilometros")
10 print(mills[2], "millas son", km[2], "kilometros")
```

Utilizando Listas

```
1 #ingreso las millas
2 mills = [34.122, 17.588, 3.187]
3
4 # Hago la conversion a kilometros
5 km = [mills[0]*1.6, mills[1]*1.6, mills[2]*1.6]
6
7 # Imprimo por pantalla el resultado
8 print(mills[0], "millas son", km[0], "kilometros")
9 print(mills[1], "millas son", km[1], "kilometros")
10 print(mills[2], "millas son", km[2], "kilometros")
```

La salida ahora queda:

```
1 > 34.122 millas son 54.5952 kilometros
2 > 17.588 millas son 28.1408 kilometros
3 > 3.187 millas son 5.0992 kilometros
```

Utilizando Listas

```
1 #ingreso las millas
2 mills = [34.122, 17.588, 3.187]
3
4 # Hago la conversion a kilometros
5 km = [mills[0]*1.6, mills[1]*1.6, mills[2]*1.6]
6
7 # Imprimo por pantalla el resultado
8 print(mills[0], "millas son", km[0], "kilometros")
9 print(mills[1], "millas son", km[1], "kilometros")
10 print(mills[2], "millas son", km[2], "kilometros")
```

La salida ahora queda:

```
1 > 34.122 millas son 54.5952 kilometros
2 > 17.588 millas son 28.1408 kilometros
3 > 3.187 millas son 5.0992 kilometros
```

- ¿Cuál es la diferencia fundamental con los programas anteriores?

Utilizando Listas

```
1 #ingreso las millas
2 mills = [34.122, 17.588, 3.187]
3
4 # Hago la conversion a kilometros
5 km = [mills[0]*1.6, mills[1]*1.6, mills[2]*1.6]
6
7 # Imprimo por pantalla el resultado
8 print(mills[0], "millas son", km[0], "kilometros")
9 print(mills[1], "millas son", km[1], "kilometros")
10 print(mills[2], "millas son", km[2], "kilometros")
```

La salida ahora queda:

```
1 > 34.122 millas son 54.5952 kilometros
2 > 17.588 millas son 28.1408 kilometros
3 > 3.187 millas son 5.0992 kilometros
```

- ¿Cuál es la diferencia fundamental con los programas anteriores? Que ahora trabajamos con listas de floats y no con un único valor

Utilizando Listas

```
1 #ingreso las millas
2 mills = [34.122, 17.588, 3.187]
3
4 # Hago la conversion a kilometros
5 km = [mills[0]*1.6, mills[1]*1.6, mills[2]*1.6]
6
7 # Imprimo por pantalla el resultado
8 print(mills[0], "millas son", km[0], "kilometros")
9 print(mills[1], "millas son", km[1], "kilometros")
10 print(mills[2], "millas son", km[2], "kilometros")
```

La salida ahora queda:

```
1 > 34.122 millas son 54.5952 kilometros
2 > 17.588 millas son 28.1408 kilometros
3 > 3.187 millas son 5.0992 kilometros
```

- ▶ ¿Cuál es la diferencia fundamental con los programas anteriores? Que ahora trabajamos con listas de floats y no con un único valor
- ▶ ¿Tiene información duplicada?

Utilizando Listas

```
1 #ingreso las millas
2 mills = [34.122, 17.588, 3.187]
3
4 # Hago la conversion a kilometros
5 km = [mills[0]*1.6, mills[1]*1.6, mills[2]*1.6]
6
7 # Imprimo por pantalla el resultado
8 print(mills[0], "millas son", km[0], "kilometros")
9 print(mills[1], "millas son", km[1], "kilometros")
10 print(mills[2], "millas son", km[2], "kilometros")
```

La salida ahora queda:

```
1 > 34.122 millas son 54.5952 kilometros
2 > 17.588 millas son 28.1408 kilometros
3 > 3.187 millas son 5.0992 kilometros
```

- ▶ ¿Cuál es la diferencia fundamental con los programas anteriores? Que ahora trabajamos con listas de floats y no con un único valor
- ▶ ¿Tiene información duplicada? ¿Cómo se podría evitar?

Estructuras de control

Estructuras de control

- ▶ Las estructuras de control nos permiten modificar el flujo de ejecución del programa.

Estructuras de control

- ▶ Las estructuras de control nos permiten modificar el flujo de ejecución del programa.
- ▶ En otras palabras, quizás más intrincadas, permiten estructurar el proceso de ejecución a partir de ciertas **condiciones lógicas** que definimos.

Estructuras de control

- ▶ Las estructuras de control nos permiten modificar el flujo de ejecución del programa.
- ▶ En otras palabras, quizás más intrincadas, permiten estructurar el proceso de ejecución a partir de ciertas **condiciones lógicas** que definimos.
- ▶ Ejemplo: Condicional `if`

Estructuras de control

- ▶ Las estructuras de control nos permiten modificar el flujo de ejecución del programa.
- ▶ En otras palabras, quizás más intrincadas, permiten estructurar el proceso de ejecución a partir de ciertas **condiciones lógicas** que definimos.
- ▶ Ejemplo: Condicional `if`

```
1 if CONDICION :  
2     PROG1
```

Estructuras de control

- ▶ Las estructuras de control nos permiten modificar el flujo de ejecución del programa.
- ▶ En otras palabras, quizás más intrincadas, permiten estructurar el proceso de ejecución a partir de ciertas **condiciones lógicas** que definimos.
- ▶ Ejemplo: Condicional `if`

```
1 if CONDICION :  
2     PROG1
```

- ▶ `CONDICION` es una expresión que arroja resultado verdadero o falso
- ▶ `PROG1` es un programa que hace algo
- ▶ `PROG1` se ejecuta **si y solo si** `CONDICION` arroja valor verdadero

Estructuras de control

- ▶ Las estructuras de control nos permiten modificar el flujo de ejecución del programa.
- ▶ En otras palabras, quizás más intrincadas, permiten estructurar el proceso de ejecución a partir de ciertas **condiciones lógicas** que definimos.
- ▶ Ejemplo: Condicional `if`

```
1 if CONDICION :  
2     PROG1
```

- ▶ `CONDICION` es una expresión que arroja resultado verdadero o falso
- ▶ `PROG1` es un programa que hace algo
- ▶ `PROG1` se ejecuta **si y solo si** `CONDICION` arroja valor verdadero
- ▶ Atención con el bloque indentado!

Ejemplo de if

```
1 if 1 > 5:  
2     print('1 es mayor que 5')  
3 if 1 < 5:  
4     print('1 es menor que 5')
```


Ejemplo de if

```
1 if 1 > 5:  
2     print('1 es mayor que 5')  
3 if 1 < 5:  
4     print('1 es menor que 5')
```

¿Cuál es su salida?

Ejemplo de if

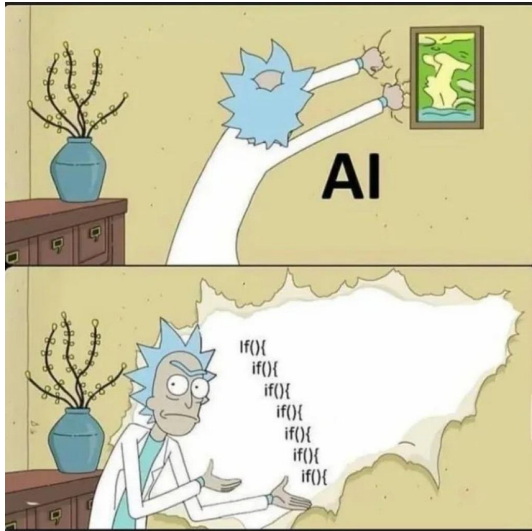
```
1 if 1 > 5:  
2     print('1 es mayor que 5')  
3 if 1 < 5:  
4     print('1 es menor que 5')
```

¿Cuál es su salida?

```
1 > '1 es menor que 5'
```

Con las estructuras de control se puede hacer cualquier cosa...

Con las estructuras de control se puede hacer cualquier cosa...



Otro ejemplo de if

```
1 a = 10
2 b = [100, 1]
3
4 if b[0] // (a * 10) == b[1]:
5     b[0] = b[0] - 1
6     b[1] = b[1] * 5
7
8 print(a, b[0], b[1])
```

Otro ejemplo de if

```
1 a = 10
2 b = [100, 1]
3
4 if b[0] // (a * 10) == b[1]:
5     b[0] = b[0] - 1
6     b[1] = b[1] * 5
7
8 print(a, b[0], b[1])
```

¿Cuál es su salida? 3 minutos para pensarlo...

Otro ejemplo de if

```
1 a = 10
2 b = [100, 1]
3
4 if b[0] // (a * 10) == b[1]:
5     b[0] = b[0] - 1
6     b[1] = b[1] * 5
7
8 print(a, b[0], b[1])
```

¿Cuál es su salida? 3 minutos para pensarlo...

```
1 > 10, 99, 5
```

Condicional: if, elif, else

Condicional: if, elif, else

```
1 if CONDICION1:  
2     PROG1  
3 elif CONDICION2:  
4     PROG2  
5 else:  
6     PROG3
```

Condicional: if, elif, else

```
1 if CONDICION1 :  
2     PROG1  
3 elif CONDICION2 :  
4     PROG2  
5 else :  
6     PROG3
```

- ▶ CONDICION1 y CONDICION2 son expresiones lógicas
- ▶ PROG1, PROG2 y PROG3 son programas
- ▶ PROG1 se ejecuta **si y solo si** CONDICION1 arroja valor TRUE

Condicional: if, elif, else

```
1 if CONDICION1 :  
2     PROG1  
3 elif CONDICION2 :  
4     PROG2  
5 else :  
6     PROG3
```

- ▶ CONDICION1 y CONDICION2 son expresiones lógicas
- ▶ PROG1, PROG2 y PROG3 son programas
- ▶ PROG1 se ejecuta **si y solo si** CONDICION1 arroja valor TRUE
- ▶ De lo contrario, se evalúa CONDICION2 y, **si es verdadera**, se ejecuta PROG2

Condicional: if, elif, else

```
1 if CONDICION1 :  
2     PROG1  
3 elif CONDICION2 :  
4     PROG2  
5 else :  
6     PROG3
```

- ▶ CONDICION1 y CONDICION2 son expresiones lógicas
- ▶ PROG1, PROG2 y PROG3 son programas
- ▶ PROG1 se ejecuta **si y solo si** CONDICION1 arroja valor TRUE
- ▶ De lo contrario, se evalúa CONDICION2 y, **si es verdadera**, se ejecuta PROG2
- ▶ Si CONDICION1 y CONDICION2 arrojan valores FALSE, se ejecuta el PROGRAMA3 de la sentencia else

Ciclos o Bucles

Ciclos o Bucles

```
1 while CONDICION:  
2     PROG1
```

Ciclos o Bucles

```
1 while CONDICION:  
2     PROG1
```

- ▶ CONDICION es una expresión que arroja resultado TRUE o FALSE
- ▶ PROG1 es un programa que hace algo
- ▶ La ejecución de PROG1 se repite **mientras** CONDICION arroja valor TRUE

Ejemplo de while

```
1 i = 0 #arranco la inicializacion en valor = 0
2 while i < 3:
3     print(i) #imprimo por pantalla
4     i = i+1 #muevo el indice una posicion
```


Ejemplo de while

```
1 i = 0 #arranco la inicializacion en valor = 0
2 while i < 3:
3     print(i) #imprimo por pantalla
4     i = i+1 #muevo el indice una posicion
```

¿Cuál es su salida?

Ejemplo de while

```
1 i = 0 #arranco la inicializacion en valor = 0
2 while i < 3:
3     print(i) #imprimo por pantalla
4     i = i+1 #muevo el indice una posicion
```

¿Cuál es su salida?

```
1 > 0
2 > 1
3 > 2
```

Ejemplo de if y while

```
1 i = 0
2 while i < 3:
3     if i % 2 == 0:
4         print(i, 'es par')
5     else:
6         print(i, 'es impar')
7     i = i + 1
```

Ejemplo de if y while

```
1 i = 0
2 while i < 3:
3     if i % 2 == 0:
4         print(i, 'es par')
5     else:
6         print(i, 'es impar')
7     i = i + 1
```

¿Cuál es su salida? Algune se anima a soplarla? (piensen cuántas veces se ejecuta el programa del bloque indentado)

Ejemplo de if y while

```
1 i = 0
2 while i < 3:
3     if i % 2 == 0:
4         print(i, 'es par')
5     else:
6         print(i, 'es impar')
7     i = i + 1
```

¿Cuál es su salida? Algune se anima a soplarla? (piensen cuántas veces se ejecuta el programa del bloque indentado)

```
1 > 0 es par
2 > 1 es impar
3 > 2 es par
```

Ojo con NO olvidarse la condición de fin del ciclo while!

Ojo con NO olvidarse la condición de fin del ciclo while!

when you forget to write an exit
condition for your while loop



Houston, we have an infinite
loop

Ejemplo de condicional y ciclo un poquito más útil...

```
1 #ingreso las millas
2 mills = [1.1, 33.4, 34.122, 17.588, 3.187, 50.]
3
4 #guardo la longitud de la lista
5 longitud = len(mills)
6
7 #hago la conversion a kilometros
8 km = [0]*longitud #lista de ceros de longitud adecuada
9
10 i = 0 # variable para indicar la posicion en la lista
11 while i < longitud: #condicion
12     km[i] = mills[i]*1.6 #conversion
13     i = i +1 #avance de posicion
```


Ejemplo de condicional y ciclo un poquito más útil...

```
1 #ingreso las millas
2 mills = [1.1, 33.4, 34.122, 17.588, 3.187, 50.]
3
4 #guardo la longitud de la lista
5 longitud = len(mills)
6
7 #hago la conversion a kilometros
8 km = [0]*longitud #lista de ceros de longitud adecuada
9
10 i = 0 # variable para indicar la posicion en la lista
11 while i < longitud: #condicion
12     km[i] = mills[i]*1.6 #conversion
13     i = i +1 #avance de posicion
```

► ¿Qué hace while i < longitud?

Ejemplo de condicional y ciclo un poquito más útil...

```
1 #ingreso las millas
2 mills = [1.1, 33.4, 34.122, 17.588, 3.187, 50.]
3
4 #guardo la longitud de la lista
5 longitud = len(mills)
6
7 #hago la conversion a kilometros
8 km = [0]*longitud #lista de ceros de longitud adecuada
9
10 i = 0 # variable para indicar la posicion en la lista
11 while i < longitud: #condicion
12     km[i] = mills[i]*1.6 #conversion
13     i = i +1 #avance de posicion
```

- ¿Qué hace `while i < longitud`? Nos indica que el bloque de código siguiente se va a ejecutar mientras el resultado de la condición lógica sea verdadera.

Ejemplo de condicional y ciclo un poquito más útil...

```
1 #ingreso las millas
2 mills = [1.1, 33.4, 34.122, 17.588, 3.187, 50.]
3
4 #guardo la longitud de la lista
5 longitud = len(mills)
6
7 #hago la conversion a kilometros
8 km = [0]*longitud #lista de ceros de longitud adecuada
9
10 i = 0 # variable para indicar la posicion en la lista
11 while i < longitud: #condicion
12     km[i] = mills[i]*1.6 #conversion
13     i = i +1 #avance de posicion
```

- ▶ ¿Qué hace `while i < longitud`? Nos indica que el bloque de código siguiente se va a ejecutar mientras el resultado de la condición lógica sea verdadera.
- ▶ Es decir, en este caso, siempre que el índice `i` sea menor a la longitud de la lista (variable `longitud`)

Ejemplo de condicional y ciclo un poquito más útil...

```
1 #ingreso las millas
2 mills = [1.1, 33.4, 34.122, 17.588, 3.187, 50.]
3
4 #guardo la longitud de la lista
5 longitud = len(mills)
6
7 #hago la conversion a kilometros
8 km = [0]*longitud #lista de ceros de longitud adecuada
9
10 i = 0 # variable para indicar la posicion en la lista
11 while i < longitud: #condicion
12     km[i] = mills[i]*1.6 #conversion
13     i = i +1 #avance de posicion
```

- ▶ ¿Qué hace `while i < longitud`? Nos indica que el bloque de código siguiente se va a ejecutar mientras el resultado de la condición lógica sea verdadera.
- ▶ Es decir, en este caso, siempre que el índice `i` sea menor a la longitud de la lista (variable `longitud`)
- ▶ ¿Funciona este programa para la lista de cualquier longitud?

A coedear se a dicho!

A coedear se a dicho!

Tener presente que:

A coedear se a dicho!

Tener presente que:

- ▶ Cualquier *valor que quiera ser reutilizado* (para un cálculo posterior, para una salida, etc.) **debe ser almacenado previamente en una variable**

A coedear se a dicho!

Tener presente que:

- ▶ Cualquier *valor que quiera ser reutilizado* (para un cálculo posterior, para una salida, etc.) **debe ser almacenado previamente en una variable**
- ▶ Siempre que sea posible **no duplicar información** en el código.

A coedear se a dicho!

Tener presente que:

- ▶ Cualquier *valor que quiera ser reutilizado* (para un cálculo posterior, para una salida, etc.) **debe ser almacenado previamente en una variable**
- ▶ Siempre que sea posible **no duplicar información** en el código.
- ▶ Y por último....

No tenerle miedo a los errores!

Se que estás ahí bendito error de sintaxis,
insomnio de mi vida, y voy a encontrarte ;

