Introducción a la programación _

Glosario

- **Algoritmo**: Serie finita de pasos para resolver un problema
- **Bug**: Error en el programa que genera un comportamiento inesperado.
- **Documentación**: Toda la información relacionada con una función o un código específico.
- **Importar**: Añadir código externo.
- **Método/Función**: Bloque de código que realiza una acción específica.
- **Refactorización**: Reestructuración y/o optimización de código.
- **Script**: Es un programa usualmente simple que se puede ejecutar desde la terminal.
- **Variable**: Contenedor de un valor o del resultado de una expresión. Su valor puede cambiar a lo largo de código.

Contenidos del Módulo

Unidad 1:

- Algoritmos
- Controles de Flujo

Unidad 2:

- Ciclos
- Funciones

Unidad 3:

- Listas
- Pandas y Numpy

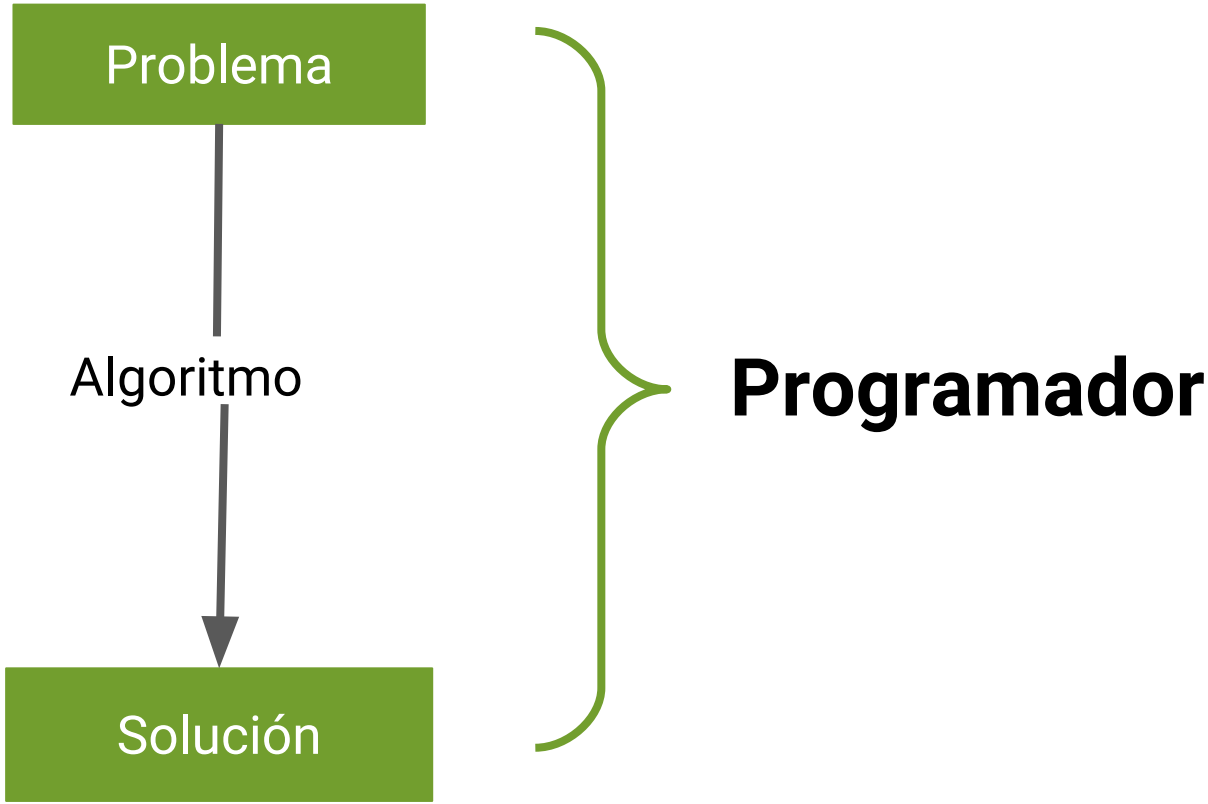
Unidad 4:

- Diccionarios
- API

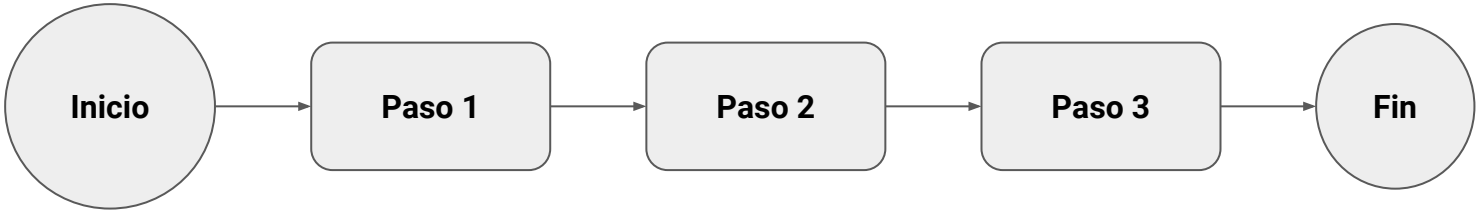
Programas necesarios

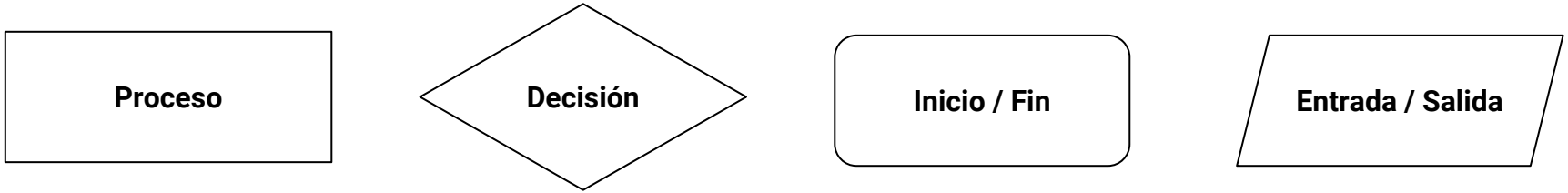
- Anaconda/Python

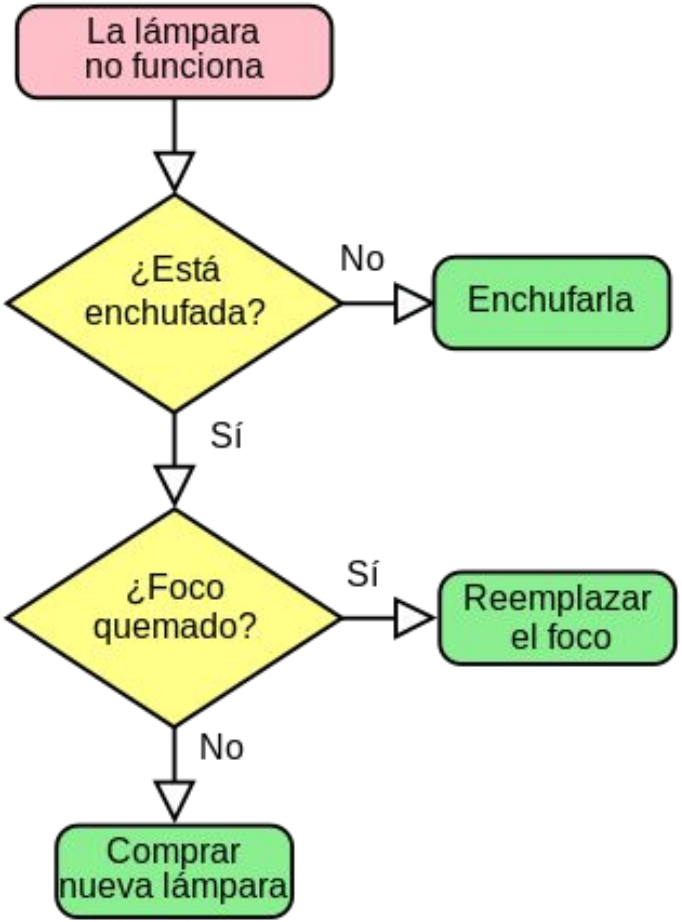
- Editor de texto (Se recomienda Visual Studio Code)


Problema
Algoritmo
Solución
Programador

RECETA DE PANQUEQUES	ALGORITMO
Ingredientes: - 1 taza de harina - 1 taza de leche - 1 huevo Preparación - Mezcle todos los ingredientes hasta tener una mezcla homogénea. - Vierta una porción de la mezcla en una sartén precalentada, esparciendo hasta tener una capa delgada de masa. - Espere 1 minuto y de vuelta la masa. - Espere otro minuto y retire el panqueque con la espátula. - Repite el proceso hasta terminar la mezcla.	1. Agregar 1 taza de harina en un bowl. 2. Agregar 1 taza de leche a la harina. 3. Agregar 1 huevo a los ingredientes previos. 4. Revolver y mezclar los 3 ingredientes. 5. Precalentar el sartén. 6. Agregar parte de la mezcla hasta cubrir el sartén y esparcir una capa delgada. 7. Esperar 1 minuto. 8. Dar vuelta la masa. 9. Esperar otro minuto. 10. Retirar el panqueque. 11. Repetir pasos del 6 al 10 hasta terminar la mezcla.


Inicio
Paso 1
Paso 2
Paso 3
Fin

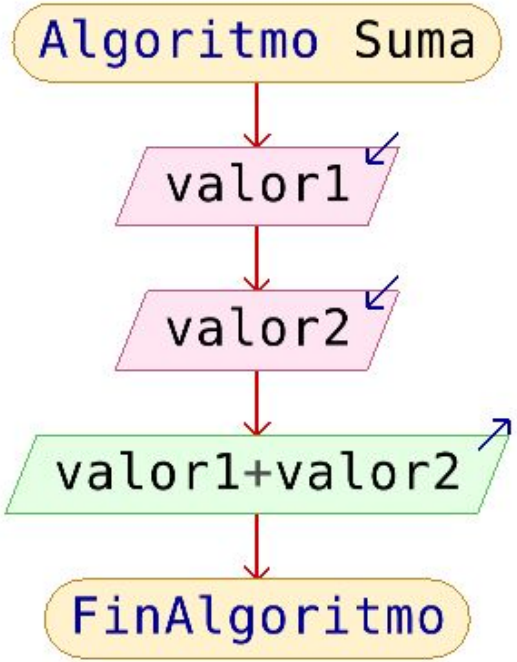

Proceso
Decisión
Inicio / Fin
Entrada / Salida


La lámpara no funciona
¿Está enchufada?
No
Enchufarla
Sí
¿Foco quemado?
Sí
Reemplazar el foco
No
Comprar nueva lámpara

Ejemplo de Pseudocódigo

```
Algoritmo Suma
    Leer valor1
    Leer Valor2
    Mostrar valor1 + valor2
FinAlgoritmo
```

Pseudocódigo a Diagrama de Flujo



Enfrentándose a un problema

1. Analizar el problema
2. Descomponer el problema
3. Determinar los pasos para resolverlo

Python

Características e importancia

- Lenguaje flexible y potente
- Buena elección para comenzar a programar
- Relevante en la industria
- De sintaxis y lectura sencilla, siguiendo algunas normas

Áreas donde se utiliza Python

- Desarrollo Web (Django, Flusk)
- Ciencia de datos y Aprendizaje de máquinas (Scikit-Learn y TensorFlow)

¿Qué crees que hace la siguiente expresión?

```
for i in range(3):
    print(“hip hip hooray!”)
```

hip hip hooray!
hip hip hooray!
hip hip hooray!

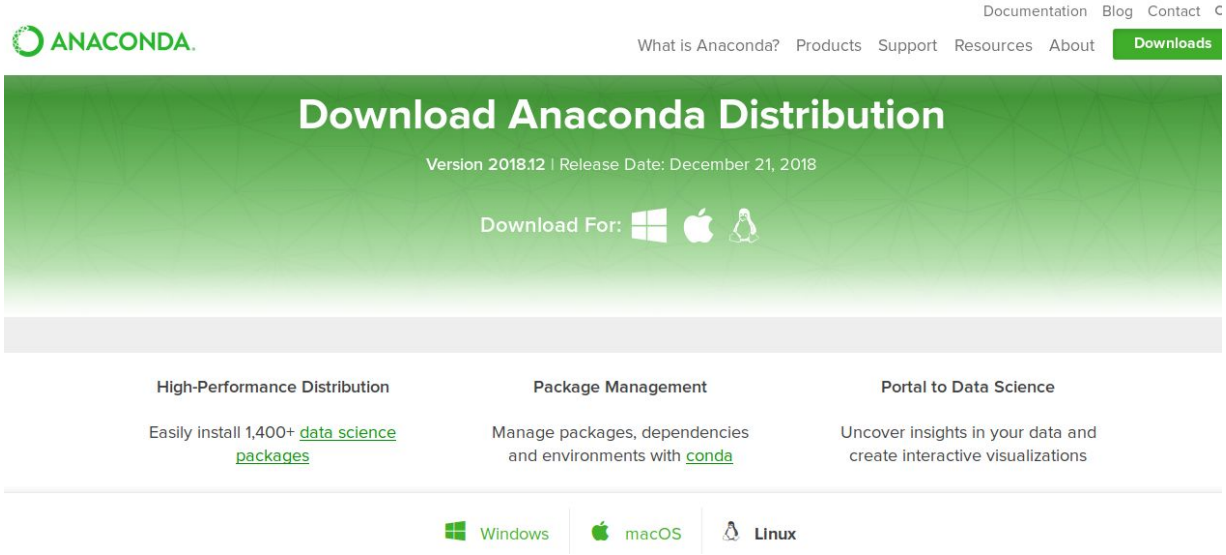
¿Cuál será el resultado de la siguiente expresión?

```
sum([1, 2, 3, 4])
```

10

Descargar Anaconda

Ingresa a https://www.anaconda.com/download/#linux



Anaconda 2018.12 For Linux Installer

Python 3.7 version *

Download

64-Bit (x86) Installer (652.5 MB)
64-Bit (Power8 and Power9) Installer (313.6 MB)
32-Bit Installer (542.7 MB)

Python 2.7 version *

Download

64-Bit (x86) Installer (628.2 MB)
64-Bit (Power8 and Power9) Installer (289.7 MB)
32-Bit Installer (518.6 MB)

Variables

- Tienen un nombre y un valor
- Siguen convenciones
- Pueden ser de distinto tipo, dependiendo del valor asignado
- Cada variable, corresponde a un **objeto** y tienen métodos asociados, dependiendo del tipo de objeto.

```
1 numero = 20
2 soy_una_variable = "Bienvenidos a Introducción a Python"
3
4 # Esto no es correcto
5 usar espacios = "No es posible usar espacios para nombrar una variable"
```

```
  File "<ipython-input-1-22634ce40c83>", line 5
    usar espacios = "No es posible usar espacios para nombrar una variable"
                  ^
SyntaxError: invalid syntax
```

```
1 # Se debe respetar mayúsculas y minúsculas.
2 # Las variables por convención comienzan con minúscula.
3 print(Numero)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-dd55f125c425> in <module>()
      1 # Se debe respetar mayúsculas y minúsculas.
      2 # Las variables por convención comienzan con minúscula.
----> 3 print(Numero)

NameError: name 'Numero' is not defined
```

String

```python
nombre = 'Carlos'
apellido = 'Santana'
```

```python
# Concatenación
print("Mi nombre es " + nombre + " " + apellido)
```

```
Mi nombre es Carlos Santana
```

```python
# Interpolación
print("Mi nombre es {} {}".format(nombre, apellido))
```

```
Mi nombre es Carlos Santana
```

```python
print(nombre.count("a"))
print(len(apellido))
print(apellido.upper())
```

```
1
7
SANTANA
```

Integers

```
1 a = 10
2 b = 2
3 print(a * b)
```

```
20
```

```
1 print(a / b + 15)
```

```
20.0
```

```
1 cadena = "2"
2 print(cadena + 2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-b3bb090f63c1> in <module>()
      1 cadena = "2"
----> 2 print(cadena + 2)

TypeError: must be str, not int
```

Integers y Scripts

argumentos.py

```python
import sys

b = sys.argv[1]

print(b + 2)
```

```
giani@giani-dev:~$ python argumentos.py 20
Traceback (most recent call last):
  File "argumentos.py", line 5, in <module>
    print(b + 2)
TypeError: must be str, not int
```

argumentos.py

```python
import sys

b = int(sys.argv[1])

print(b + 2)
```

```
giani@giani-dev:~$ python argumentos.py 20
22
```

Tipos de objeto

- **Integer**: Corresponde a un número entero.
- **String:** Corresponde a un carácter o una cadena de caracteres.
- **Float**: Corresponde a un número decimal.
- **Time**: Corresponde a una fecha y hora.
- **Boolean**: Corresponde a True o False. Son el resultado de una evaluación.
- **None**: corresponde a la ausencia de un valor.

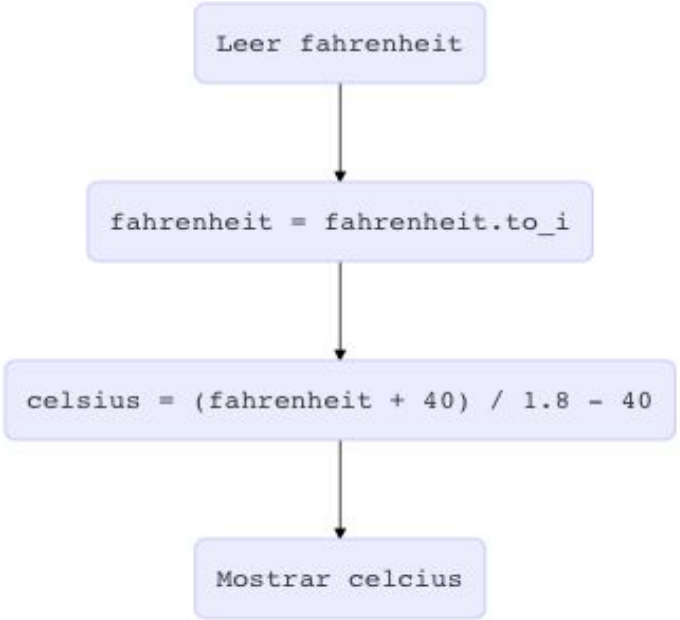
Operadores aritméticos

OPERADOR	NOMBRE	EJEMPLO	RESULTADO
+	Suma	2 + 3	5
-	Resta	2 - 3	-1
*	Multiplicación	3 * 4	12
/	División	12 / 4	3
**	Potencia	2 ** 4	16

Orden de las operaciones

OPERADOR	NOMBRE
**	Exponenciación (potencia)
* , / , %	Multiplicación, división y módulo
+ , -	Suma y resta

Ejercicio Fahrenheit - Algoritmo



Ejercicio Fahrenheit - Código

```
fahrenheit = int(input())
celsius = (fahrenheit + 40) / 1.8 - 40
print(“la temperatura es de {} celsius”.format(celsius))
```

Operadores de comparación

OPERADOR	NOMBRE	EJEMPLO	RESULTADO
==	Igual a	2 == 2	true
!=	Distinto a	2 != 2	false
>	Mayor a	3 > 4	false
>=	Mayor o igual a	3 >= 3	true
<	Menor a	4 < 3	false
<=	Menor o igual a	3 <= 4	true

Manejo de Flujo

Paso 1: Crear if para evaluar si el primer valor es mayor

```
valor1 = int(input(“Ingrese valor 1”))
valor2 = int(input(“Ingrese valor 2”))
if valor1 >= valor2:
    print(“valor1 {} es mayor”.format(valor1))
```

Paso 2: Agregar else para manejar el caso contrario

```
valor1 = int(input(“Ingrese valor 1”))
valor2 = int(input(“Ingrese valor 2”))
if valor1 >= valor2:
    print(“valor1 {} es mayor”.format(valor1))
else:
    print(“valor2 {} es mayor”.format(valor2))
```

Paso 3: Manejar cuando ambos números son iguales

```
valor1 = int(input(“Ingrese valor 1”))
valor2 = int(input(“Ingrese valor 2”))
if valor1 > valor2:
    print(“valor1 {} es mayor”.format(valor1))
else:
    if valor1 == valor2:
        print(“Ambos valores son iguales”)
    else:
        print(“valor2 {} es mayor”.format(valor2))
```

Paso 4: Reescribir el código usando elif

```
valor1 = int(input(“Ingrese valor 1”))
valor2 = int(input(“Ingrese valor 2”))
if valor1 > valor2:
    print(“valor1 {} es mayor”.format(valor1))
elif valor1 == valor2:
    print(“Ambos valores son iguales”)
else:
    print(“valor2 {} es mayor”.format(valor2))
```

Operadores lógicos

OPERADOR	NOMBRE	EJEMPLO	RESULTADO
&	y (and)	False & True	Devuelve true si ambos operandos son true. En este ejemplo se devuelve false.
|	o (or)	False | True	Devuelve true si al menos uno de los operandos es true. En este ejemplo devuelve true.
!=	distinto de (not)	True != False	Devuelve lo opuesto al resultado de la evaluación. En este ejemplo devuelve true.

Eliminando if anidados

```
edad = 30
zurdo = True
if edad >= 18:
    if zurdo is True:
        print(“Es zurdo y mayor de edad”)
```

```
edad = 30
zurdo = True
if edad >= 18 and zurdo is True:
    print(“Es zurdo y mayor de edad”)
```

Ejercicio de integración
Paso a Paso

Paso 1: Identificar entradas

mayor_de_tres.py

```
1  import sys
2
3  # Entradas
4  primer_numero = int(sys.argv[1])
5  segundo_numero = int(sys.argv[2])
6  tercer_numero = int(sys.argv[3])
```

Paso 2: Identificar salidas

mayor_de_tres.py

```
1  import sys
2
3  # Entradas
4  primer_numero = int(sys.argv[1])
5  segundo_numero = int(sys.argv[2])
6  tercer_numero = int(sys.argv[3])
7
8  # Salidas
9  print("El primer número es el mayor")
10 print("El segundo número es el mayor")
11 print("El tercer número es el mayor")
```

Paso 3: Manejar 3 salidas con if, elif y else

mayor_de_tres.py

```
1  import sys
2
3  # Entradas
4  primer_numero = int(sys.argv[1])
5  segundo_numero = int(sys.argv[2])
6  tercer_numero = int(sys.argv[3])
7
8  # Salidas
9
10 if:
11     print("El primer número es el mayor")
12 elif:
13     print("El segundo número es el mayor")
14 else:
15     print("El tercer número es el mayor")
```

Paso 4: Traducir condición del if a código

mayor_de_tres.py

```
1  import sys
2
3  # Entradas
4  primer_numero = int(sys.argv[1])
5  segundo_numero = int(sys.argv[2])
6  tercer_numero = int(sys.argv[3])
7
8  # Salidas
9
10 if primer_numero > segundo_numero and primer_numero > tercer_numero:
11     print("El primer número es el mayor")
12 elif:
13     print("El segundo número es el mayor")
14 else:
15     print("El tercer número es el mayor")
```

Paso 5: Traducir condición del elif a código

mayor_de_tres.py

```
1  import sys
2
3  # Entradas
4  primer_numero = int(sys.argv[1])
5  segundo_numero = int(sys.argv[2])
6  tercer_numero = int(sys.argv[3])
7
8  # Salidas
9
10 if primer_numero > segundo_numero and primer_numero > tercer_numero:
11     print("El primer número es el mayor")
12 elif segundo_numero > tercer_número:
13     print("El segundo número es el mayor")
14 else:
15     print("El tercer número es el mayor")
```

Ejercicio de refactorización
Paso a Paso

Paso 1: Se muestra el código sin refactorizar

refactorizar.py

```
1  mayor_de_edad = True
2  zurdo = False
3
4  if mayor_de_edad is True:
5      if zurdo is True:
6          print("Mayor de edad y zurdo!")
7      else:
8          print("Mayor de edad pero no zurdo!")
9  else:
10     if zurdo is True:
11         print("Menor de edad y zurdo")
12     else:
13         print("Menor de edad pero no zurdo!")
```

Paso 2: Eliminar if anidados usando el operador and

refactorizar.py

```
1  mayor_de_edad = True
2  zurdo = False
3
4  if mayor_de_edad is True and zurdo is True:
5      print("Mayor de edad y zurdo")
6  elif mayor_de_edad is True and zurdo is False:
7      print("Mayor de edad pero no zurdo")
8  elif mayor_de_edad is False and zurdo is True:
9      print("Menor de edad y zurdo")
10 else:
11     print("Menor de edad y no zurdo")
```

Paso 3: Refactorizar comparaciones condicionales

refactorizar.py

```
1  mayor_de_edad = True
2  zurdo = False
3
4  if mayor_de_edad and zurdo:
5      print("Mayor de edad y zurdo")
6  elif mayor_de_edad and zurdo is False:
7      print("Mayor de edad pero no zurdo")
8  elif mayor_de_edad is False and zurdo:
9      print("Menor de edad y zurdo")
10 else:
11     print("Menor de edad y no zurdo")
```