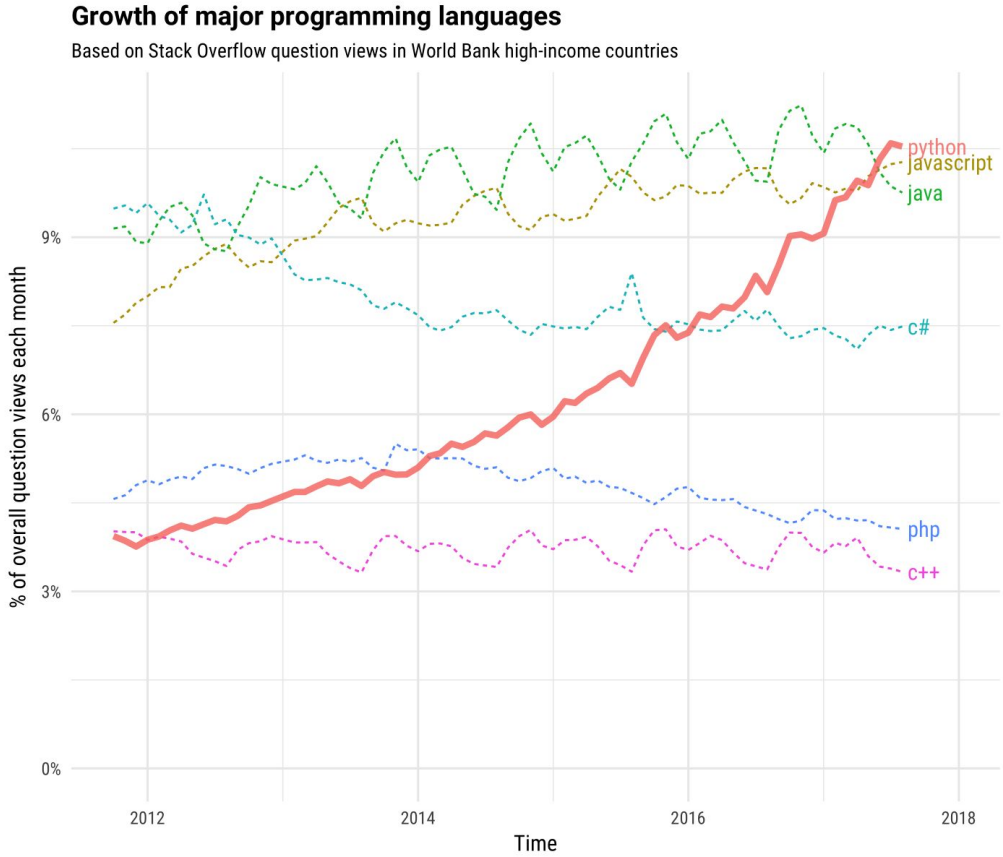
IA PUCP - Diplomado de Desarrollo de Aplicaciones de Inteligencia Artificial
Python para Ciencia de Datos

Introducción (rápida) a Python

Contenido de la sesión

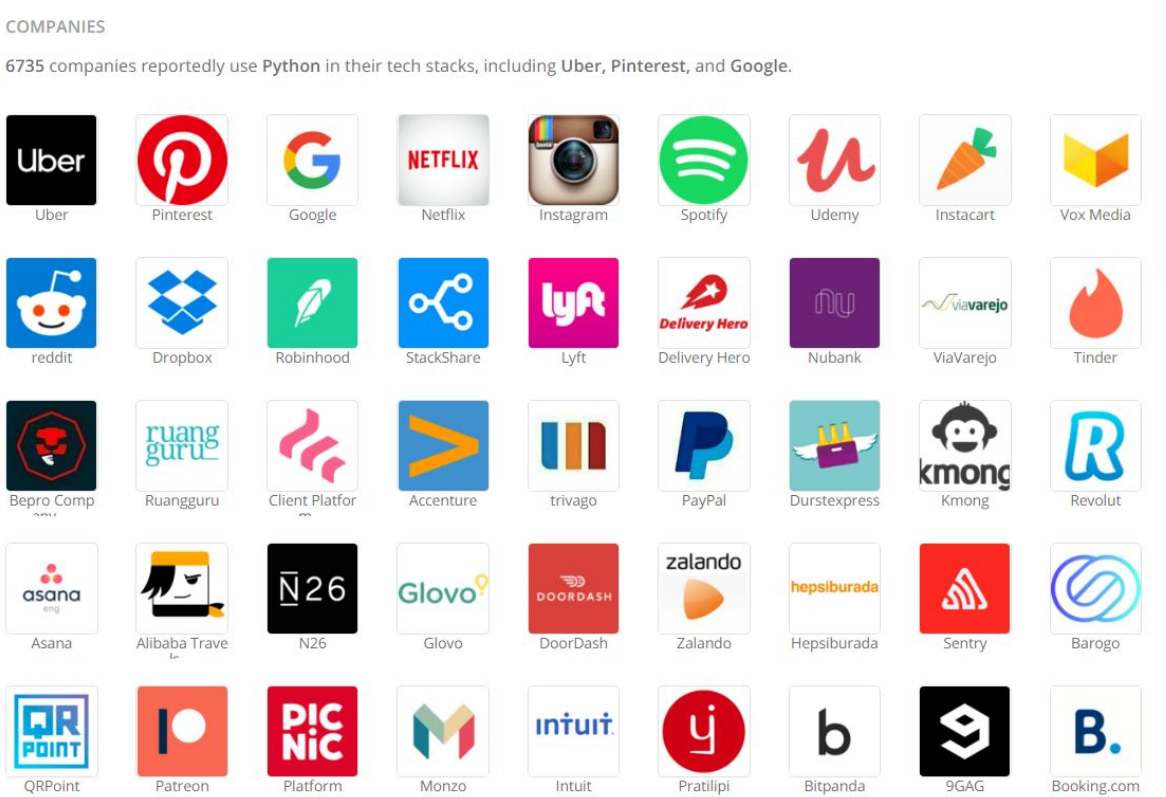
- ¿Por qué Python?
- Ecosistema Python para la Ciencia de Datos
- Programación en Python

Python: un lenguaje de programación en crecimiento



https://news.codecademy.com/why-learn-python/

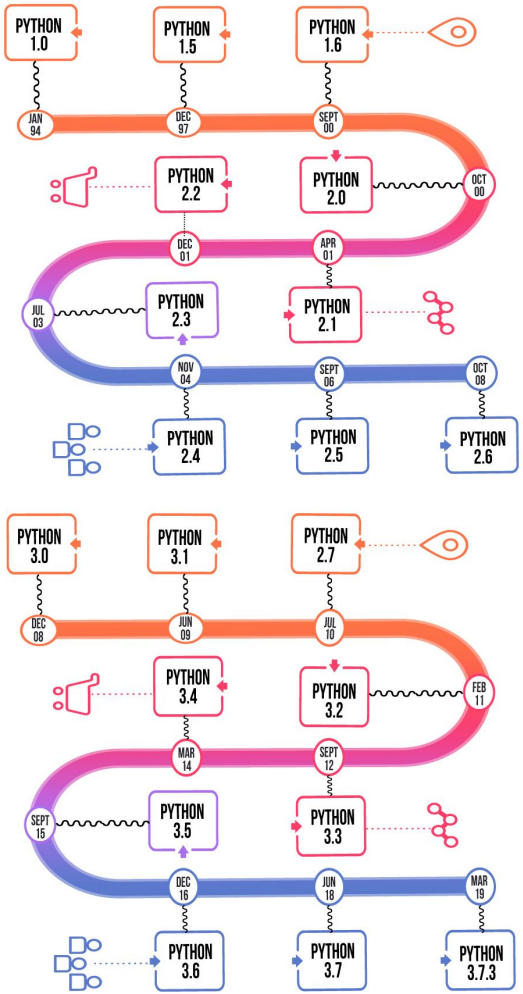
¿Quiénes usan Python?



https://stacksh
are.io/python

Desarrollo de Python

- Diseñado por Guido van Rossum en 1991
- El soporte para Python 2.x finalizó el 1 de enero del 2020
- La versión actual con soporte es la rama 3.x y comenzó en Diciembre del 2008.



https://www.geeksforgeeks.org/history-of-python/

Comencemos con Python

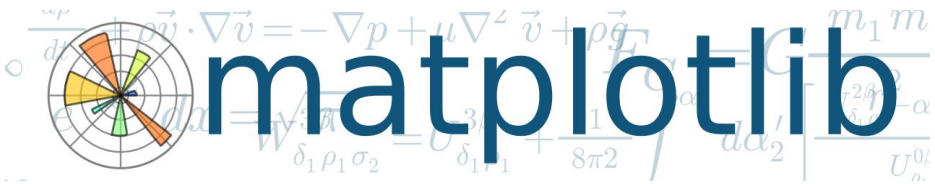
Autograder.io https://grader.labs.org.pe/

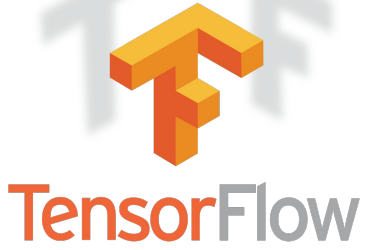
01 - ¡Hola Mundo!

Escriba un programa en el lenguaje de Programación Python 3.x que imprima la frase “Hello World” sin las comillas. Recuerde que deben coincidir las mayúsculas y minúsculas.

El nombre del programa debe ser: `hello_world.py`

Ecosistema Python para Data Science





https://en.wikipedia.org/wiki/NumPy
https://desenfasados.com/scipy-modulo-para-machine-learning/
https://pandas.pydata.org/docs/whatsnew/v1.0.0.html
https://es.wikipedia.org/wiki/Scikit-learn
https://cloud.google.com/blog/products/ai-machine-learning/introducing-pytorch-across-google-cloud
http://sitiobigdata.com/2019/02/09/google-colab-regresion-lineal-pyspark/
http://www.pythondiario.com/2017/12/visualizacion-de-datos-con-python-y.html

Características de Python

- Legible
- Todos son objetos
- Tipos dinámicos
- Iteraciones
- Espacios de nombre
- Gestión automática de la memoria
- Lenguaje interpretado

Objetos

- los programas manipulan **objetos de datos**
- los objetos tienen un **tipo (type)** que define la clase de cosas que los programas pueden hacerles
  - Ana es humana, por lo tanto puede caminar, hablar español, etc.
  - Fido es un perro, por lo tanto puede caminar, “guau, guau”, etc.
- Los objetos son
  - escalares (no pueden ser subdivididos)
  - no escalares (tienen un estructura interna que puede ser accesada)

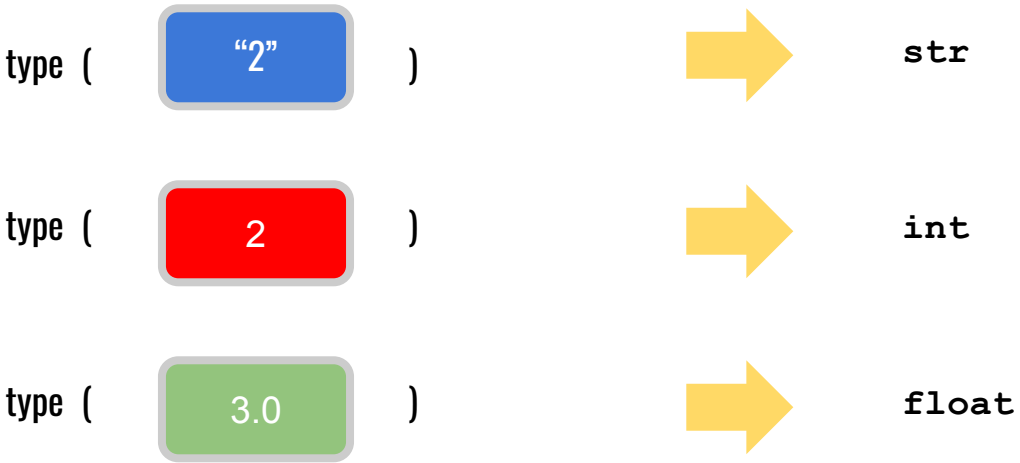
Objetos escalares

- **`int`** - representa enteros, por ej. 5
- **`float`** - representa números reales, por ej. 3.27
- **`bool`** - representa los valores booleanos **`True`** (verdadero) y **`False`** (falso)
- **`NoneType`** - especial y tiene un único valor: `None`

Podemos usar `type()` para saber el tipo de dato de un objeto

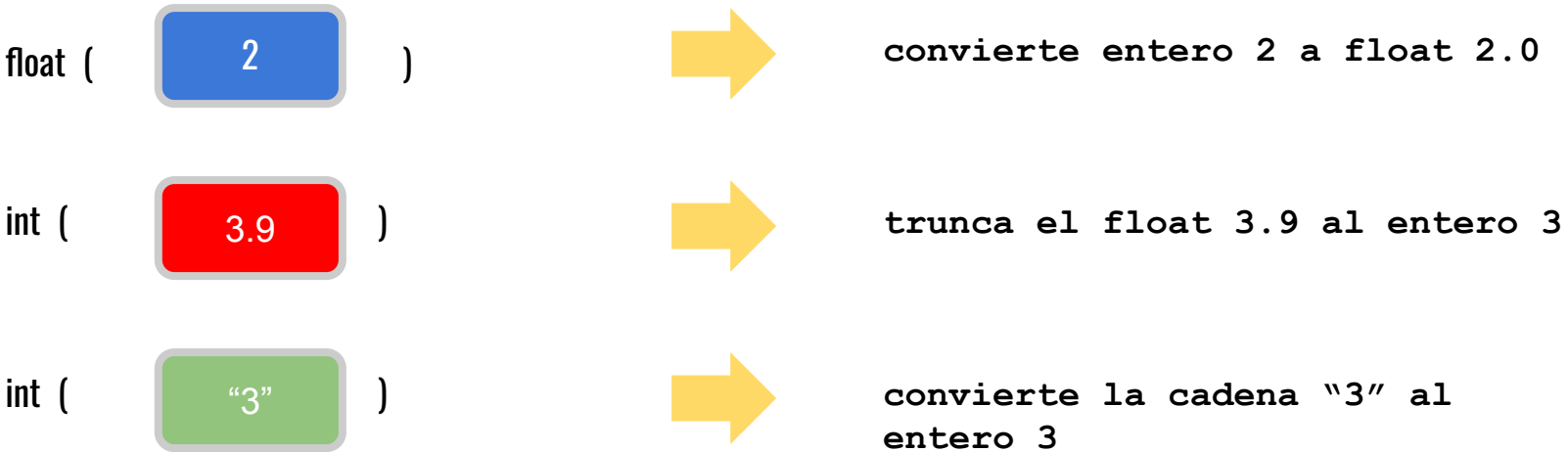
Averiguando el “tipo” de objetos

- Usando el operador “type”



Conversión de Tipos

- Explícita
  - Por operación de ***cast***



Expresiones

- combina objetos y operadores para formar expresiones
- una expresión tiene un valor, que es de un tipo de objeto en particular
- la sintaxis de una expresión es la siguiente

  `<objeto> <operador> <objeto>`

Operadores en `int` y `float`

- `i + j` → la suma
- `i - j` → la resta
- `i * j` → el producto

Si ambos operandos son enteros, el resultado será entero

Si alguno de los operadores (o ambos) es float el resultado será float

- `i / j` → la división

el resultado es float

- `i % j` → el residuo de dividir `i` entre `j`
- `i ** j` → `i` elevado a la `j`

Operaciones simples

- los paréntesis indican a Python hacer estas operaciones primero
- precedencia de operadores sin paréntesis
  - `**`
  - `*`
  - `/`
  - `+` y `-` son ejecutadas de izquierda a derecha, conforme aparecen en la expresión

Uniendo valores y variables

- el signo de igual (=) indica la asignación de un valor a un nombre de variable

variable valor

```
pi = 3.14159
```

```
pi_aprox = 22/7
```

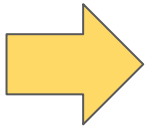
el valor es almacenado en la memoria de la computadora

la asignación une el valor y la variable

recuperar el valor al llamar a la variable al tipear `pi`

¿Cómo funciona la asignación?

```
a = 5
```



3 operaciones!

¿Cómo funciona la asignación?

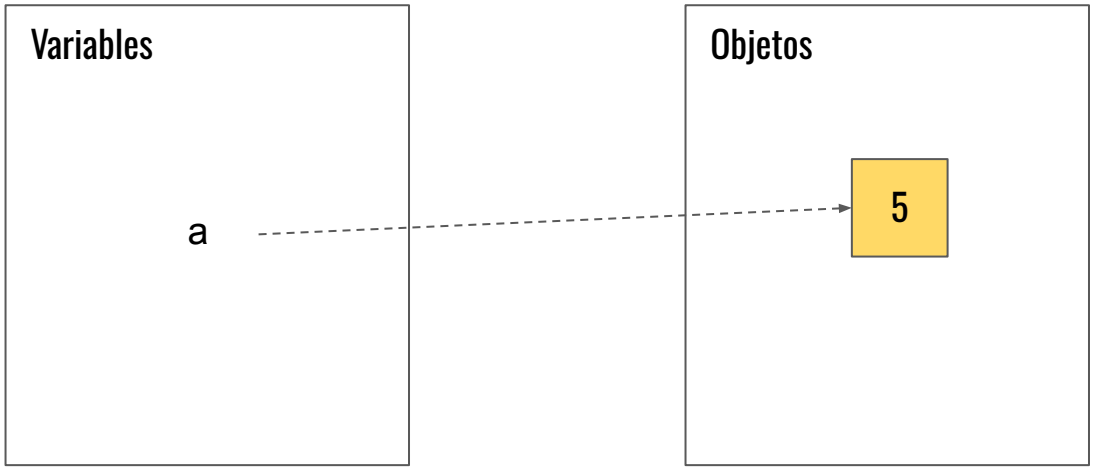
```
a = 5
```

¿Cómo funciona la asignación?

```
a = 5
```

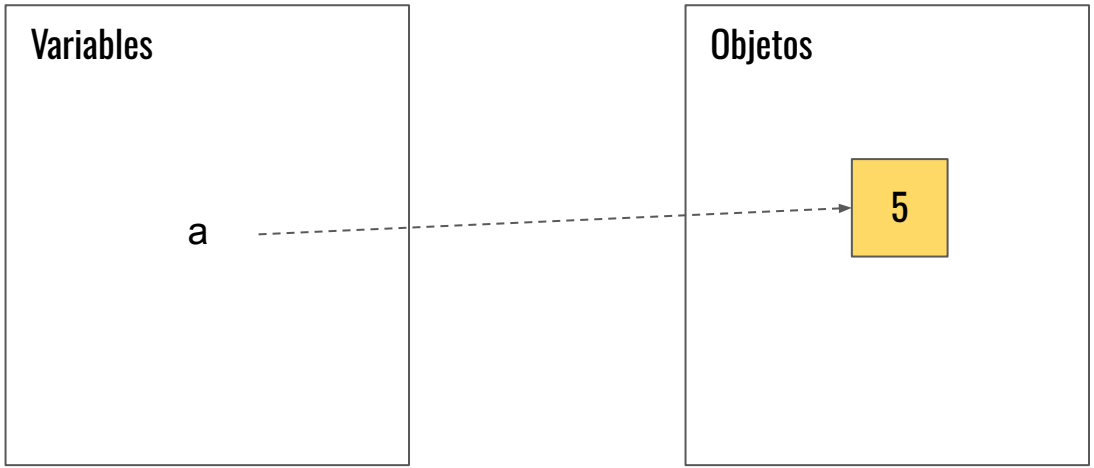
¿Cómo funciona la asignación?

```
a = 5
```



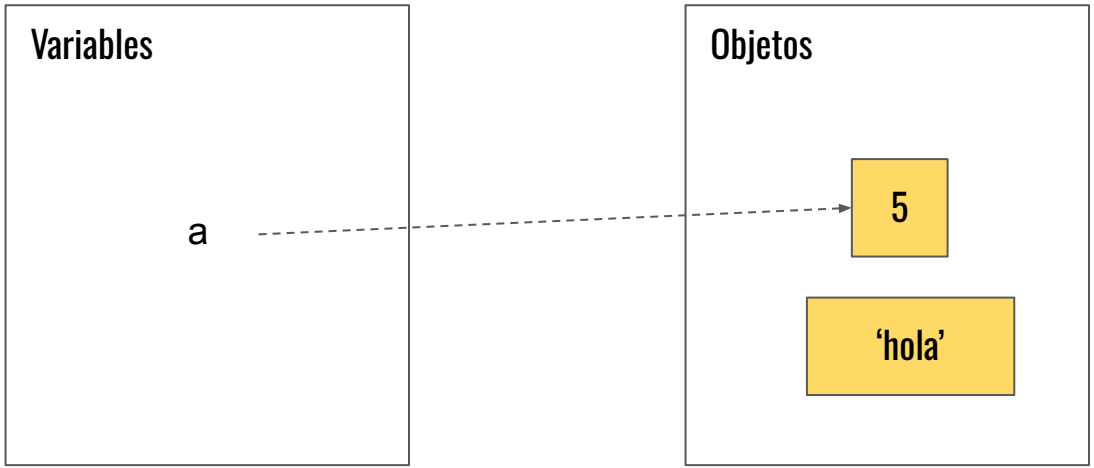
¿Cómo funciona la asignación?

```
a = 5
a = 'hola'
```



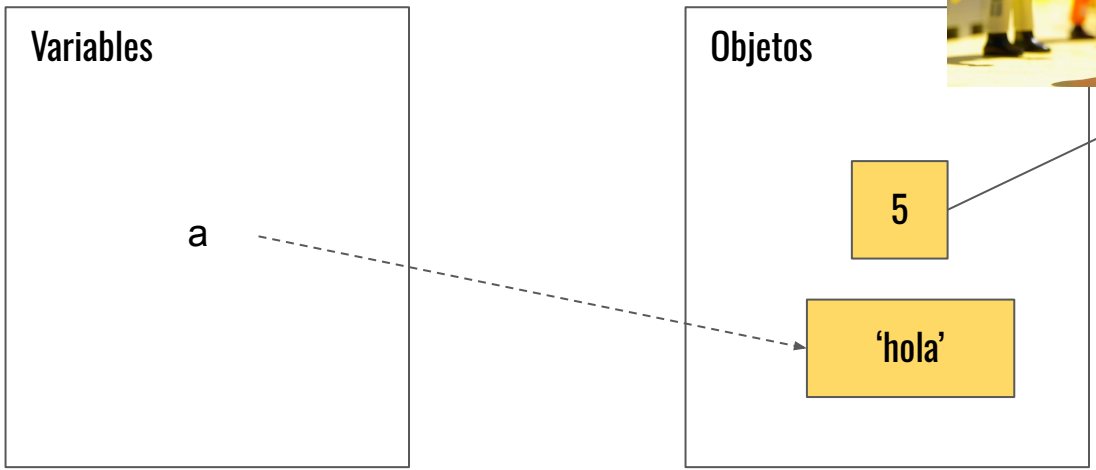
¿Cómo funciona la asignación?

```
a = 5
a = 'hola'
```



¿Cómo funciona la asignación?

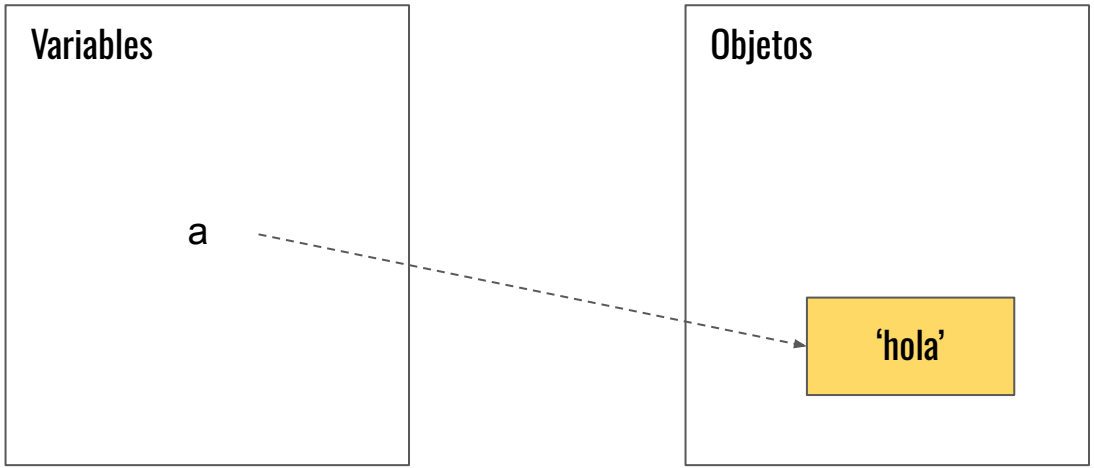
```
a = 5
a = 'hola'
```



¿Cómo funciona la asignación?

```
a = 5
a = 'hola'
```

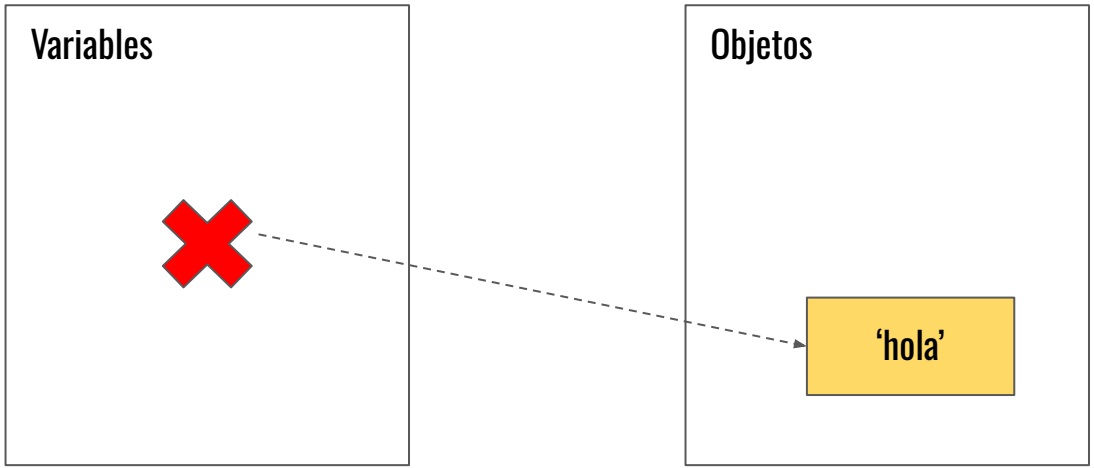
`del a`



¿Cómo funciona la asignación?

```
a = 5
a = 'hola'
```

`del a`



¿Cómo funciona la asignación?

```
a = 5
a = 'hola'
```

```
del a
```

¿Cómo funciona la asignación?

```
a = 5
a = 'hola'
```

`del a`

¿Cómo funciona la asignación?

```
a = 5
a = 'hola'
```

```
del a
```

Salida: `print`

- `print` se usar para mostrar salidas del programa en la consola

```
print(“hola mundo”)
```

Salida: `print`

- `print` se usar para mostrar salidas del programa en la consola

```
print(“hola”, “mundo”, “!”)
```

Funciona con múltiples variables

Entrada: `input`

- `input` permite realizar lecturas – devuelve un `str`

```
nombre = input("Nombre:")
```

Entrada: `input`

- `input` permite realizar lecturas – devuelve un `str`

```
nombre = input(“Nombre:”)
```

Este contenido se mostrará en la salida, equivalente al uso de un `print`

Pregunta:

¿Qué tipo de dato tendría la expresión a continuación si **el usuario ingresa un número entero como 3**?

```
a = input()
```

```
3
```

A) int

B) float

C) str

D) Error

Pregunta:

¿Qué tipo de dato tendría la expresión a continuación si **el usuario ingresa un número entero como 3**?

```
a = input()
```

```
3
```

La función `input` siempre devuelve una cadena de texto (`str`)

A) int

B) float

C) str

D) Error

Pregunta:

¿Qué tipo de dato tendría la expresión a continuación si **el usuario ingresa un número entero como 5**?

```
a = int(input())
```

```
5
```

A) int

B) float

C) str

D) Error

Pregunta:

¿Qué tipo de dato tendría la expresión a continuación si **el usuario ingresa un número entero como 5**?

```
a = int(input())
```

```
5
```

La función `input` devuelve una cadena de texto (`str`) que convertimos explícitamente a un entero (`int`)

A) int

B) float

C) str

D) Error

Pregunta:

¿Qué tipo de dato tendría la expresión a continuación si **el usuario ingresa una cadena como Hola?**

```
a = int(input())
```

Hola

A) int

B) float

C) str

D) Error

Pregunta:

¿Qué tipo de dato tendría la expresión a continuación si **el usuario ingresa una cadena como Hola?**

```
a = int(input())
```

Hola

Convertir una cadena de texto (`str`) que no tenga un valor numérico a un entero (`int`) generará un error

A) int

B) float

C) str

D) Error

Recordemos algunos operadores aritméticos

¿Cuál es el resultado?

```
a = 5
b = a % 2
print(b)
```

A) 0

B) 1

C) 2

D) 3

¿Cuál es el resultado?

```
a = 5
b = a % 2
print(b)
```

El operador módulo o residuo devuelve el valor del residuo de la división entera

A) 0

B) 1

C) 2

D) 3

¿Cuál es el resultado?

```
a = 6
b = a % 2
print(b)
```

A) 0

B) 1

C) 2

D) 3

¿Cuál es el resultado?

```
a = 6
b = a % 2
print(b)
```

A) 0

B) 1

C) 2

D) 3

Pregunta:

```
b = a % 2
print(b)
```

Si **a** es un entero positivo, ¿cuales son los valores que podría tener **b**?

A) {0}

B) {1}

C) {0, 1}

D) Enteros positivos

Pregunta:

```
b = a % 2
print(b)
```

Si **a** es un entero positivo, ¿cuales son los valores que podría tener **b**?

A) {0}

B) {1}

C) {0, 1}

D) Enteros positivos

Operadores de comparación en: `int, float y string`

Por lo general, vamos a querer hacer comparaciones

- Considere que **`i y j`** son nombres de variables
- Expresiones de comparación retornan **`bool`**
  - Mayor: **`i > j`**
  - Mayor o Igual: **`i >= j`**
  - Menor: **`i < j`**
  - Menor o Igual: **`i <= j`**
  - Igual: **`i == j`**
  - Diferente: **`i != j`**

Pregunta:

¿Qué tipo de dato tendría la expresión a continuación si **a** es un entero positivo?

```
(a % 2)==0
```

A) int

B) float

C) str

D) boolean

Pregunta:

¿Qué tipo de dato tendría la expresión a continuación si **a** es un entero positivo?

```
(a % 2)==0
```

A) int

B) float

C) str

D) boolean

Pregunta:

¿Qué valor tendrá la siguiente expresión si a es un número **par**?

```
(a % 2)==0
```

A) True

B) False

Pregunta:

¿Qué valor tendrá la siguiente expresión si a es un número **par**?

```
(a % 2)==0
```

A) True

B) False

Pregunta:

¿Qué valor tendrá la siguiente expresión si a es un número **impar**?

```
(a % 2)==0
```

A) True

B) False

Pregunta:

¿Qué valor tendrá la siguiente expresión si a es un número **impar**?

```
(a % 2)==0
```

A) True

B) False

Asignación + Operaciones aritméticas

¿Cuál es el resultado?

```
a = 5
a = a + 1
print(a)
```

A) 5

B) 4

C) 6

D) Error

¿Cuál es el resultado?

```
a = 5
a = a + 1
print(a)
```

`a = a + 1` equivale a incrementar el valor de `a` en `1`

A) 5

B) 4

C) 6

D) Error

¿Cuál es el resultado?

```
a = 5
a += 1
print(a)
```

A) 5

B) 4

C) 6

D) Error

¿Cuál es el resultado?

```
a = 5
a += 1
print(a)
```

`a = a + 1` se puede escribir “resumidamente” como `a+=1`

A) 5

B) 4

C) 6

D) Error

Operadores de Asignación & aritméticos

- `a += b`
- `a -= b`
- `a *= b`
- `a /= b`
- `a %= b`
- `a **= b`
- `a//=b`

Operadores de Asignación & aritméticos

- `a += b` → **`a = a + b`**
- `a -= b` → **`a = a - b`**
- `a *= b` → **`a = a*b`**
- `a /= b` → **`a = a / b`**
- `a %= b` → **`a = a % b`**
- `a **= b` → **`a = a**b`**
- `a//=b` → **`a = a//b`**

¿Cuál es el resultado?

```
a = 5
a += 1
a **= 2
a = a + 4
a //= 10
a -= 1
print(a)
```

A) 0

B) 1

C) 2

D) 3

¿Cuál es el resultado?

```
a = 5
a += 1
a **= 2
a = a + 4
a //= 10
a -= 1
print(a)
```

A) 0

B) 1

C) 2

D) 3

Pregunta:

¿Qué tipo de dato tendría la expresión a continuación?

```
a = 3 / 2
```

A) int

B) float

C) str

D) boolean

Pregunta:

¿Qué tipo de dato tendría la expresión a continuación?

```
a = 3 / 2
```

A) int

B) float

C) str

D) boolean

Pregunta:

¿Qué tipo de dato tendría la expresión a continuación?

```
a = 3 // 2
```

A) int

B) float

C) str

D) boolean

Pregunta:

¿Qué tipo de dato tendría la expresión a continuación?

```
a = 3 // 2
```

A) int

B) float

C) str

D) boolean

Cadenas

- Letras, caracteres especiales, espacios y dígitos
- se delimita con comillas simples o dobles
- se pueden concatenar
  - `nombre = “pablo”`
  - `saludos = “hola” + nombre`
- algunas operaciones
  - `“hola”*4`

¿Cuál es el resultado?

```
a = 5
a += 1
a **= 2
a = a + 4
a //= 10
a -= 1
print(a)
```

A) 0

B) 1

C) 2

D) 3

¿Cuál es el resultado?

```
a = 5
a += 1
a **= 2
a = a + 4
a //= 10
a -= 1
print(a)
```

A) 0

B) 1

C) 2

D) 3

Cadenas

- Letras, caracteres especiales, espacios y dígitos
- se delimita con comillas simples o dobles
- se pueden concatenar
  - `nombre = “pablo”`
  - `saludos = “hola” + nombre`
- algunas operaciones
  - `“hola”*4`

Entrada / Salida: `print` e `input`

- `print` se usar para mostrar salidas del programa en la consola
- `input` permite realizar lecturas – devuelve un `str`

```
nombre = input("Nombre:")
print("hola mi nombre es", nombre)
```

Operadores de comparación en: `int, float y string`

- `i y j` son nombres de variables
- Expresiones de comparación retornan `bool`
  - `i > j`
  - `i >= j`
  - `i < j`
  - `i <= j`
  - `i == j`
  - `i != j`

Operadores lógicos en : `bools`

- `a` y `b` son nombres de variables con valores booleanos
- `not a` → `True` si `a` es `False`
  `False` si `a` es `True`
- `a and b` → `True` si ambos son `True`
- `a or b` → `True` si alguno es `True`

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	?	?
True	False	?	?
False	True	?	?
False	False	?	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	**?**	?
True	False	?	?
False	True	?	?
False	False	?	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	?
True	False	?	?
False	True	?	?
False	False	?	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	?
True	False	?	?
False	True	?	?
False	False	?	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	True
True	False	?	?
False	True	?	?
False	False	?	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	True
True	False	**?**	?
False	True	?	?
False	False	?	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	True
True	False	False	?
False	True	?	?
False	False	?	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	True
True	False	False	**?**
False	True	?	?
False	False	?	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	?	?
False	False	?	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	**?**	?
False	False	?	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	?
False	False	?	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	**?**
False	False	?	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	?	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	**?**	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	?

Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	?

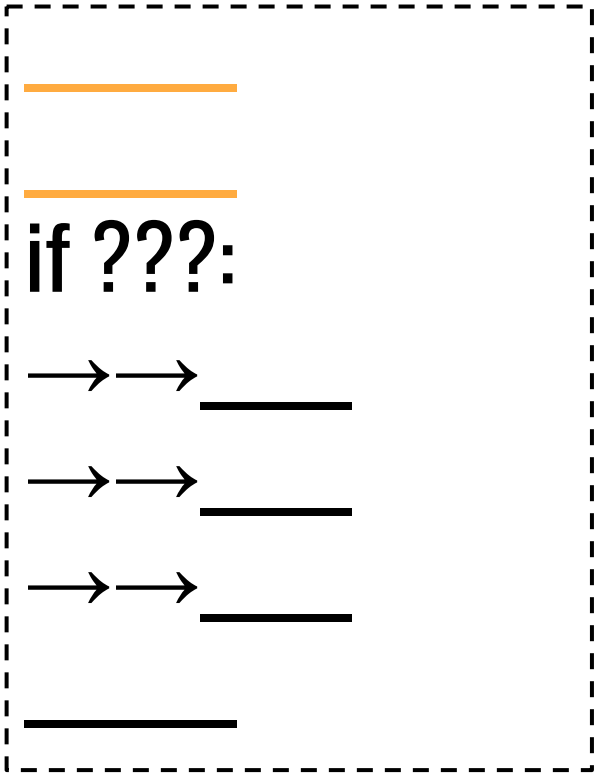
Operadores lógicos en : `bools`

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Instrucción `if` en Python

```
_____
_____
if ???:
    ____
    ____
    ____

_____
```

Instrucción `if` en Python

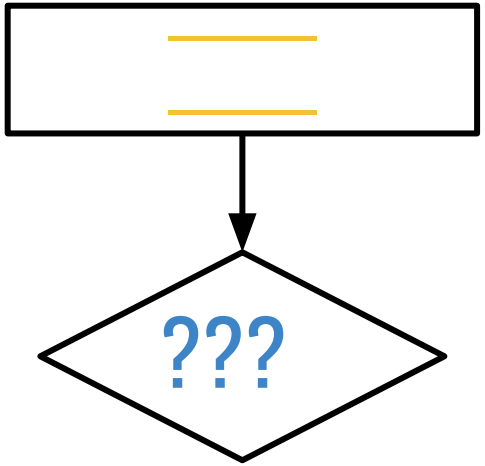


Este código se ejecuta de manera secuencial

Instrucción `if` en Python

```
_____
_____
if ???:
    ____
    ____
    ____
_____
```

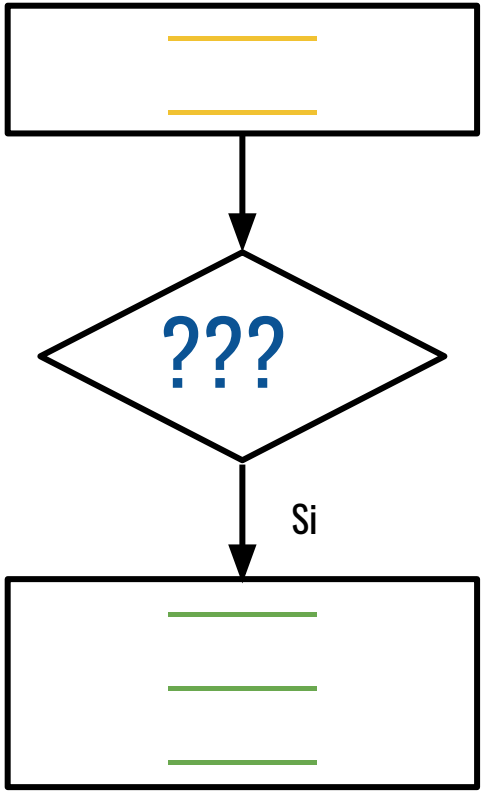
La condición o pregunta, será una expresión con valor `True` o `False`



Instrucción `if` en Python

```
if ???:
    →→
    →→
    →→
```

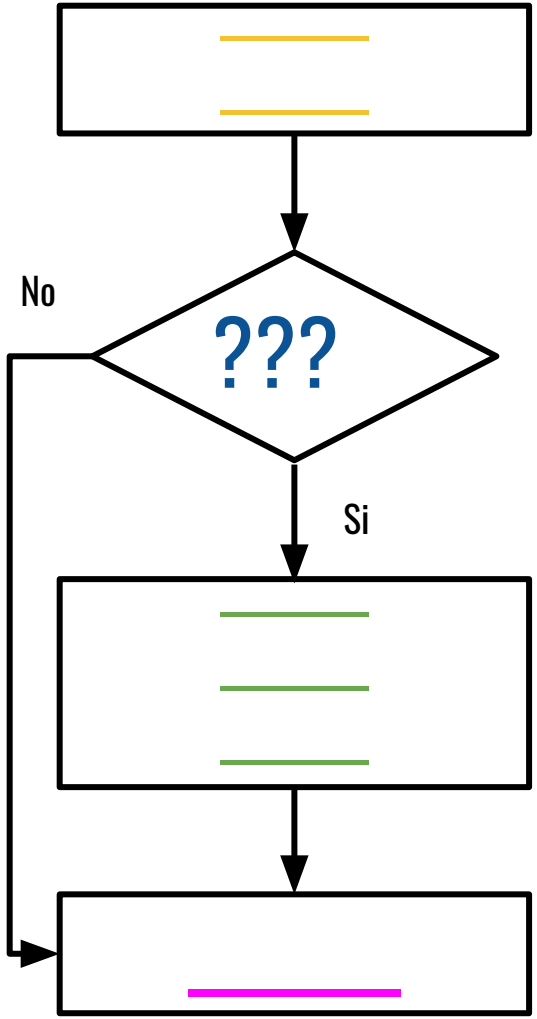
Este bloque de código se ejecuta solo si la condición es `True`



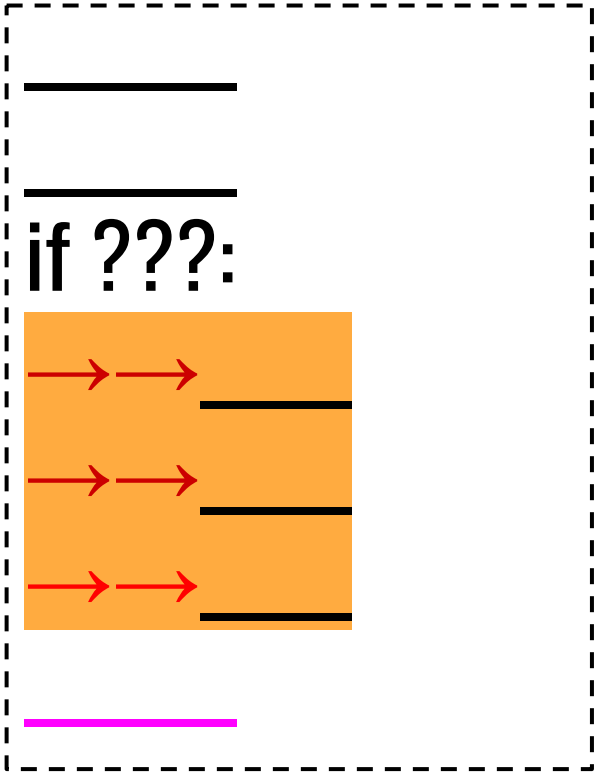
Instrucción `if` en Python

```
if ???:
    →→
    →→
    →→
```

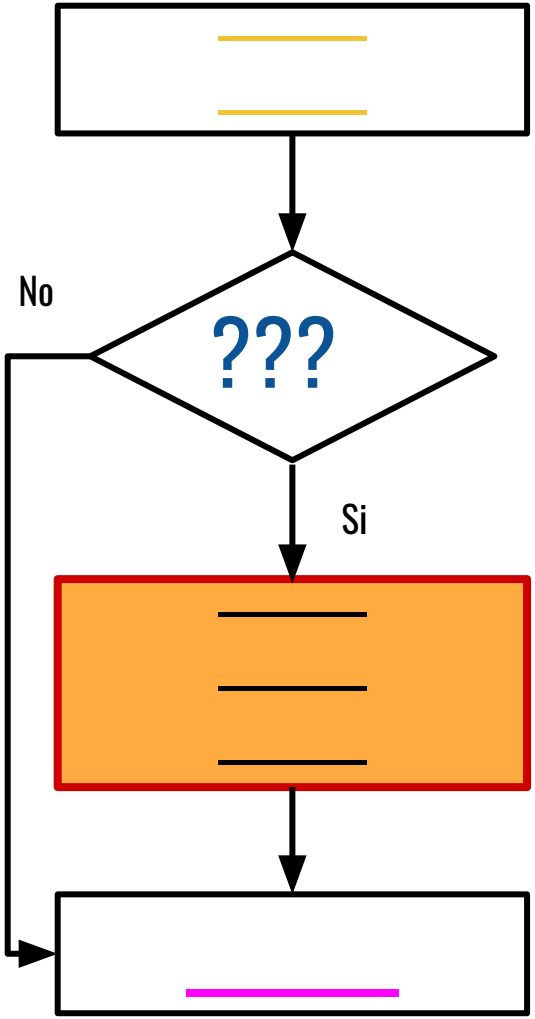
Finalmente, se ejecuta el código fuera del bloque `if`



Instrucción `if` en Python



En Python los bloques son definidos por identación



Ejercicio: Saludando

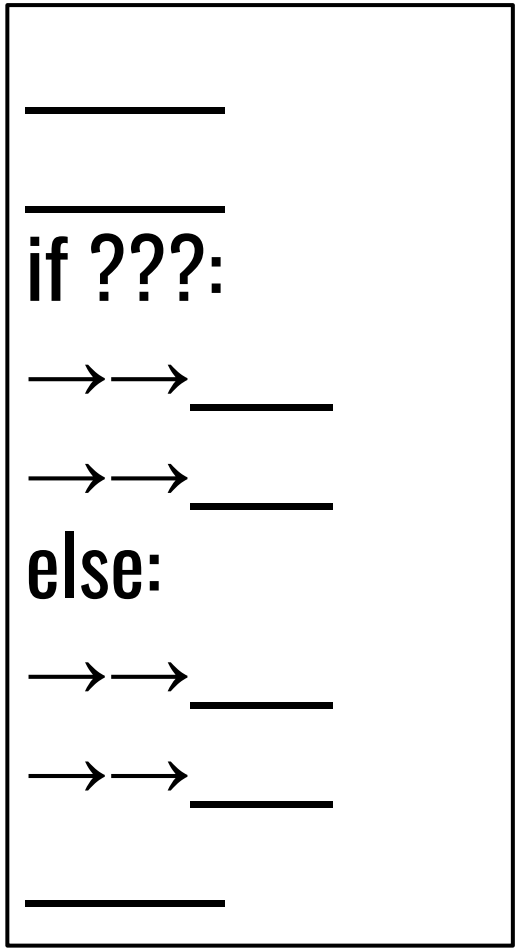
Escriba un programa en Python 3.x que lea un número entero que represente una hora en formato de 24h y sea capaz de decir buenos días,o buenas tardes(dependiendo de la hora) – asuma que solo tendremos horarios de la mañana y la tarde.

Hint:
Menor que 12: Buenos dias
Mayor o igual que 12: Buenas tardes

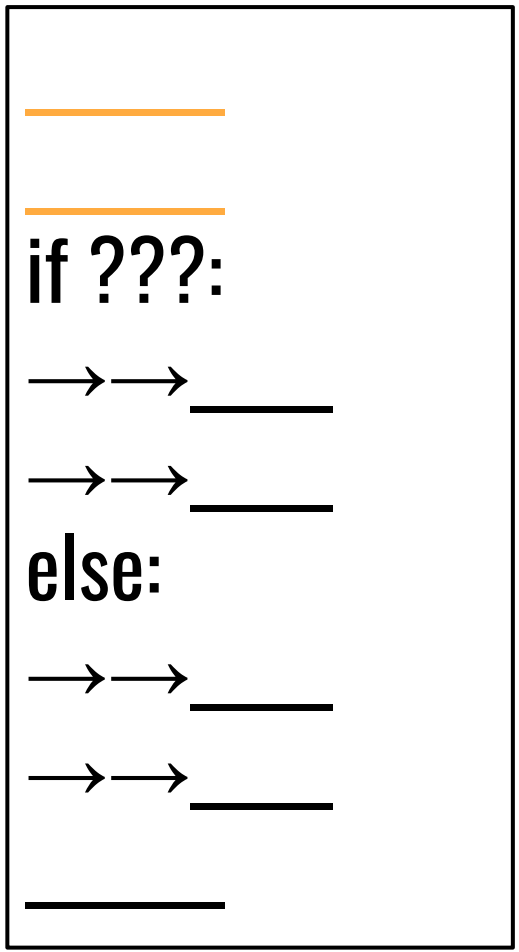
Ejercicio: Saludando

```
hora = int(input("Hora:"))
if hora < 12:
→→print("Buenos dias")
if hora>= 12:
→→print("Buenas tardes")
```

Instrucción `else`

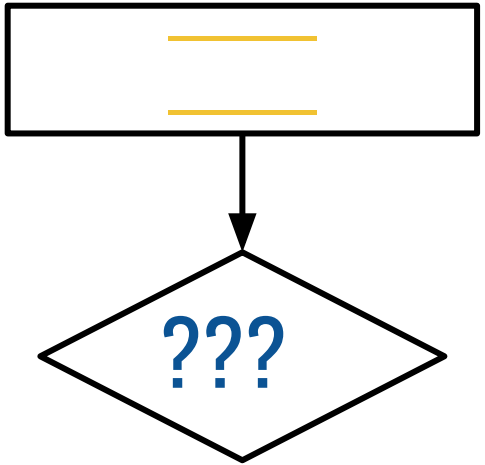


Instrucción `else`



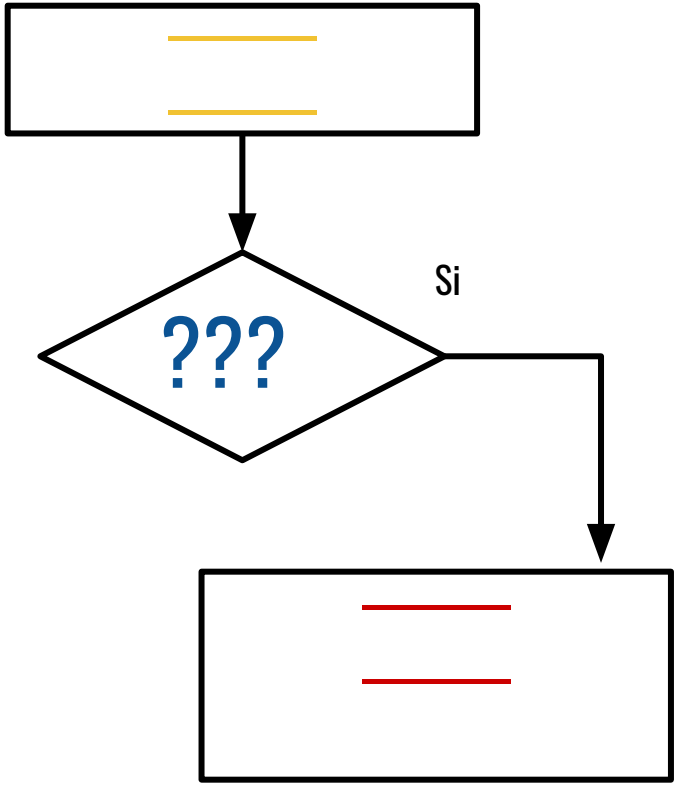
Instrucción `else`

```
_____

_____
if ???:
    ____
    ____
else:
    ____
    ____

_____
```



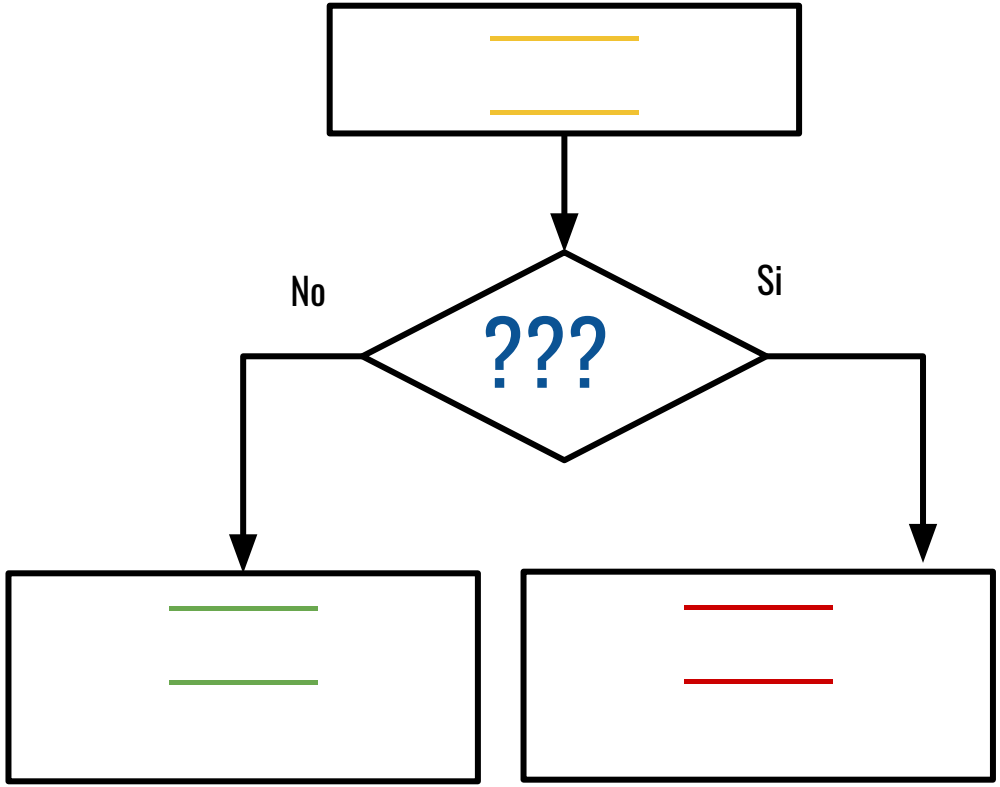
Instrucción `else`

```
_____
_____
if ???:
    ____
    ____
else:
    ____
    ____
_____
```



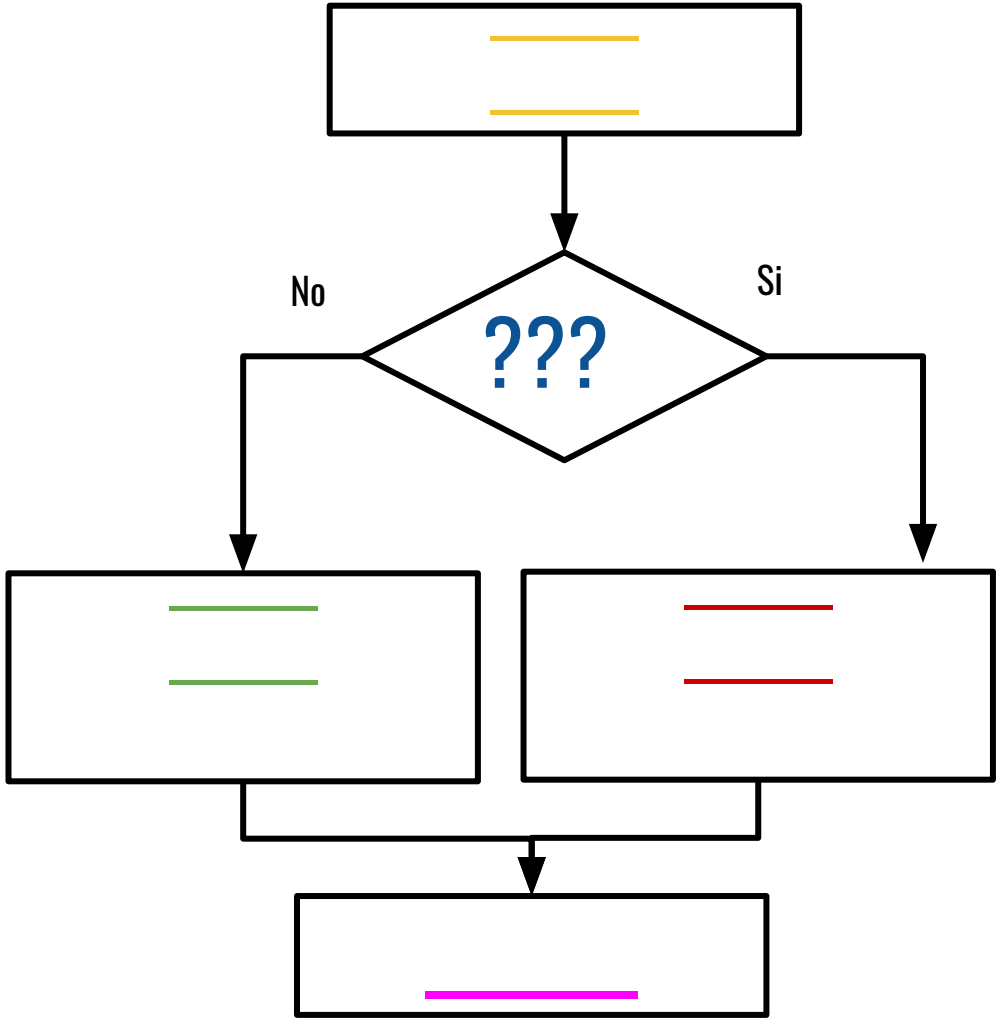
Instrucción `else`

```
_____

_____
if ???:
    ____
    ____
else:
    ____
    ____

_____
```

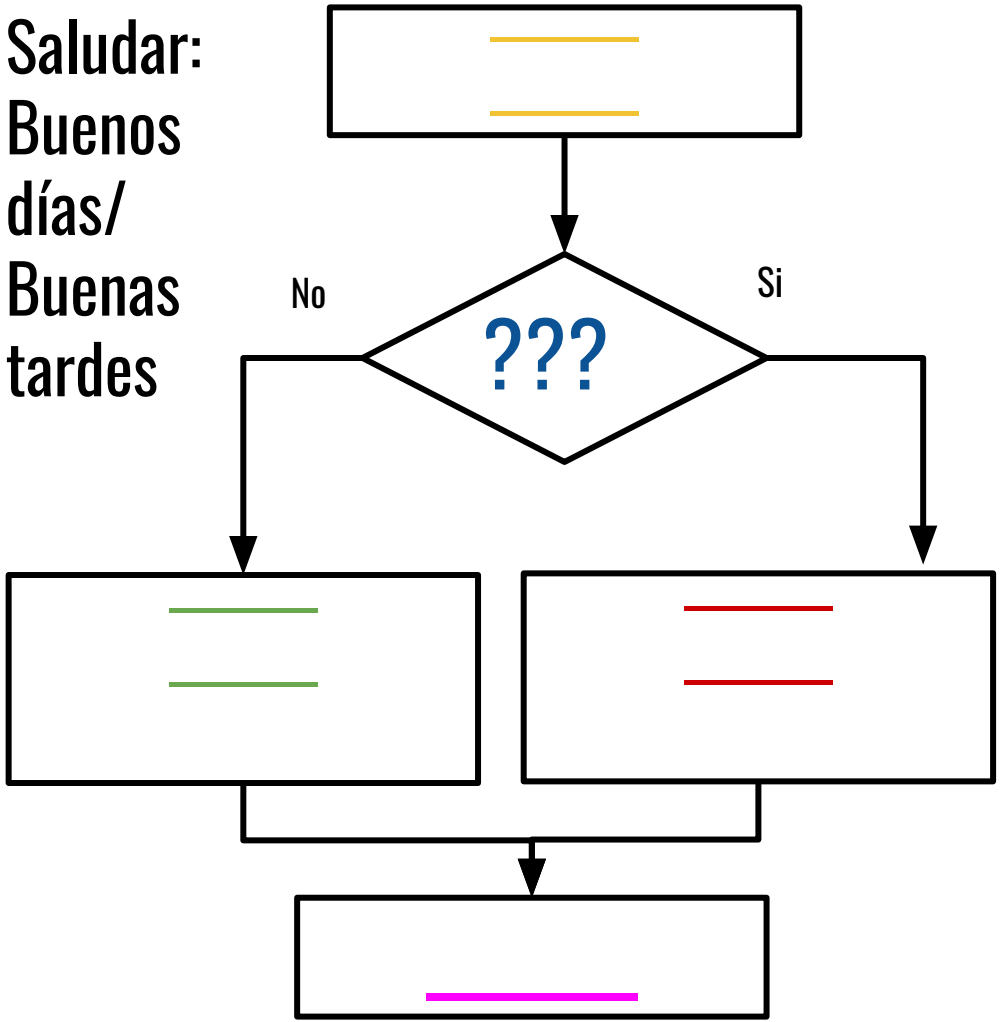


Instrucción `else`

```
_____
_____
if ???:
    ____
    ____
else:
    ____
    ____
_____
```



Saludar: Buenos días/ Buenas tardes



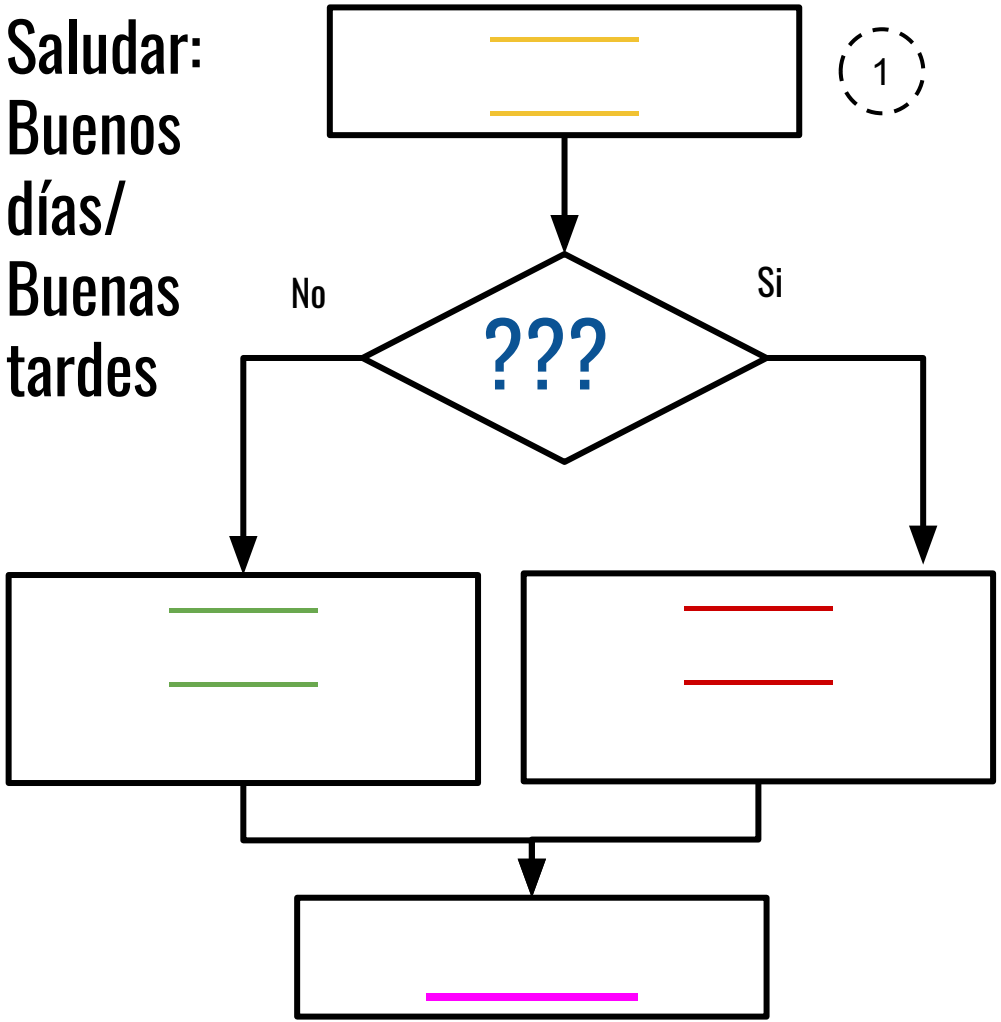
A) print(“Buenas tardes”)

B) hora = int(input(“Hora:”))

C) hora < 12

D) print(“Buenos días”)

Saludar: Buenos días/ Buenas tardes



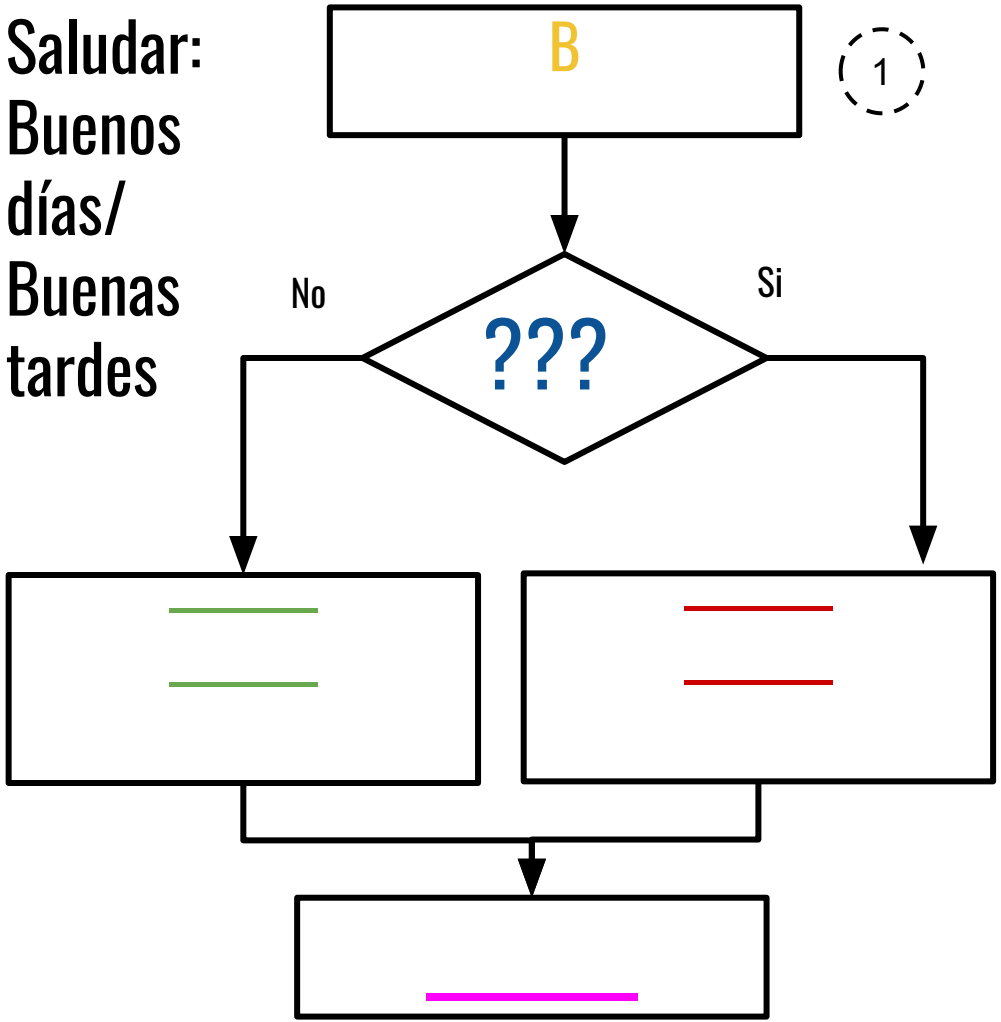
A) print(“Buenas tardes”)

B) hora = int(input(“Hora:”))

C) hora < 12

D) print(“Buenos días”)

Saludar: Buenos días/ Buenas tardes



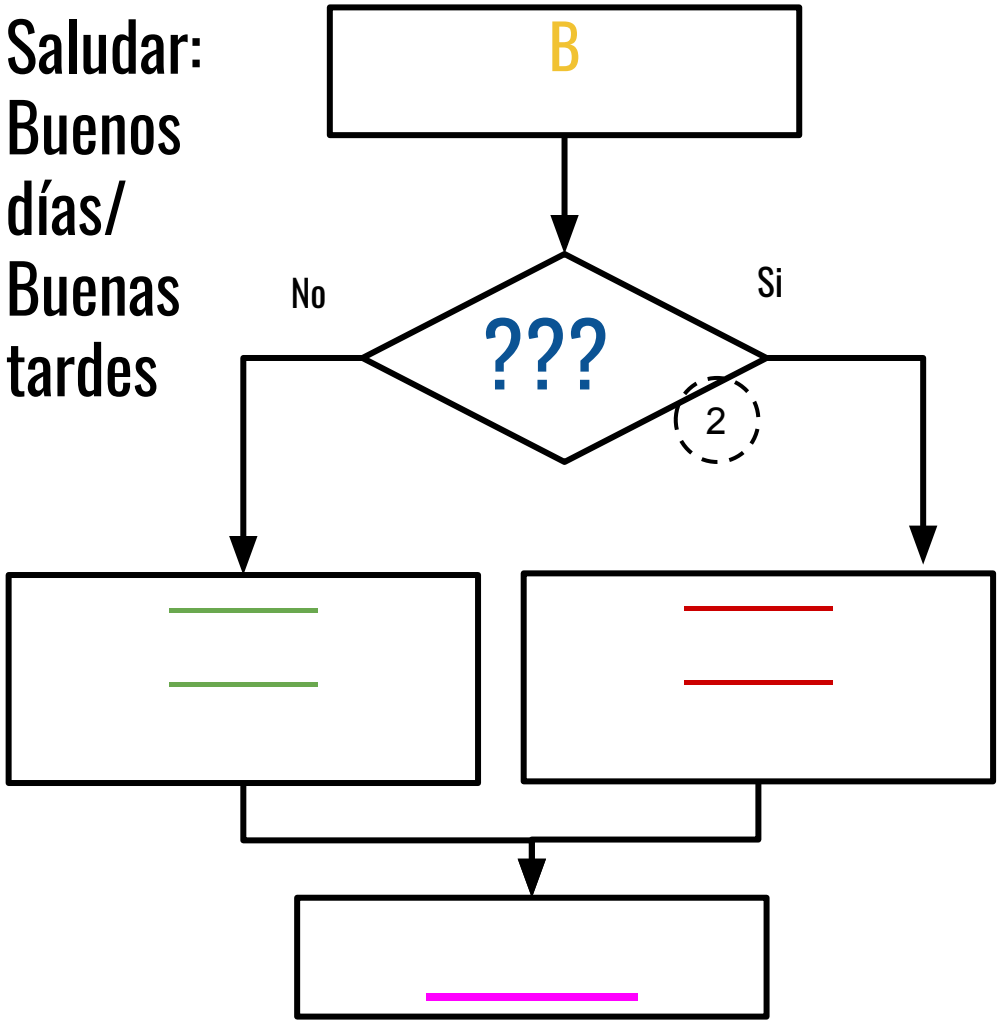
A) print(“Buenas tardes”)

B) hora = int(input(“Hora:”))

C) hora < 12

D) print(“Buenos días”)

Saludar: Buenos días/ Buenas tardes



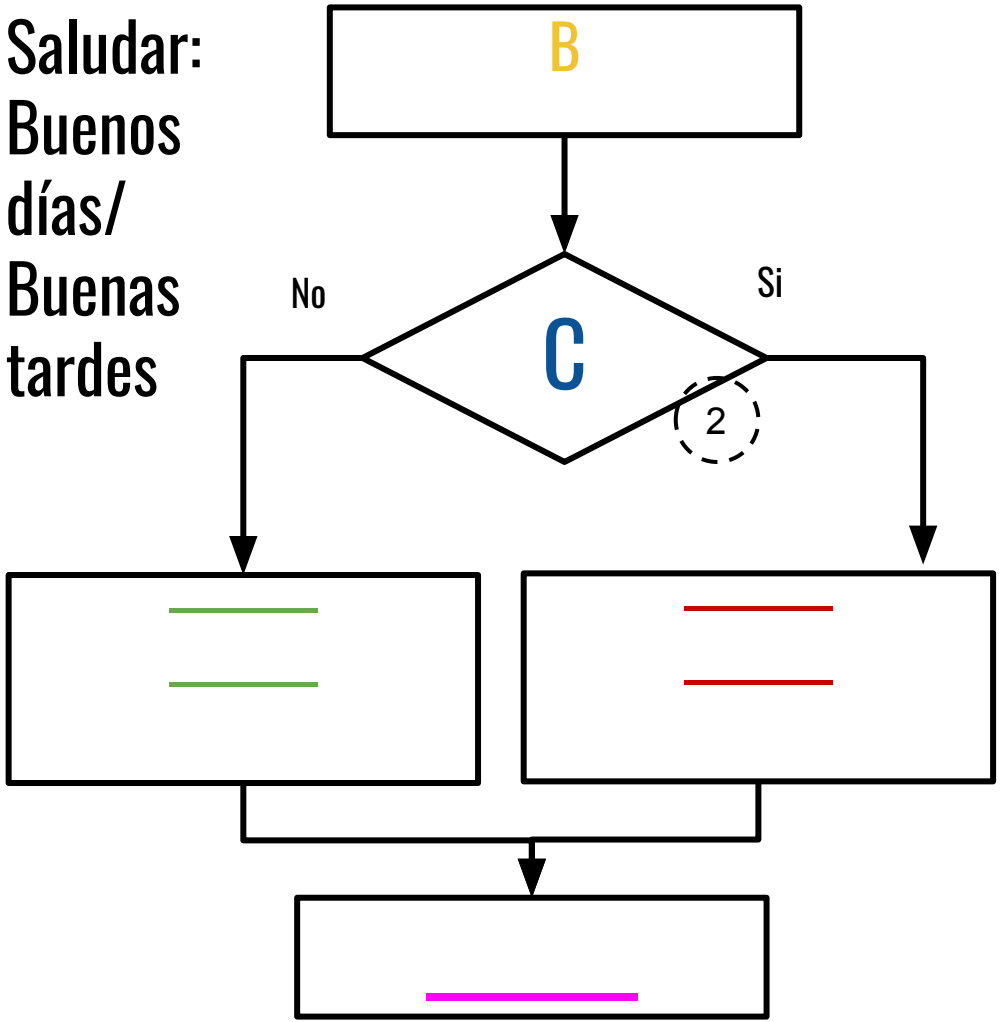
A) print(“Buenas tardes”)

B) hora = int(input(“Hora:”))

C) hora < 12

D) print(“Buenos días”)

Saludar: Buenos días/ Buenas tardes



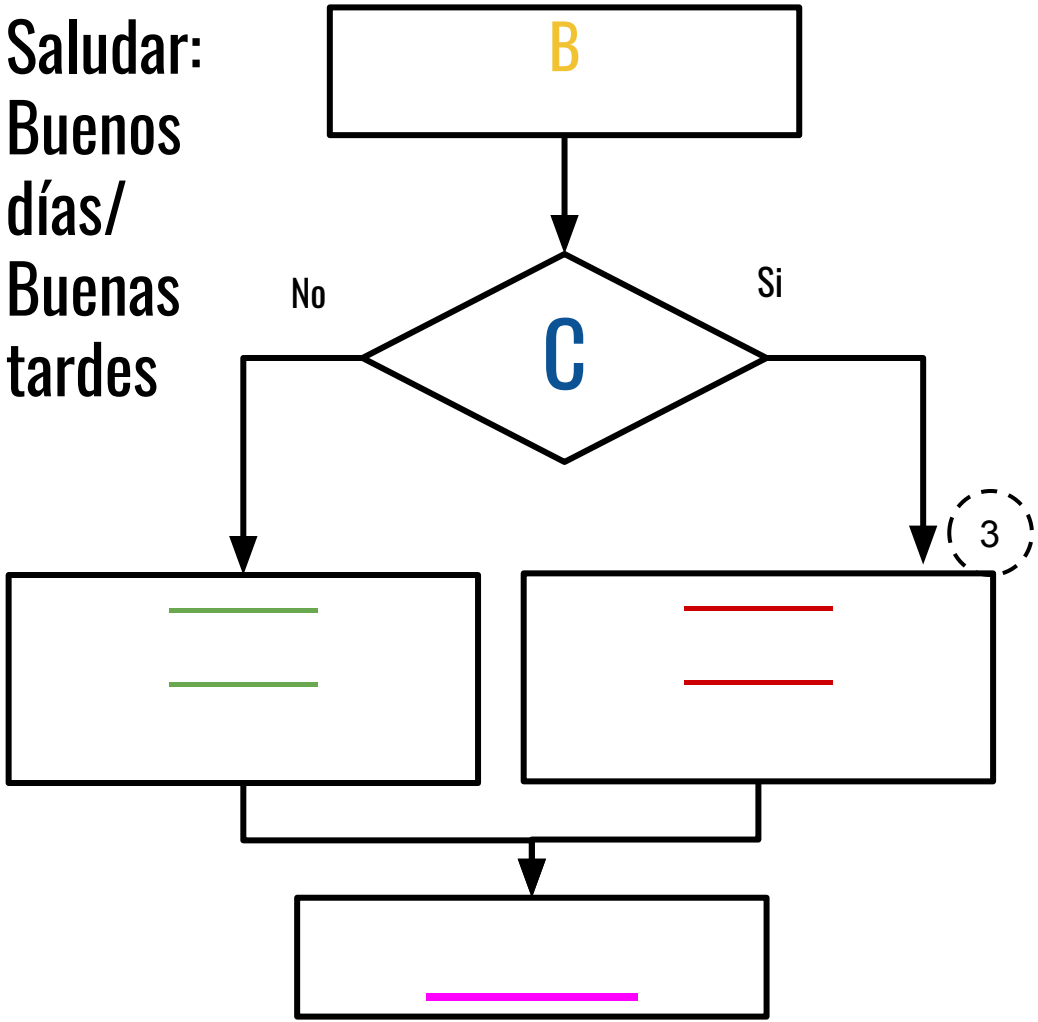
A) print(“Buenas tardes”)

B) hora = int(input(“Hora:”))

C) hora < 12

D) print(“Buenos días”)

Saludar: Buenos días/ Buenas tardes



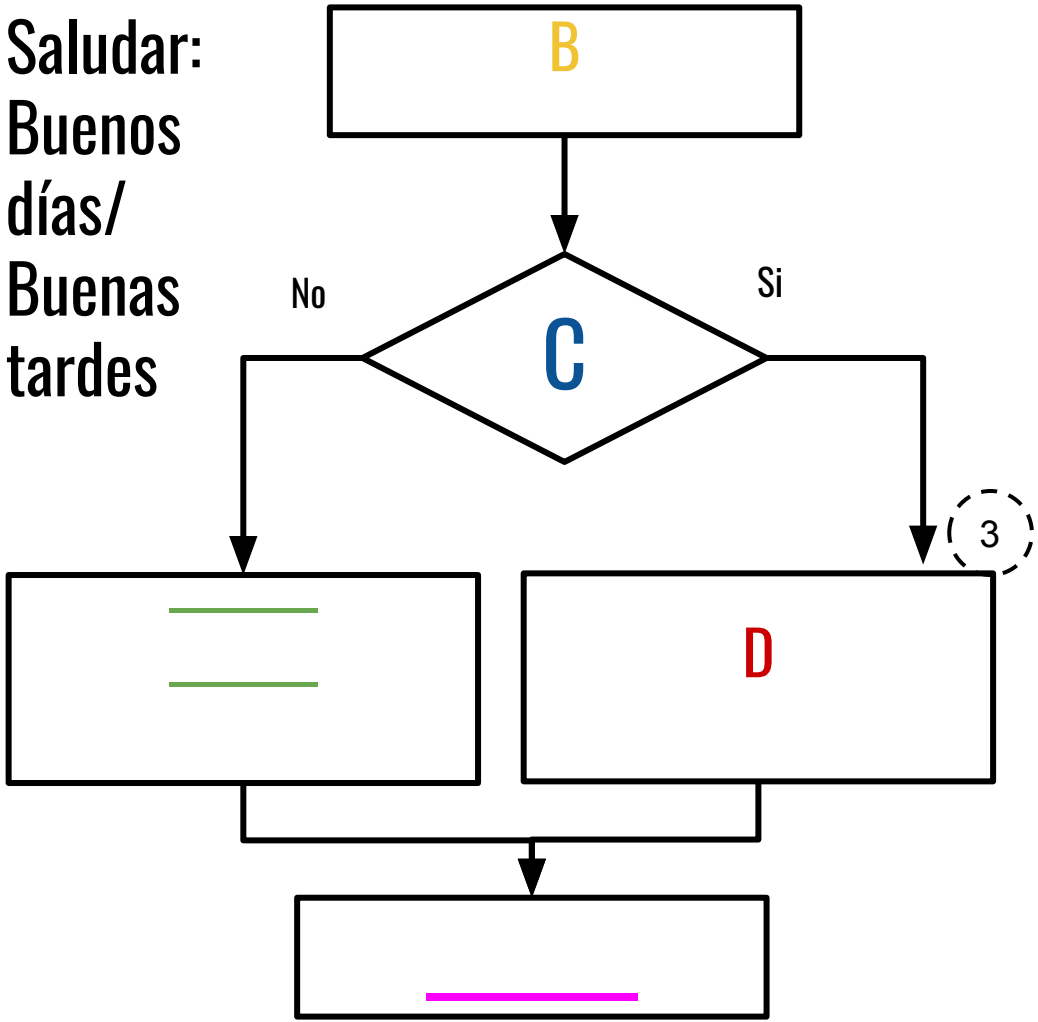
A) print(“Buenas tardes”)

B) hora = int(input(“Hora:”))

C) hora < 12

D) print(“Buenos días”)

Saludar: Buenos días/ Buenas tardes



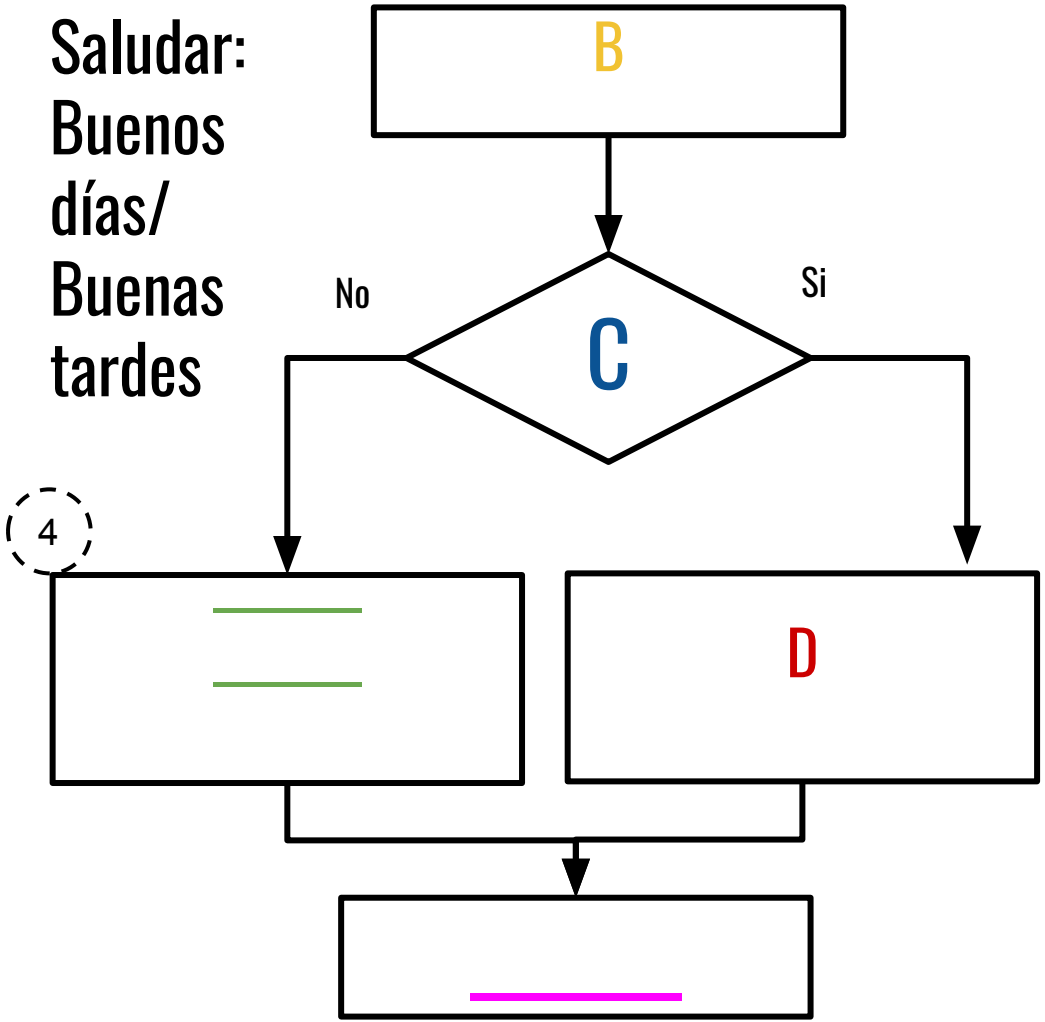
A) print(“Buenas tardes”)

B) hora = int(input(“Hora:”))

C) hora < 12

D) print(“Buenos días”)

Saludar: Buenos días/ Buenas tardes



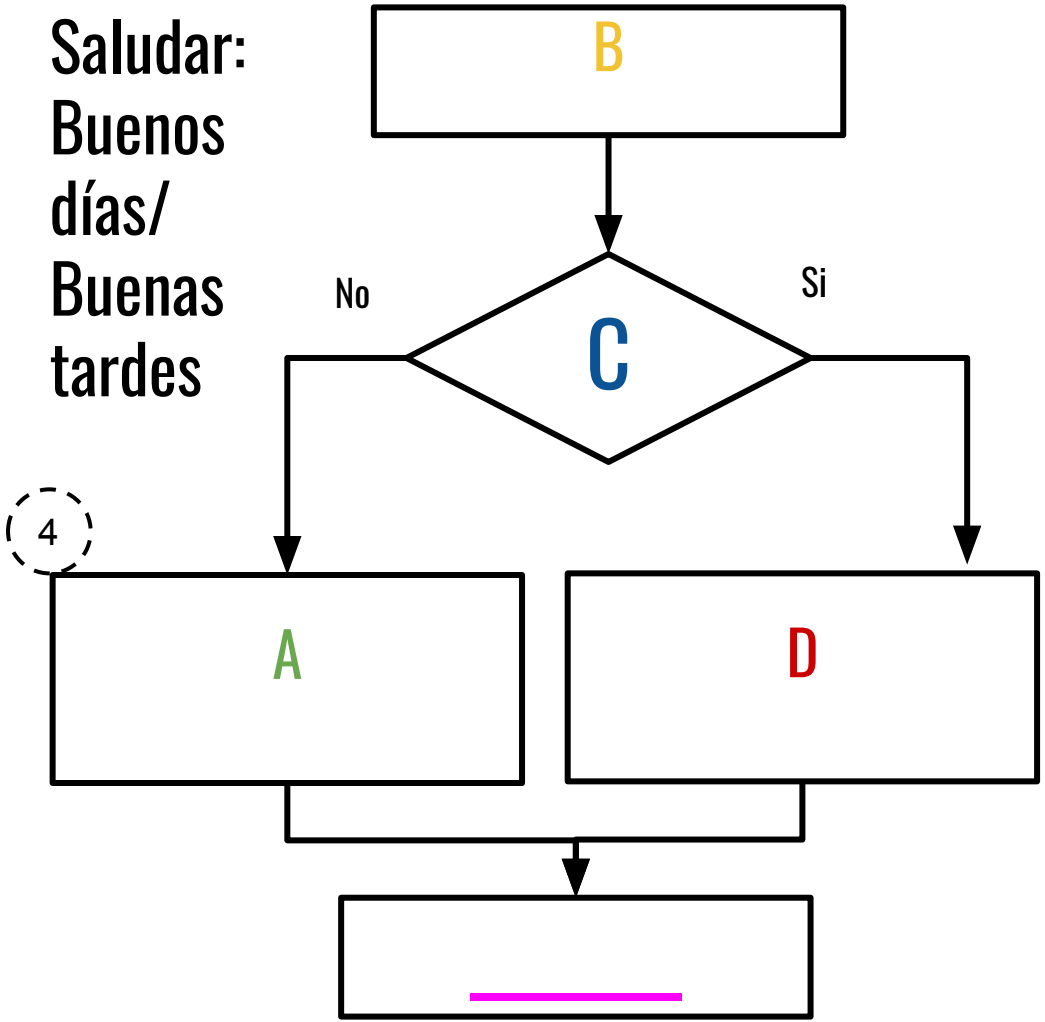
A) print(“Buenas tardes”)

B) hora = int(input(“Hora:”))

C) hora < 12

D) print(“Buenos días”)

Saludar: Buenos días/ Buenas tardes



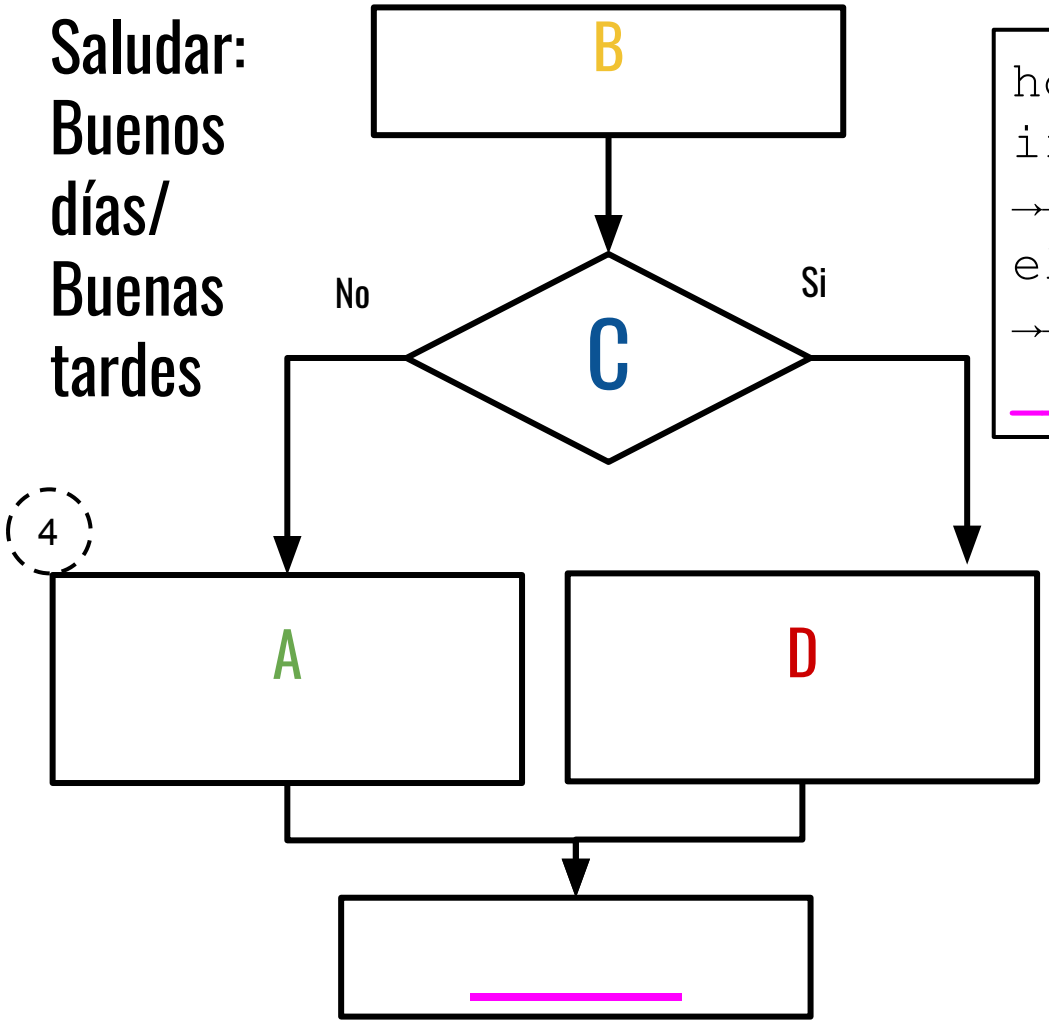
A) print(“Buenas tardes”)

B) hora = int(input(“Hora:”))

C) hora < 12

D) print(“Buenos días”)

Saludar: Buenos días/ Buenas tardes



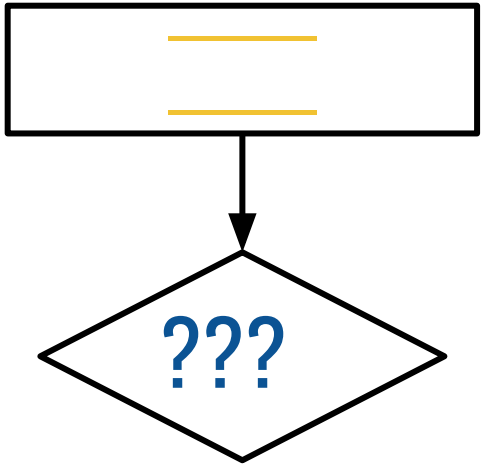

```
hora = int(input("Hora:"))
if hora < 12:
→→print("Buenos dias")
else:
→→print("Buenas tardes")
```

Instrucción `elif`

```
if ???:
    ____
elif ???:
    ____
else:
    ____

____
```

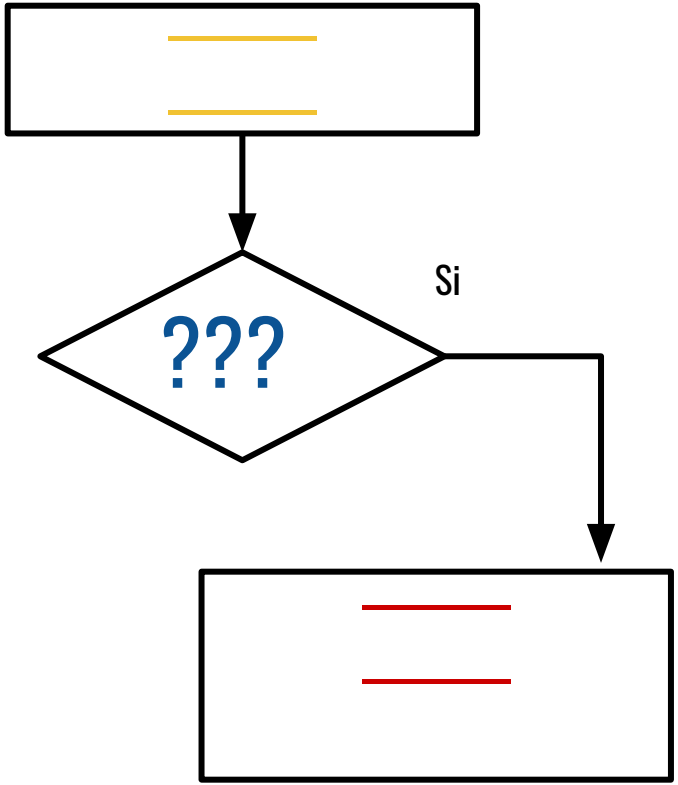
Instrucción `elif`

```
_____
if ???:
→→____
elif ???:
→→____
else:
→→____

_____
```



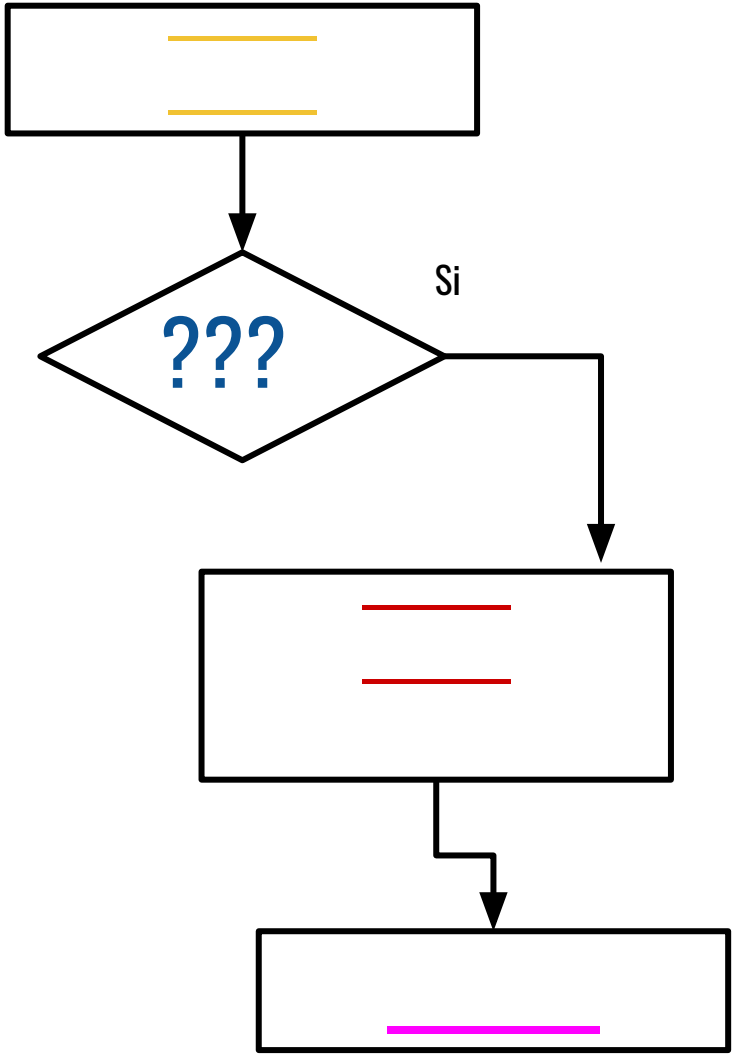
Instrucción `elif`

```
_____
if ???:
→→____
elif ???:
→→____
else:
→→____

_____
```



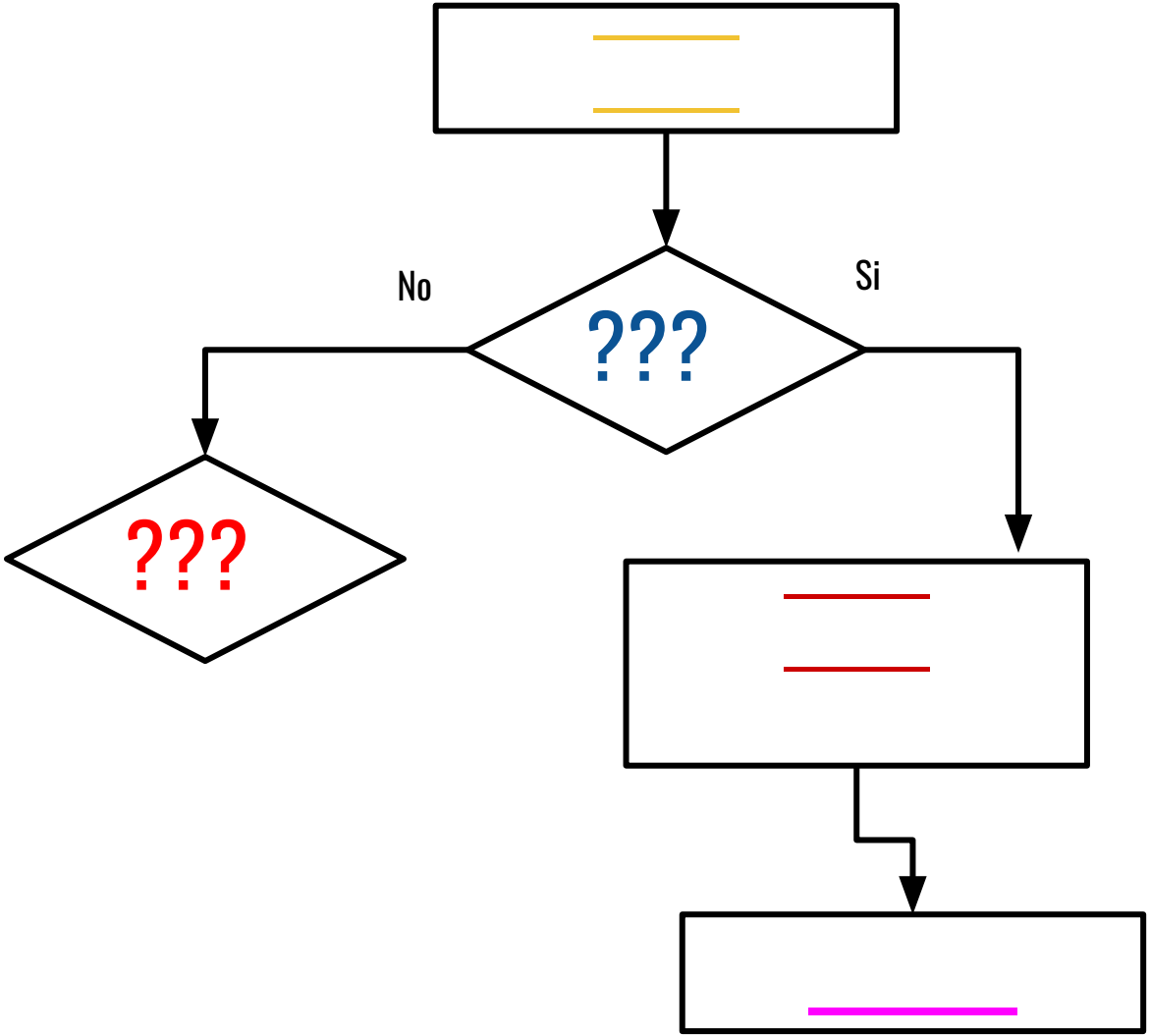
Instrucción `elif`

```
_____
if ???:
→→____
elif ???:
→→____
else:
→→____

_____
```



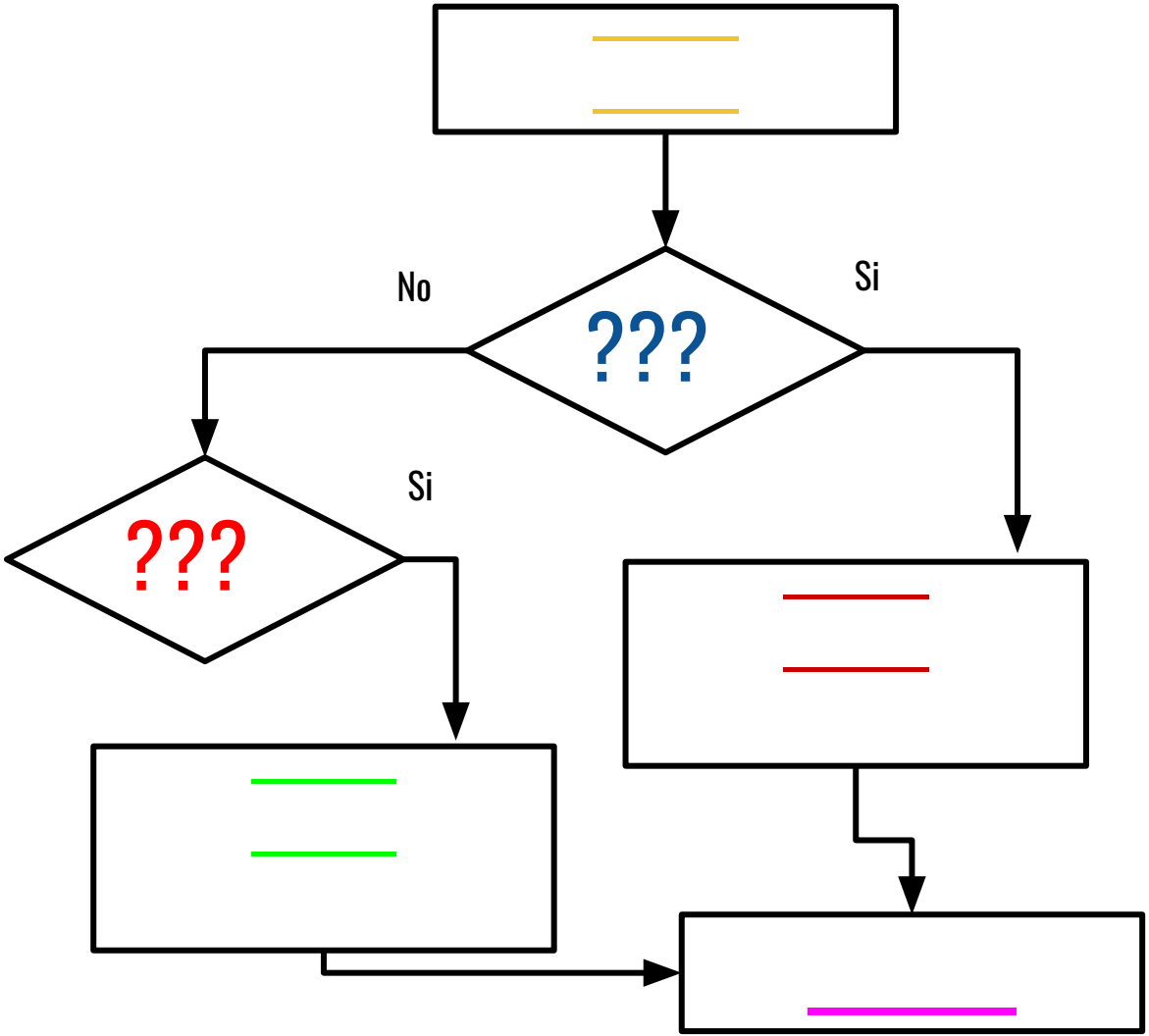
Instrucción `elif`

```
if ???:
    ____
elif ???:
    ____
else:
    ____

____
```



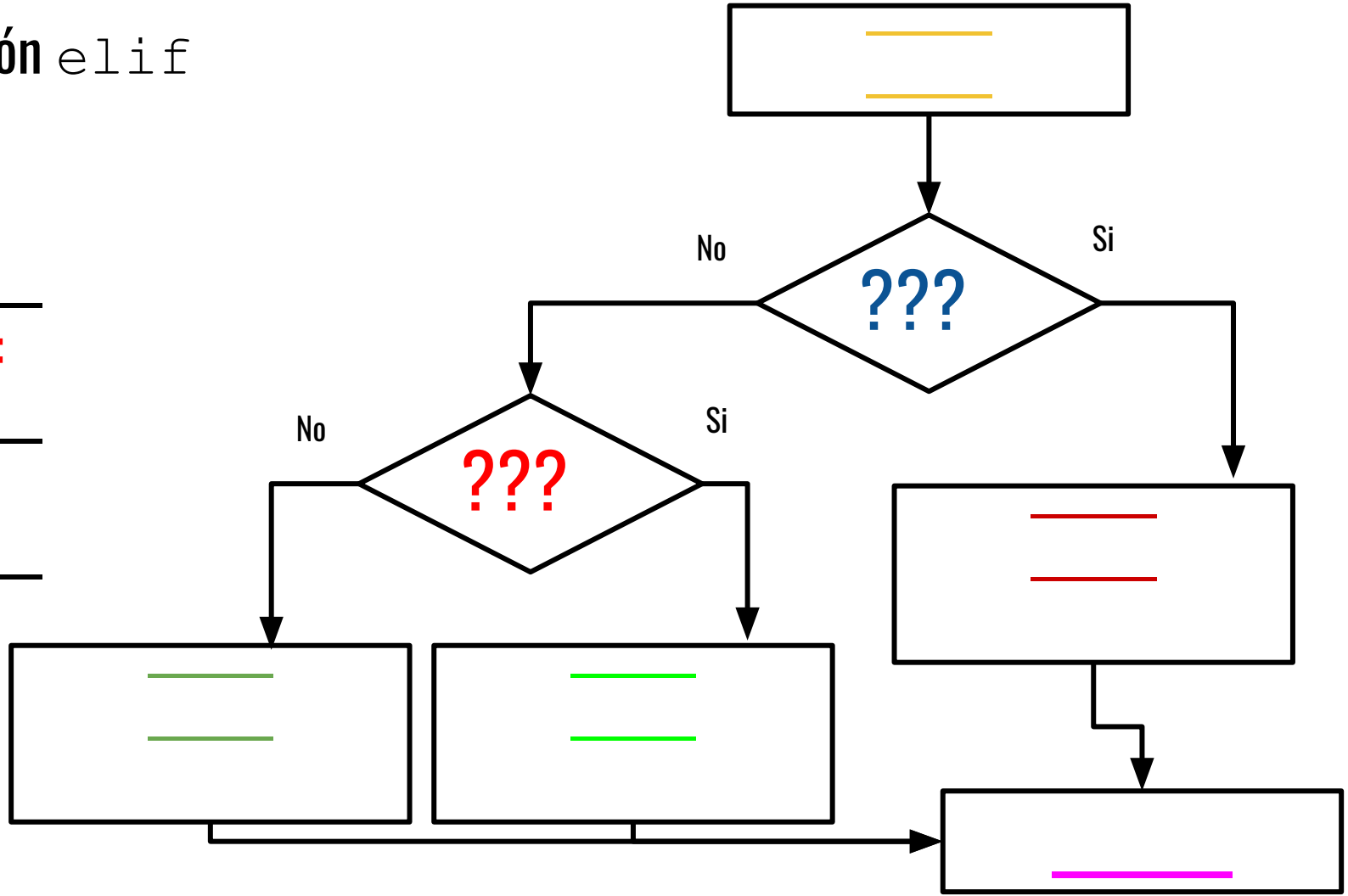
Instrucción `elif`

```
if ???:
    ____
elif ???:
    ____
else:
    ____

____
```



Instrucción `elif`

```
if ???:
    ____
elif ???:
    ____
else:
    ____

____
```



Instrucción `elif`

`elif` permite agregar nuevas condiciones