

# Parte I: Intro pensamiento computacional

## Clase 03: Ciclos

Daniela Opitz, Diego Caro  
[dopitz@udd.cl](mailto:dopitz@udd.cl)



Basada en presentaciones oficiales de libro Introduction to Programming in Python (Sedgewick, Wayne, Dondero).

Disponible en <https://introcs.cs.princeton.edu/python>

# Clase de Hoy

- Diagramas de flujo para if - elif - else
- Conversion de tipos (repaso)
- Uso de if para chequear errores
- Uso de while

# Condicionales: obedeciendo reglas

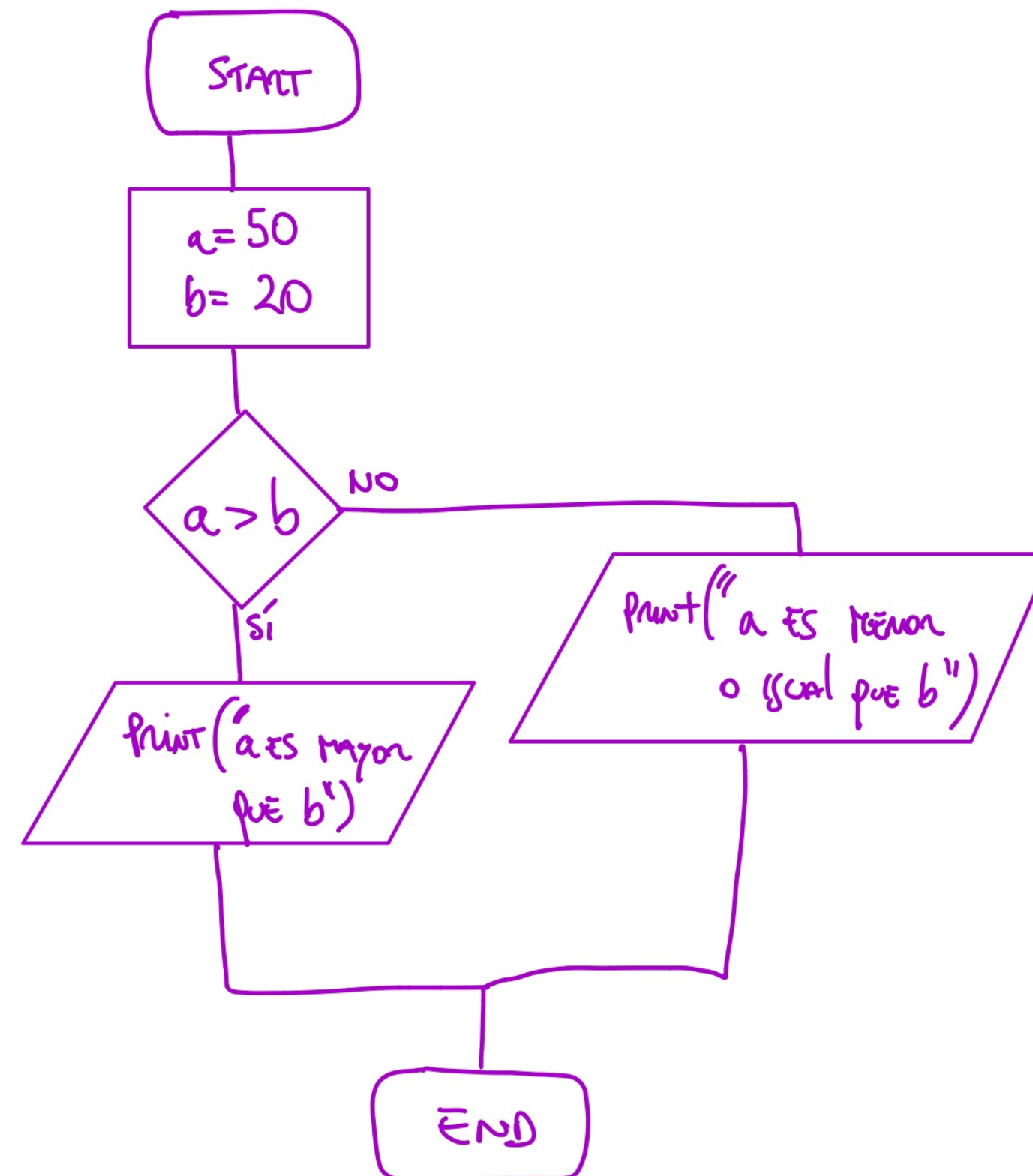
- Mundo ideal: "Siri, escribe el código para el proyecto del curso de programación"
  - No. Faltan siglos para eso.
  - (comentario: Mmm podría ser, pero solo si alguna ingeniera de Siri ya diseñó un programa para que resolviera el proyecto del curso de programación)
- Mundo real:
  1. Descomponer el problema en etapas
  2. Para cada etapa reconocer qué reglas se deben cumplir
  3. Ejecutar acciones por cada regla
- En Python: condicionar la ejecución de código a una regla
  - if statement
  - Si la condición se cumple, ejecuta el código del bloque
  - Línea 4 -> bloque de código!

```
1 a = 50
2 b = 10
3 if a > b:
4     print('a es mayor que b')
```

# Diagramas de flujo

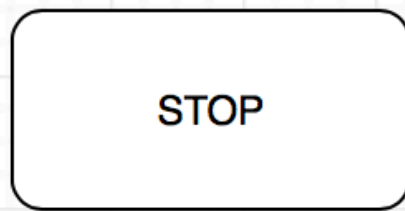
- Un diagrama de flujo representa visualmente un algoritmo.
- Podemos mapear directamente un diagrama de flujo a código en Python (y viceversa).

```
a = 50
b = 20
if a > b:
    print('a es mayor que b')
else:
    print('a es menor o igual que b')
```



## 1. Delimitadores

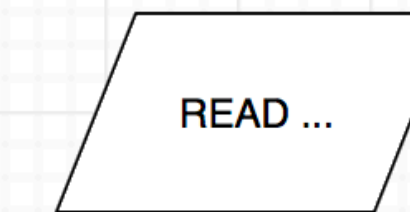
Comienzo y final de algoritmo



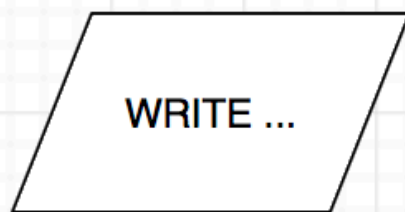
## 2. Datos

Input y Output

x = input()

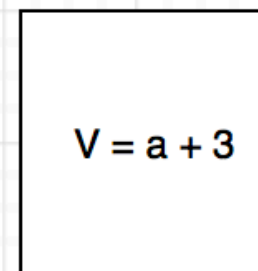


print(x)



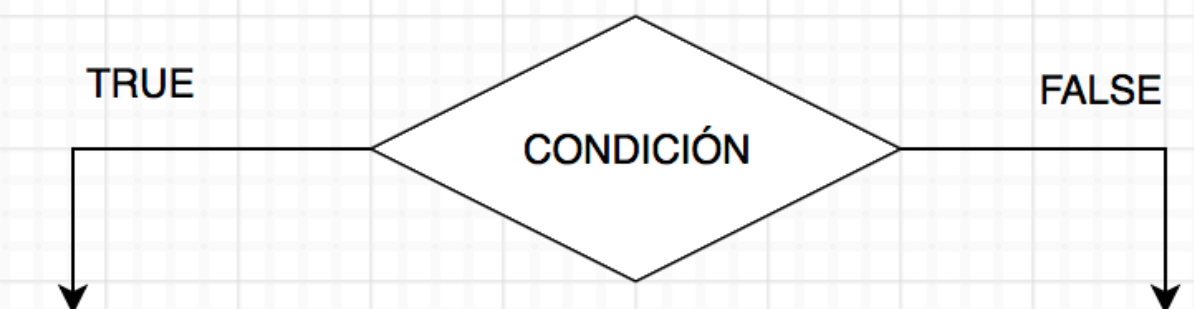
## 3. Procesos

Variables



## 4. Decisión

Condicionales





# Año biciesto

- Pregunta: ¿Cuándo un año es bisiestro?

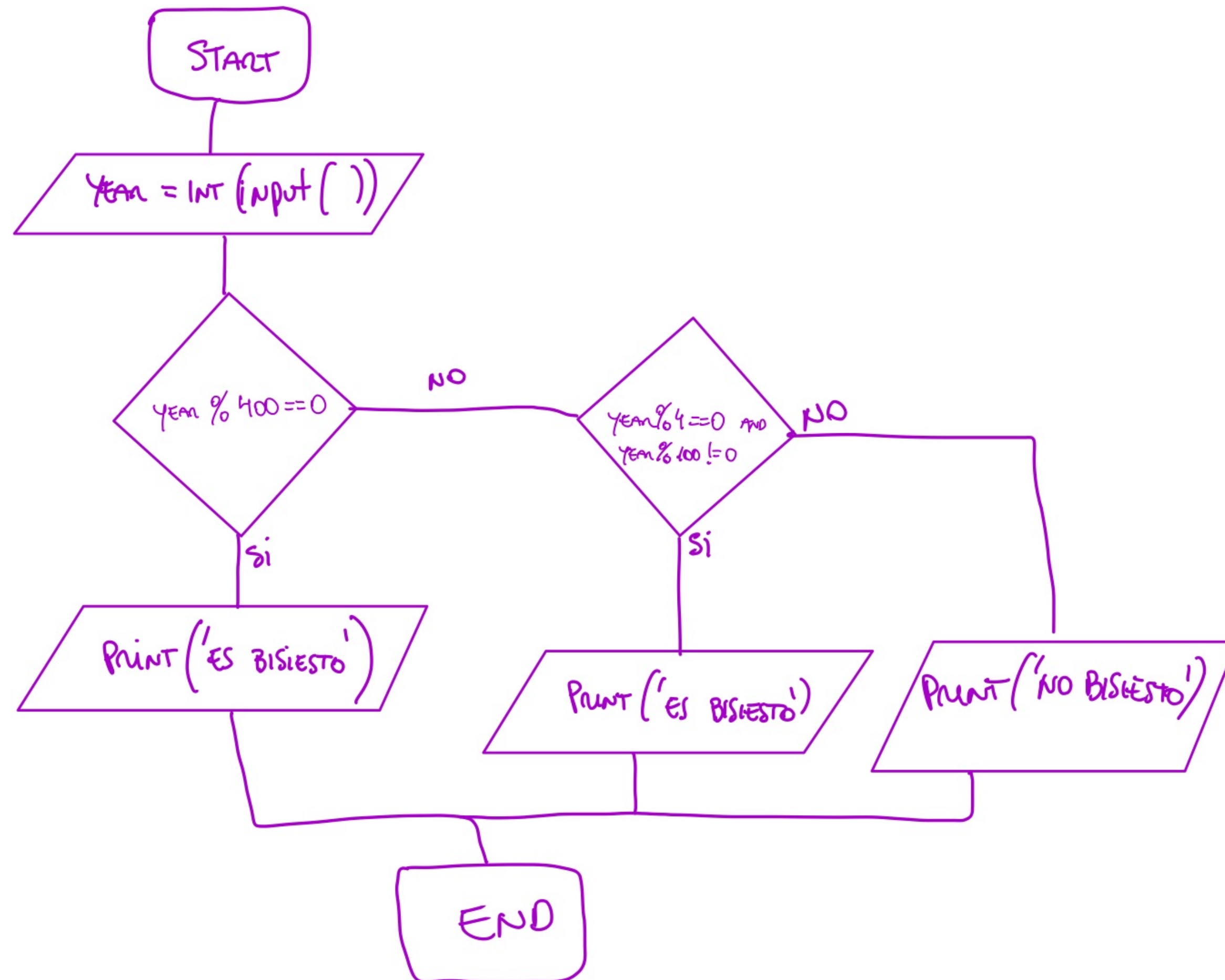
- Respuesta: existen dos casos

- Caso 1: Cuando es divisible por 400

→  $\text{year \% } 400 == 0$

- Caso 2: Cuando es divisible por 4 pero no por 100

→  $\text{year \% } 4 == 0 \text{ and } \text{year \% } 100 != 0$



Las '{}' se  
reemplazan  
por year

```
year = int(input('Ingrese año: '))
if year%400 == 0:
    print('{} es bisiestro'.format(year))
elif year%4 == 0 and year % 100 != 0:
    print('{} es bisiestro'.format(year))
else:
    print('{} NO es bisiestro'.format(year))
```

# Alternativa

**Alternativa 2: juntando ambas condiciones con operador or**

```
year = int(input('Ingrese año: '))
if year%400 == 0 or (year%4 == 0 and year % 100 != 0):
    print('{} es bisiesto'.format(year))
else:
    print('{} NO es bisiesto'.format(year))
```

# Conversión de Tipos

- **Explícito:** directamente en el código
  - Convertir texto a entero
  - Convertir entero a float
  - Convertir flotante a entero
- **Implícito:** *automágico* por Python\*
  - Operaciones entre números de distinto tipo
  - Multiplicación entre un entero n y un string s devuelve el string s concatenado n veces.

```
1 # tipo-explicit.py
2 print(type('2'))
3 print(type(int('2')))
4 print(type(float('2')))
5 print(type(float(2)))
6 print(type(str(2)))
7 print(type(str(2.0)))
```

**Salida:**

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'float'>
<class 'str'>
<class 'str'>
```

**P: ¿Cuál es el resultado de la línea 6 y 7?**

```
1 # tipo-implicit.py
2 print(type(2+2.0))
3 print(type(2+float(2)))
4 print(type(2*'hola'))
```

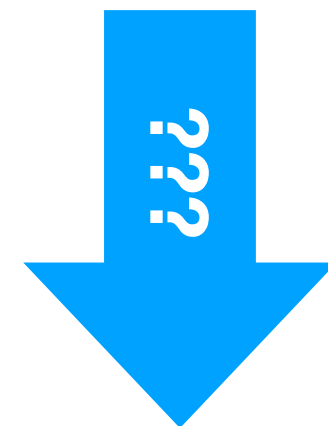
**Salida:**

```
<class 'float'>
<class 'float'>
<class 'str'>
```

**P: ¿Cuál es el resultado de la línea 4?**

# Conversion de Tipos

```
1 # pizzas.py
2 i = input('¿Cuántas pizzas individuales desea?: ')
3 m = input('¿Cuántas pizzas mediantas desea?: ')
4 f = input('¿Cuántas pizzas familiares desea?: ')
5 total = 4600*i + 7850*m + 10750*f
6 print('Total a pagar:', total)
```

[illegible]

## ¿Por qué se imprimen muchos 1s?

Dado que la variable i, m y f son de tipo string (notar que no se convierten a int), la multiplicación definida en la 5 repite 4600 veces lo que está en la variable i, 7875 veces lo que está en la variable m, y 10750 veces lo que está en la variable f.

Name ▲	Type	Size	Value
f	str	1	0
i	str	1	1
m	str	1	0

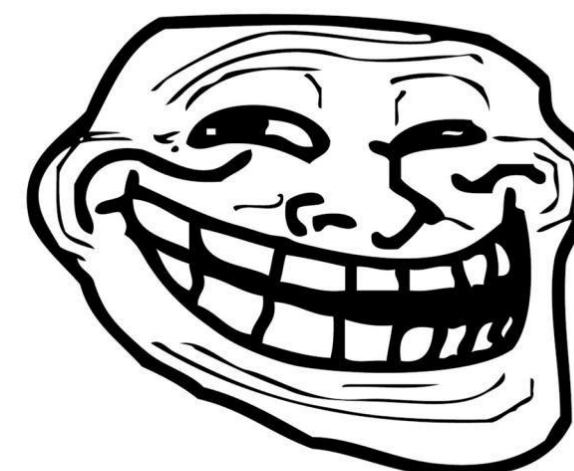
## Variables son de tipo str...



# Chequeo de Errores



```
$ python3 pizzas2.py
¿Cuántas pizzas individuales desea?: 2
¿Cuántas pizzas mediantas desea?: -1
¿Cuántas pizzas familiares desea?: 0
Erro en el número de pizzas
Total a pagar: 1350
```



**problem?**

# Uso de if para chequear errores

## Solución

```
1 # 04_pizzas2.py
2 i = int(input('¿Cuántas pizzas individuales desea?: '))
3 m = int(input('¿Cuántas pizzas mediantas desea?: '))
4 f = int(input('¿Cuántas pizzas familiares desea?: '))
5
6 if i < 0 or m < 0 or f < 0:
7     print('Error en el número de pizzas, intently otra vez')
8 else:
9     total = 4600*i + 7850*m + 10750*f
10    print('Total a pagar:', total)
11
```

# ¿Que imprime este código?

- ¿Qué hace este programa?

```
1 a = int(input())
2 b = int(input())
3 if b < a:
4     t = b
5     b = a
6     a = t
7 print(a)
8 print(b)
```

<http://www.pythontutor.com/visualize.html>

# Ciclo **while**

- **while** permite repetir la ejecución de un grupo de instrucciones mientras se cumpla una condición (es decir, mientras la condición tenga el valor **True**).

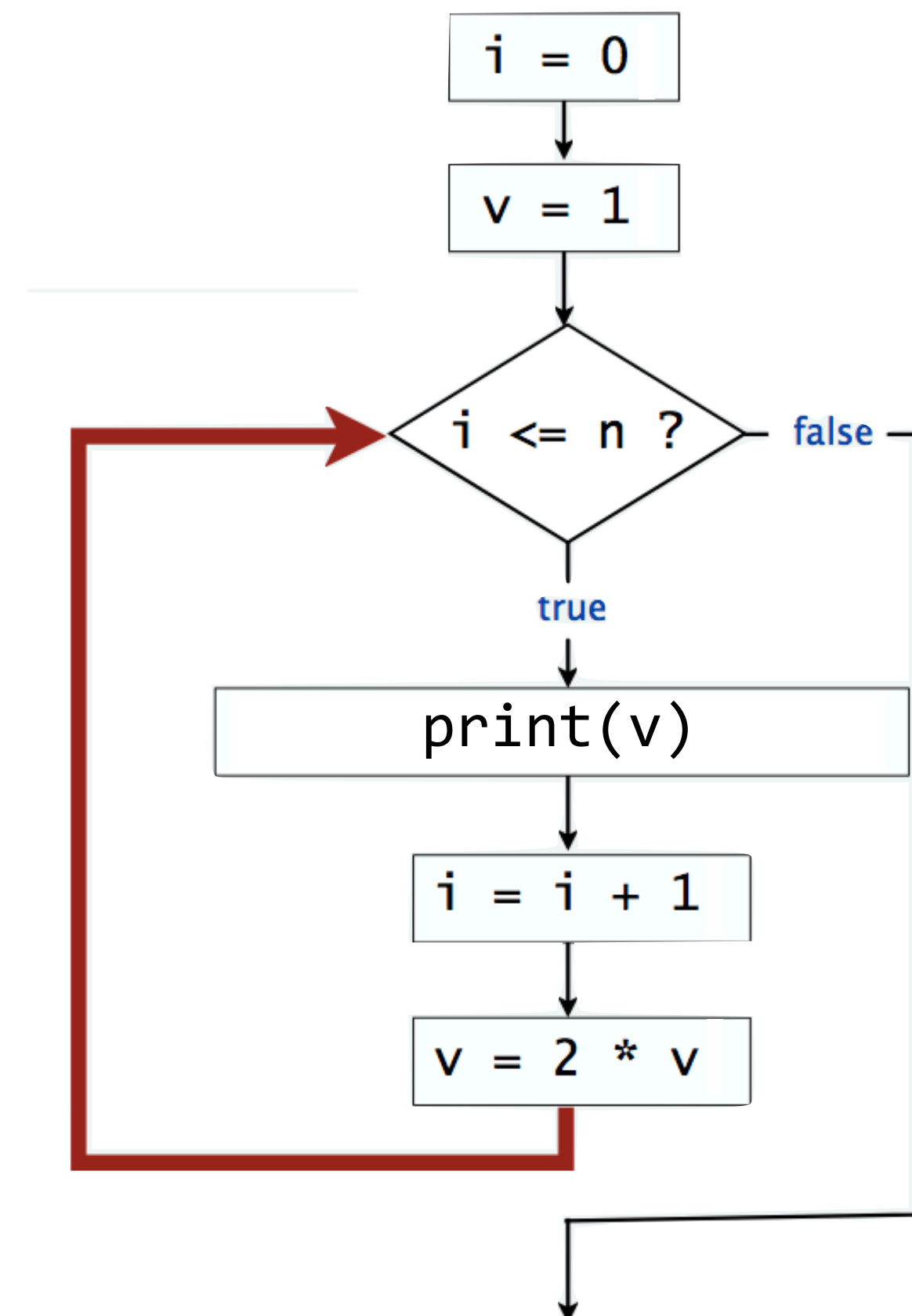
- Sintaxis:

**while** <condición>:  
    <instrucción>

- Ejemplo: Imprime las potencias desde 2<sup>0</sup> a 2<sup>n</sup>

```
i = 0
v = 1
while i <= n:
    print(v)
    i = i + 1
    v = 2*v
```

Expresión booleana




# Ciclo **while**


## Input

```
1 # n desde teclado
2 n = int(input())
3 i = 0
4 v = 1
5 while i <= n:
6     print(v)
7     i = i + 1
8     v = 2*v
```

## Traza



i	v	i <= n
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	true
7	128	false



## Output

```
1
2
4
8
16
32
64
128
```



# Ejercicio

- Escriba un programa que imprima la tabla del 7
- Escriba un programa que imprima las tablas del 1 al 12