

Parte I: Intro pensamiento computacional

Clase 03: Ciclos

Daniela Opitz, Diego Caro
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

- Conversion de tipos
- Uso de if para chequear errores
- Uso de la consola
- 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')
```

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'))
```

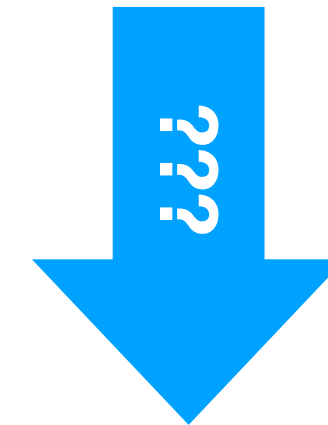
Salida:

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

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

Conversion de Tipos

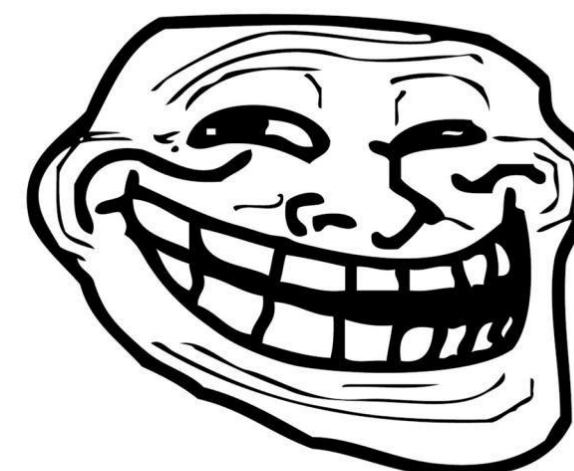
```
1 # pizzas.py
2 i = input('¿Cuántas pizzas individuales desea?: ')
3 m = input('¿Cuántas pizzas mediantas desea?: ')
4 f = input('¿Cuántas pizzas familiares desea?: ')
5 total = 4600*i + 7850*m + 10750*f
6 print('Total a pagar:', total)
```

[illegible]

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

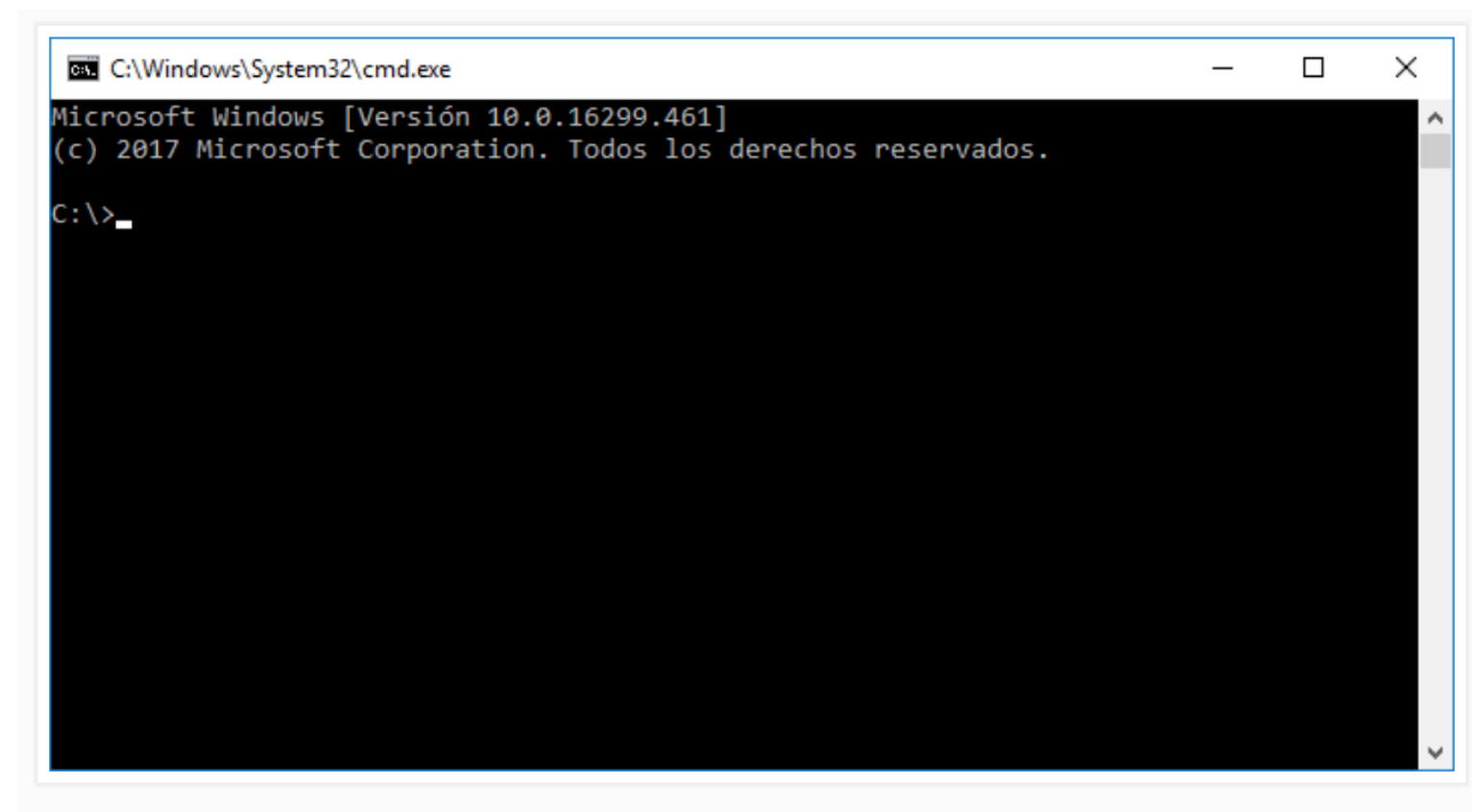
```
1 # 04_pizzas2.py
2 i = int(input('¿Cuántas pizzas individuales desea?: '))
3 m = int(input('¿Cuántas pizzas mediantas desea?: '))
4 f = int(input('¿Cuántas pizzas familiares desea?: '))
5
6 if i < 0 or m < 0 or f < 0:
7     print('Error en el número de pizzas, intently otra vez')
8 else:
9     total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

Uso de la Consola

- **Windows**

>>**cd**: Cambia a otro directorio.

>>**dir**: Muestra una lista de archivos y subdirectorios en un directorio

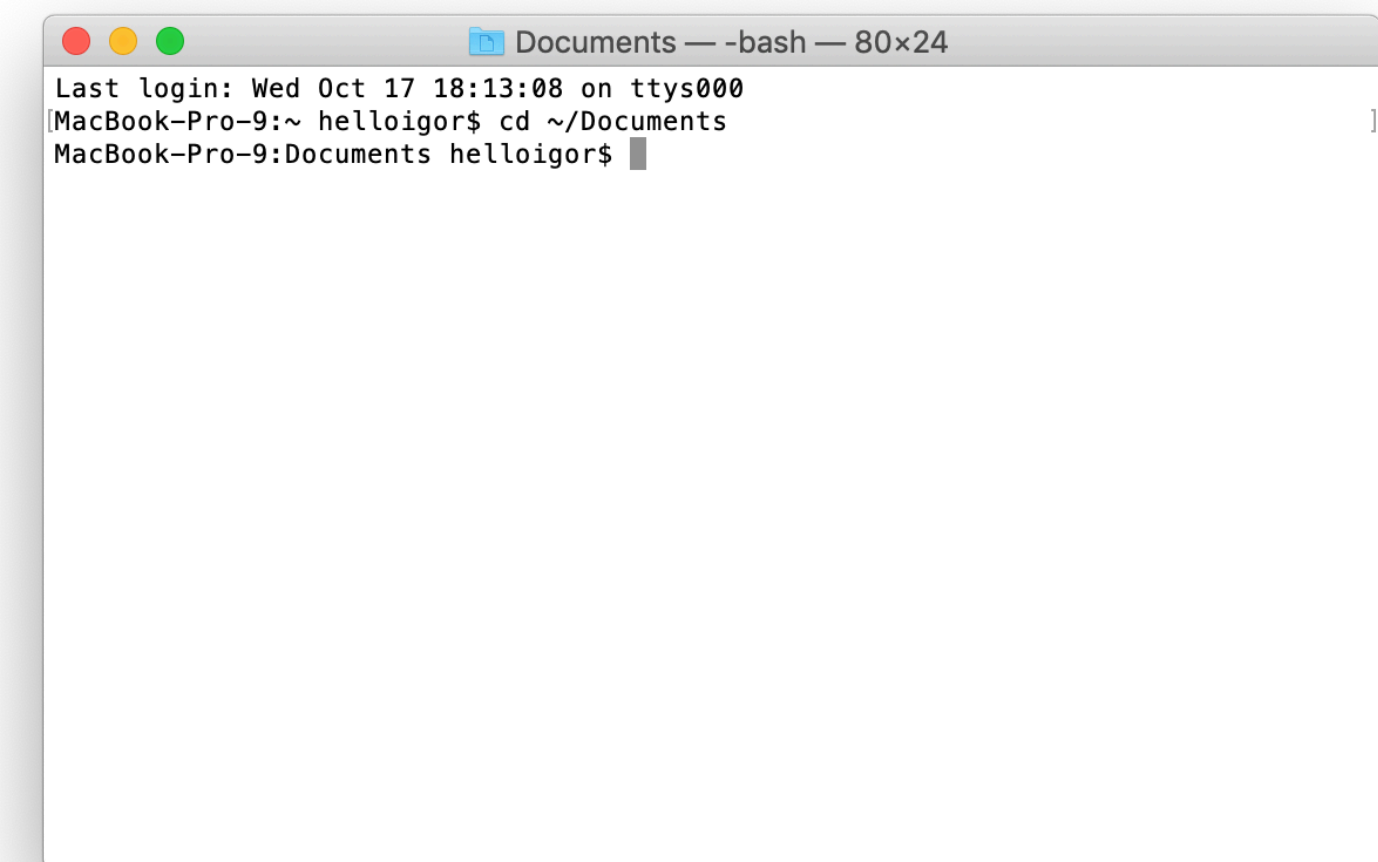


cmd (Windows)

- **Mac**

>>**cd**: Cambia a otro directorio

>>**ls**: Muestra una lista de archivos y subdirectorios en un directorio



Terminal (Mac)

Argumentos (-Consola)

- ¿Qué hace este programa?

```
1 # argv contiene los argumentos entregados  
2 # por el usuario en la consola  
3 from sys import argv  
4 a = int(argv[1])  
5 b = int(argv[2])  
6 if b < a:  
7     t = b  
8     b = a  
9     a = t  
10 print(a)  
11 print(b)
```

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

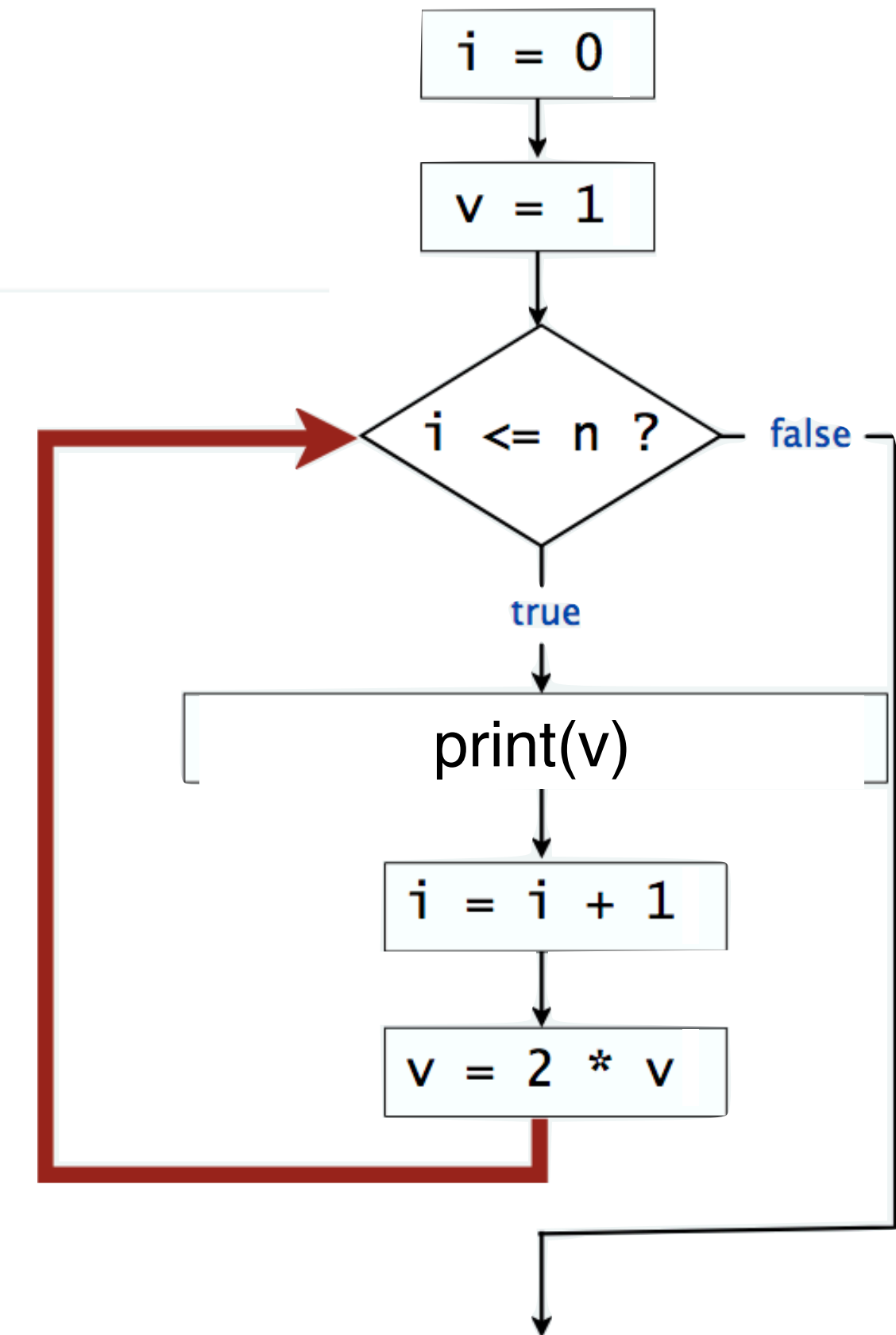
Ciclo while

- Ejecutar código mientras se cumple una condición:
 - Evaluar una expresión booleana
 - Si la expresión es True, ejecutar un bloque de código
 - **Repetir**

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

Expresión booleana

Imprime las potencias de dos desde 2^0 a 2^n




Ciclo **while**


Input

```
1 from sys import argv
2 n = int(argv[1])
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

```
$ python3 potencia2.py 7
1
2
4
8
16
32
64
128
```