udp UNIVERSIDAD DIEGO PORTALES

Introducción a Python

Introducción

2026

- Estructuras de control
  - Ciclos anidados

```
try:
    start = date(int(self.start_year.
                     self.months.index(se
                     int(self.start_day.ge

    end = date(int(self.end_year.get(
                     self.months.index(se
                     int(self.end_day.get
```

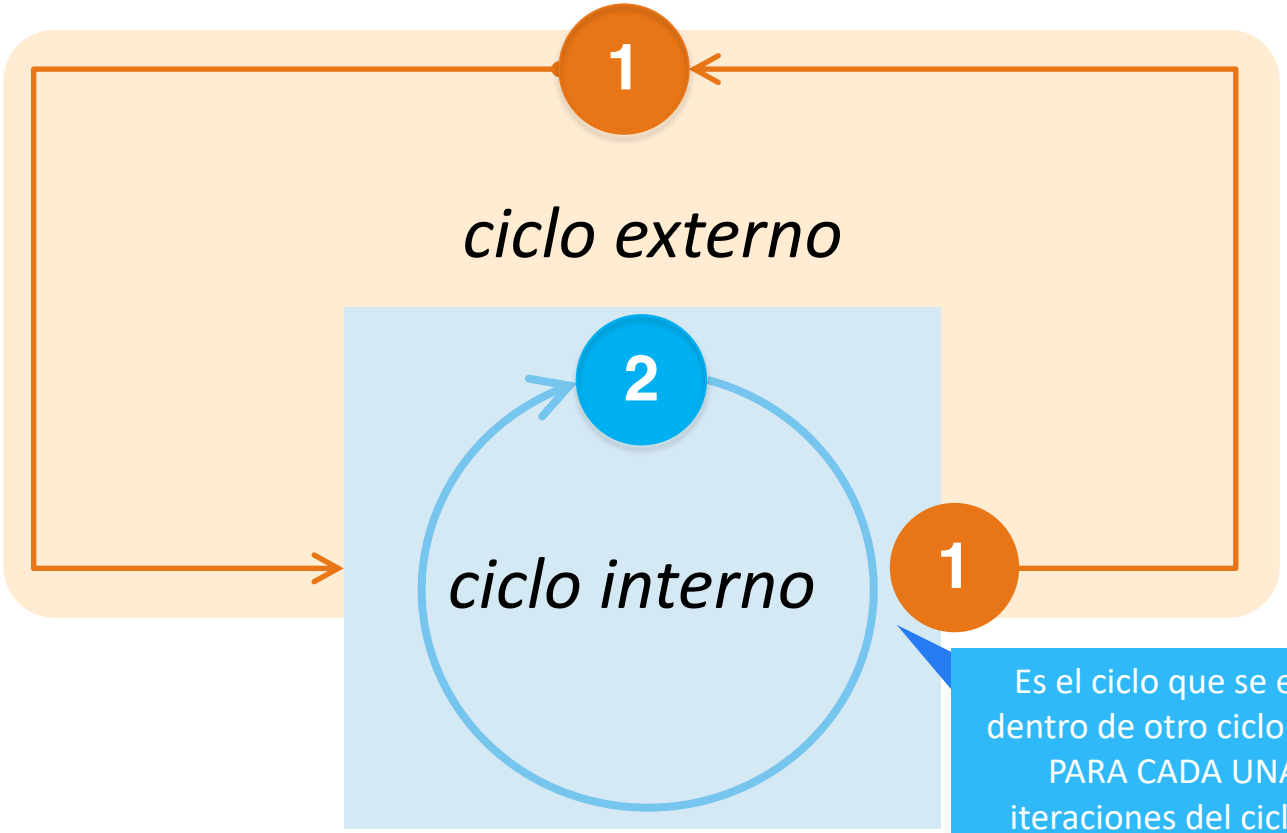
Cuando tenemos un ciclo, la operación se realiza según alguna condición de término

1

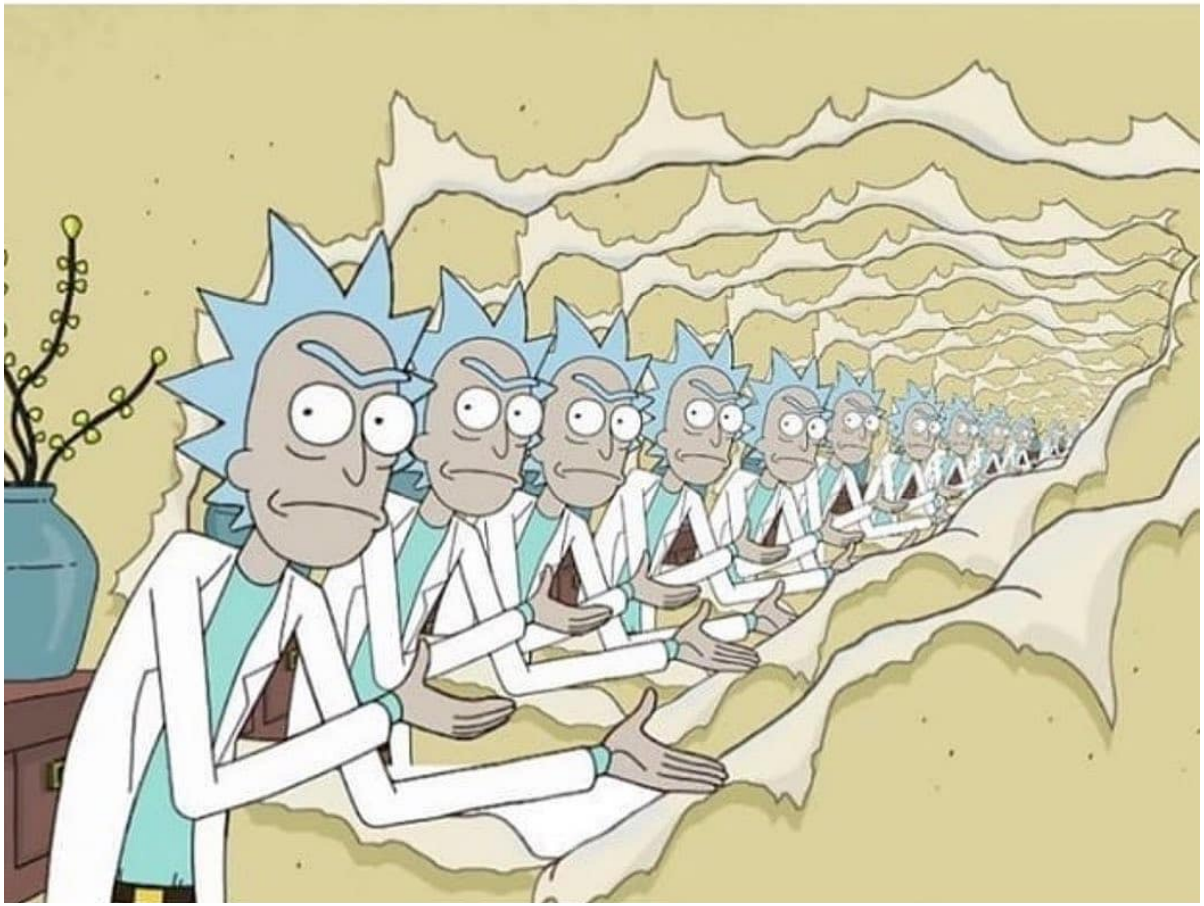
ciclo o loop

dentro del ciclo loop se encuentran las instrucciones que se repiten mientras dure el ciclo

Cuando hay (al menos) un ciclo dentro de otro ciclo, se habla de ciclos anidados.



When you forget to break out of the while loop



Ciclos anidados

Estructura:

```
ciclo externo (se ejecuta n veces):
    ciclo interno (se ejecuta m veces):
        instrucciones de ciclo interno
```

se ejecuta $n \times m$ veces, si son independientes

Un ciclo anidado

¿Qué aparece en pantalla al ejecutar este algoritmo?

```
i = 1
while i<5:
    j = 1
    while j<5:
        print(i,",",j)
        j = j+1
    i = i+1
```

Ciclo Interno

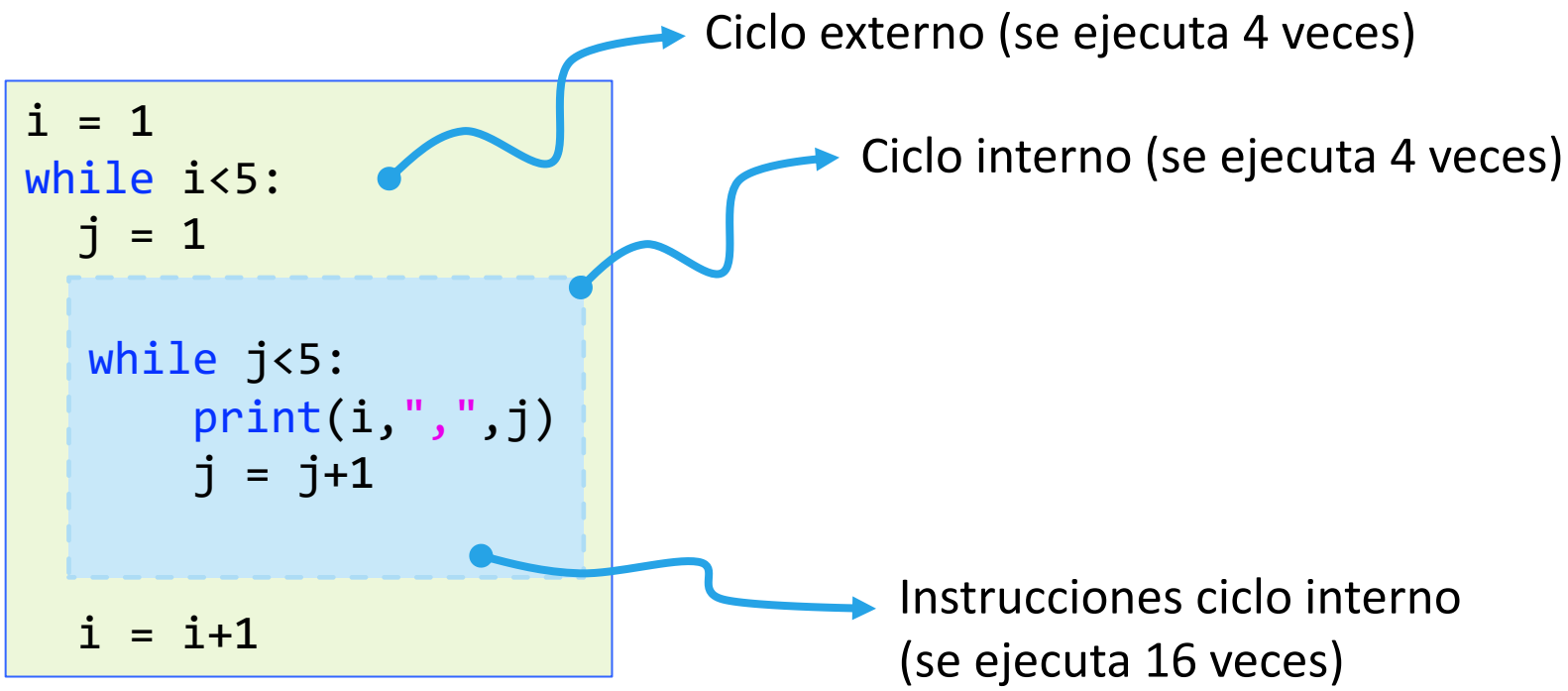
```
i = 1
while i<5:
  j = 1
  while j<5:
      print(i,",",j)
      j = j+1

  i = i+1
```

Ciclo interno (comienza con j=1)

```
while j<5:
    print(i,",",j)
    j = j+1
```

Solo una vez que terminan **todas las iteraciones del ciclo interno** se evalúa si se realiza una siguiente iteración del ciclo externo.

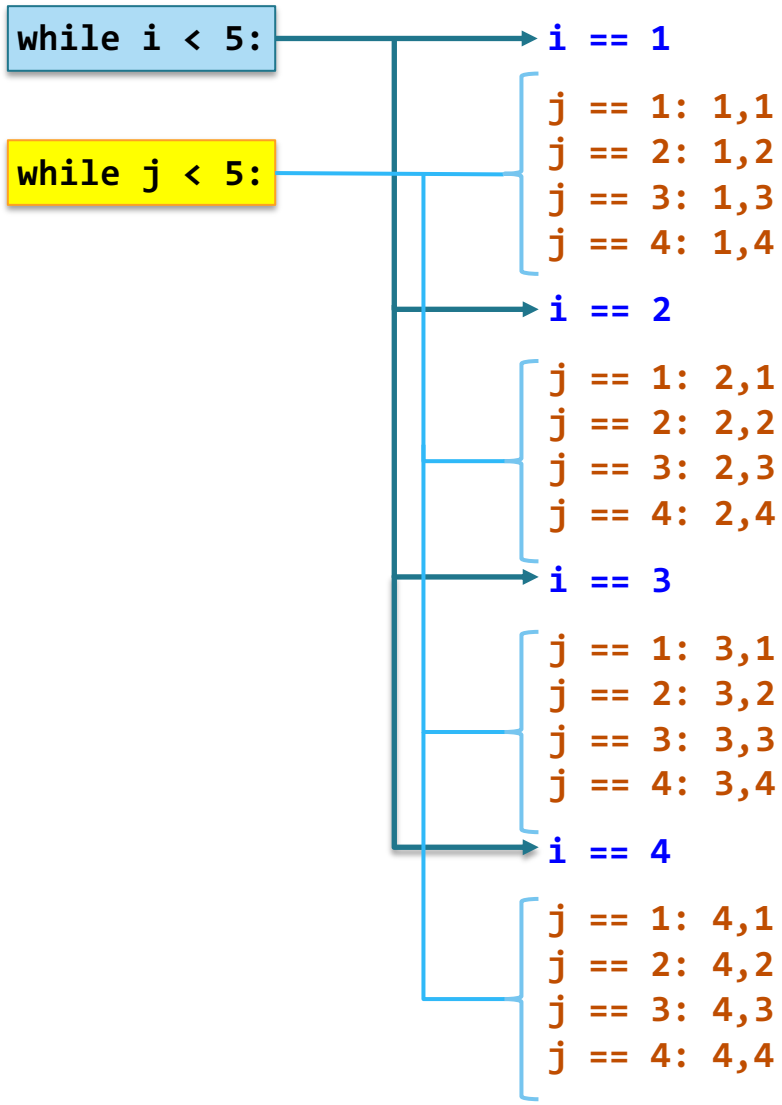
Un ciclo anidado




```
i = 1
while i<5:
  j = 1

  while j<5:
      print(i,",",j)
      j = j+1

  i = i+1
```

Un ciclo anidado

```
i = 1
while i<5:
    j = 1
    while j<5:
        print(i,",",j)
        j = j+1
    i = i+1
```



Estructura:

- ciclo externo (se ejecuta n veces):
  - ciclo interno (se ejecuta *F(n)* veces):
    instrucciones de ciclo interno

se ejecuta F(a)+F(b)+…+F(n) veces, ya que n depende del ciclo externo

se ejecuta n×m veces, si son independientes

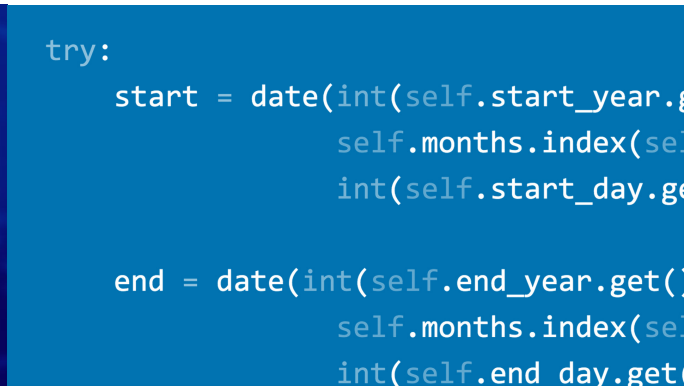
¿Qué aparece en pantalla al ejecutar este algoritmo?

```
i = 1
while i<5:
    j = 1
    while j<=i:
        print(i,",",j)
        j = j+1
    i = i+1
```

respuesta

```
1 , 1
2 , 1
2 , 2
3 , 1
3 , 2
3 , 3
4 , 1
4 , 2
4 , 3
4 , 4
```

- Estructuras de control
  - lista de números (range)
  - ciclo FOR
  - ciclos Anidados
  - Ejercicios



Para practicar

1) Crea un programa que despliegue el siguiente patrón de números:

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

El número de filas lo ingresa el usuario.

2) Crea un programa que despliegue el siguiente patrón de números:

1
2 3
4 5 6
7 8 9 10

El número de filas lo ingresa el usuario.

3) Crea un programa que despliegue el valor de la siguiente expresión

$1^1+(2^1+2^2)+(3^1+3^2+3^3)+\ldots+(N^1+N^2+\ldots+N^N)$, el valor N se solicita al usuario.

Por ejemplo, si N = 3 se debería desplegar 46.

4) Escribe el código que permite resolver la siguiente serie **1!+ 2! + 3! + …+ N!**, el valor de N se solicita al usuario. Por ejemplo, si N = 3 se debería desplegar 9.

Ejercicios

(Ruteo) Escribe lo que aparece en pantalla al ejecutar el siguiente programa en Python:

```
for i in range(0,5):
    for j in range(0,i):
        print(i,",",j)
```

Escribe un programa en Python que te permita generar la mitad derecha de un triángulo. El usuario ingresa el alto. Por ejemplo:

```
Ingrese alto piramide:
5
#
##
###
####
#####
```

(Ruteo) ¿Qué aparece en pantalla al ejecutar este programa con datos 3 y 4? ¿y con datos 5 y 3?

```
alto=int(input("Alto del rectangulo:"))
ancho=int(input("Ancho del rectangulo:"))
for i in range(1, alto+1):
    for j in range(1, ancho+1):
        print("#", end="")
    print(" ")
```

Escribe un programa en Python que despliegue el avance de los primeros 3 minutos de un cronómetro minutero. Es decir, que aparezca en pantalla:

```
0:00
0:01
0:02
0:03
….
0:59
1:00
1:01
1:02
…
1:59
2:00
2:01
…
2:59
3:00
```