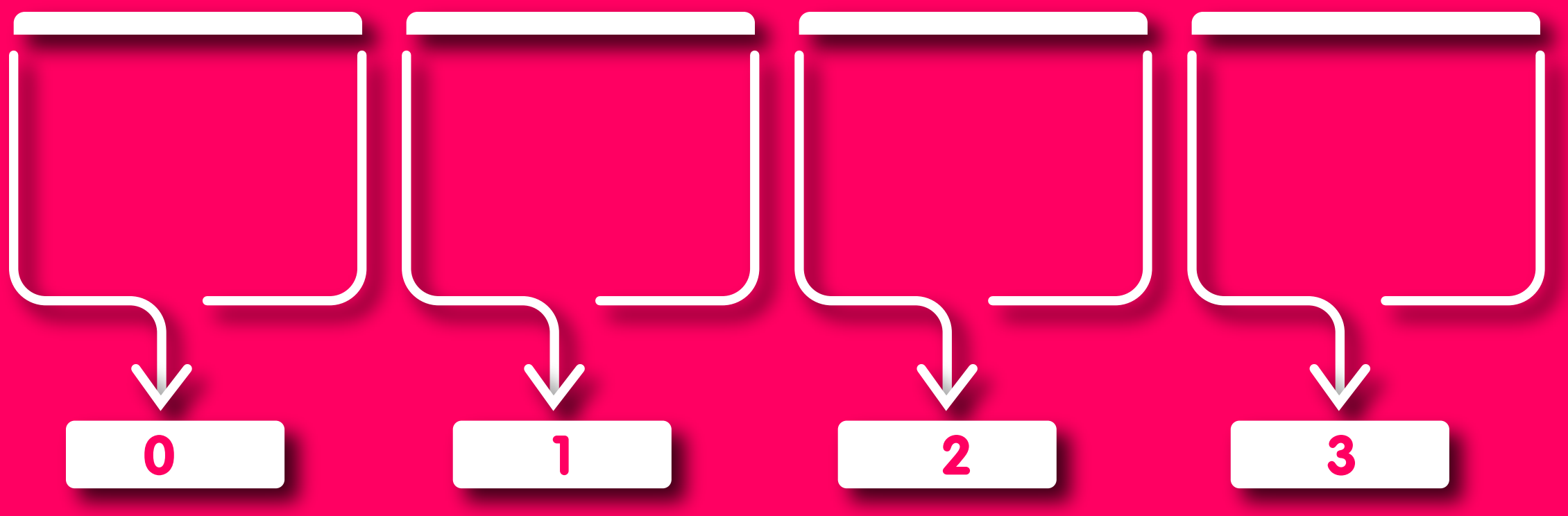
El futuro digital es de todos

MinTIC



Listas

Patrones de recorrido

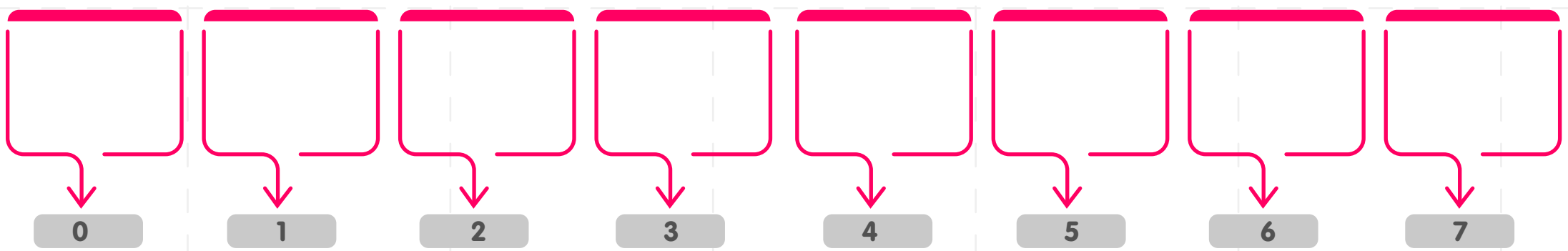
Universidad de Caldas

Hablar de manejo de listas sin hablar de ciclos, es prácticamente imposible ya que están directamente relacionados. Hoy vamos a ver algunos patrones de recorrido de listas que están muy relacionados con los patrones que ya vimos de ciclos, Bienvenidos.

Recorrer listas, es decir, pasar por cada uno de sus elementos, para consultarlos, cambiarlos, para hacer algún cálculo con ellos es fundamental, es una operación que debemos aprender de memoria porque sin saber recorrer los elementos de la lista no vamos a poder hacer ninguna operación.

Al igual que con los ciclos, existen una serie de patrones, muy básicos incluso, que pueden ayudar a la manipulación inicial.

El primero se llama el **iterador**, este es un patrón que permite recorrer una lista, además proporciona un acceso secuencial a los elementos de esta; es decir que va recorriendo uno a uno los elementos en secuencia hasta llegar al último elemento de la lista, así:



- Contar elementos que cumplen una condición.
- Realizar una operación que involucra a todos los elementos de la lista.
- Buscar un elemento que cumple una condición única con respecto a los otros.

Todos estos elementos y estrategias que involucran el manejo de todos los elementos es un patrón de iteración.

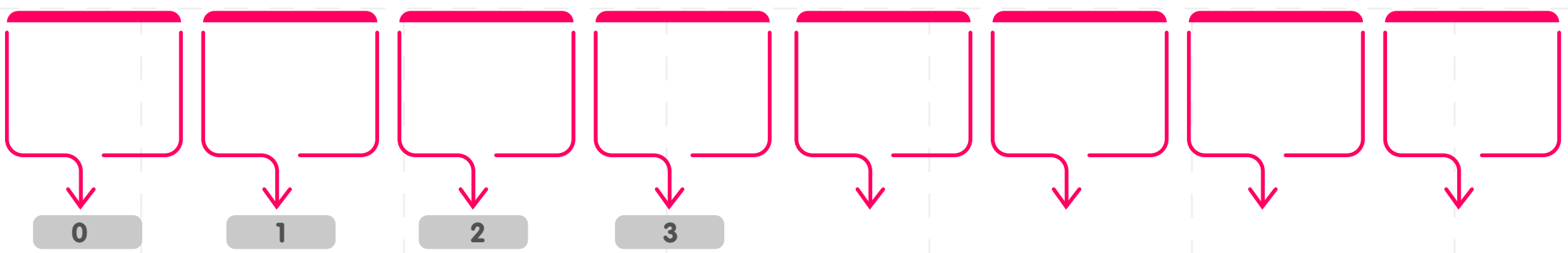
Veamos un ejemplo en el cual tenemos una lista de notas y se quiere calcular cuántos han aprobado el curso.

```
def calcular_aprobados (lista_notas)
    Cantidad_aprobados= 0
    for nota in lista_notas: # recorrido de la lista
        if nota >= 3.0 :
        cantidad_aprobados += 1
    return cantidad_aprobados
```

Aquí ya hay elementos que debemos recordar de patrones anteriores, en este caso para contar se requiere iniciar con un valor de cero (0), es importante revisar con cuidado el uso del FOR que es un poco diferente, (se lee de la siguiente manera): para cada nota en la lista de notas, si la notas es mayor o igual a 3.0 aumenta la cantidad de aprobados y retorna la cantidad de aprobados.

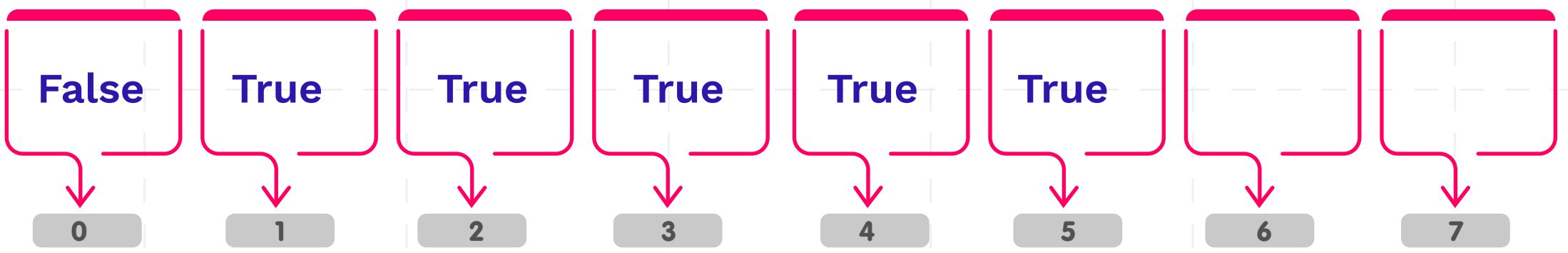
Aspecto importante:
Hacer lectura del **for** ya que ahí es donde se está haciendo el patrón total del recorrido de la lista.

Es posible realizar un recorrido parcial, esto pasa por ejemplo cuando el programa busca el primer elemento que cumple una condición, inmediatamente lo encuentra ya no tiene que buscarlos más.



- Buscar el primer elemento que cumpla una condición
- La lista satisface una condición
- Puede recorrer toda la lista en algunos casos

En el siguiente ejemplo vamos a analizar la iteración parcial contando votos para saber en un grupo de estudiantes cuántos de ellos van a madrugar a las 5:00 am a clase.



Mayoria absoluta = 5 votos
Votos a favor = 5

En el ejemplo se cuenta con 8 estudiantes, por lo tanto la mayoría absoluta es 5, la primera persona vota por no ir a esa hora y las siguientes 5 votan en forma afirmativa, así que no necesitamos los dos votos restantes.

Una primera versión del primer iterador parcial, es el de utilizar una bandera o un centinela, que básicamente es una variable que sirve de control para ver si se ingresa o no a un sitio, entonces veámoslo como una bandera que mientras esté arriba es porque se puede ingresar al ciclo y mientras esté abajo es porque ya termina el ciclo, básicamente cuando está arriba la bandera es verdadera y cuando está abajo es falsa.

```
def aprobar_propuesta (lista_notas)
    mayoria_absoluta= len(lista_votos)// 2 + 1
    indice= 0
    centinela = True
    votos_favor= 0
    while centinela:
        if lista_votos[indice] :
          votos_favor+= 1
    indice +=1
    centinel= indice < len(lista_votos) and votos_favor < mayoria_absoluta
    if votos_favor >= mayoria_absoluta:
        return True
    else:
        return False
```

El primer patrón de recorrido parcial, permite contar el número de votos de la propuesta hasta que se alcance la mayoría absoluta o hasta que le pregunte a todas las personas.

```
def aprobar_propuesta (lista_notas)
    mayoria_absoluta= len(lista_votos)// 2 + 1
    votos_favor= 0
    for voto in lista_votos:
        if voto:
            votos_favor+= 1
        if votos_favor >= mayoria_absoluta:
            return True
return False
```

El segundo patrón de recorrido parcial es **break**, romper el ciclo, generalmente se enseña que un ciclo no debe romperse con la instrucción **break**, esto pudo darse hace tiempo ya que los procesadores de antes eran demorados en recuperar el control de flujo y podría crear efectos con respecto a la ejecución, pero hoy en día no es cierto.

Lo que estamos haciendo acá es revisar el ciclo con cada elemento y si uno de ellos cumple la condición retorno a true se interrumpe el ciclo, si recorre todos los elementos y no se interrumpió es porque ninguno cumplió la condición y retorna a falso.

En resumen hemos visto los patrones de recorrido, patrones de sitios que son muy similares, se dispone de una forma específica de iterador total, también de un iterador parcial que no recorre toda la lista y podemos utilizar la bandera o centinela y el break ya con esto puedes avanzar significativamente en tu reto, al iniciar muchos de quienes leen esta revista hace 3 o 4 semanas no conocían ni siquiera que era python y ahora ya estamos haciendo programas.

Muchas gracias por su participación en esta semana , seguimos la próxima semana.


Mision
TIC2022

Universidad de Caldas