El futuro digital es de todos

MinTIC

Hechos QUE CONECTAN

Tutoría Magistral: Estructuras de control

Cuerpo de tutores 8 AM a 12 M

Jorge Yair Moreno Celis – Expositor
Dylan Yesid Villalba Roa- Expositor

Universidad Industrial de Santander

UIS

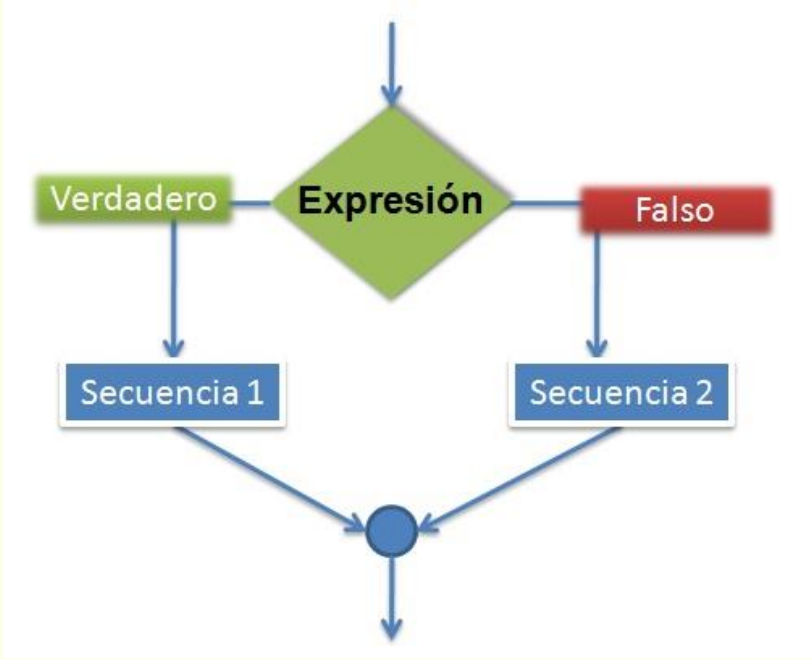
MisionTIC2022

Agenda

- Condicionales
- Variables de control
- Bucles
- Funciones
- Ejercicios

Estructuras de control

- **Condicionales**



Estructuras de control en Python

- **Condicional if**

Uno de los condicionales más comunes es el empleado por el enunciado if, el cuál presenta la siguiente estructura:

```
if Condición :
    #código
    #código
    #código
#código fuera del if
```

- **Condicional if - else**

La cláusula if-else le permite ejecutar un conjunto de enunciados si la comparación es verdadera y un conjunto de enunciados diferentes si la comparación es falsa.

```
if Condición :
    #código
    #código
else:
    #código
    #código
#código fuera del if/else
```

- **Condicional elif**

Cuando se anidan varios niveles de enunciados if/else, puede ser difícil determinar cuáles expresiones lógicas deben ser verdaderas (o falsas) con la finalidad de ejecutar cada conjunto de enunciados.

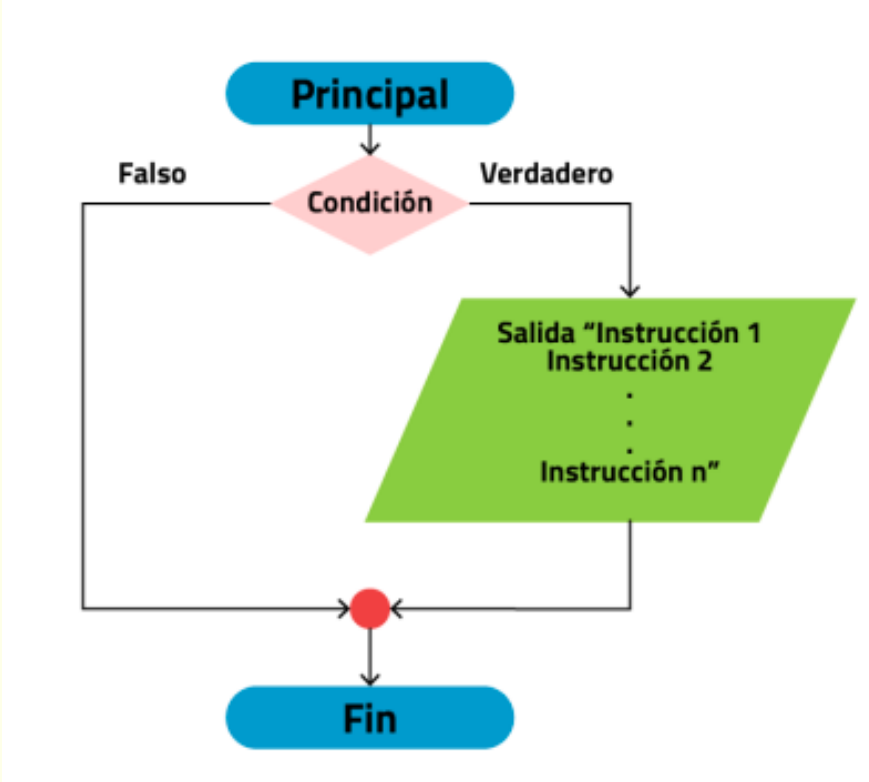
La función elif le permite comprobar criterios múltiples mientras se mantiene el código fácil de leer.

```
if Condición_1:
    #código
    #código
elif Condición_2:
    #código
    #código
else:
    #código
    #código
#código fuera del if/elif/else
```

Ejercicios de estructuras de control en Python

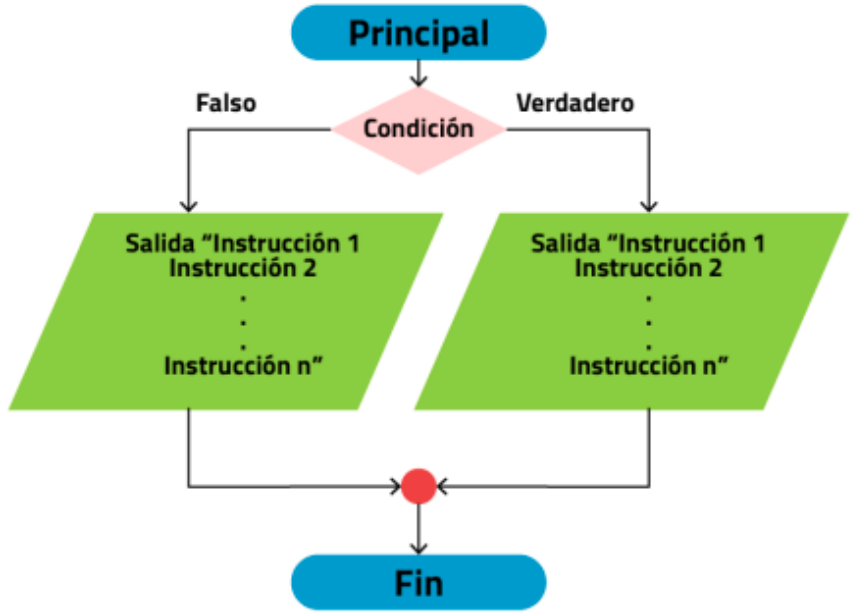
Realizar un algoritmo que determine si un número es un entero positivo:

- Cree una variable llamada número e ingrese el valor por consola
- Estructure la condición if
- Imprima el resultado



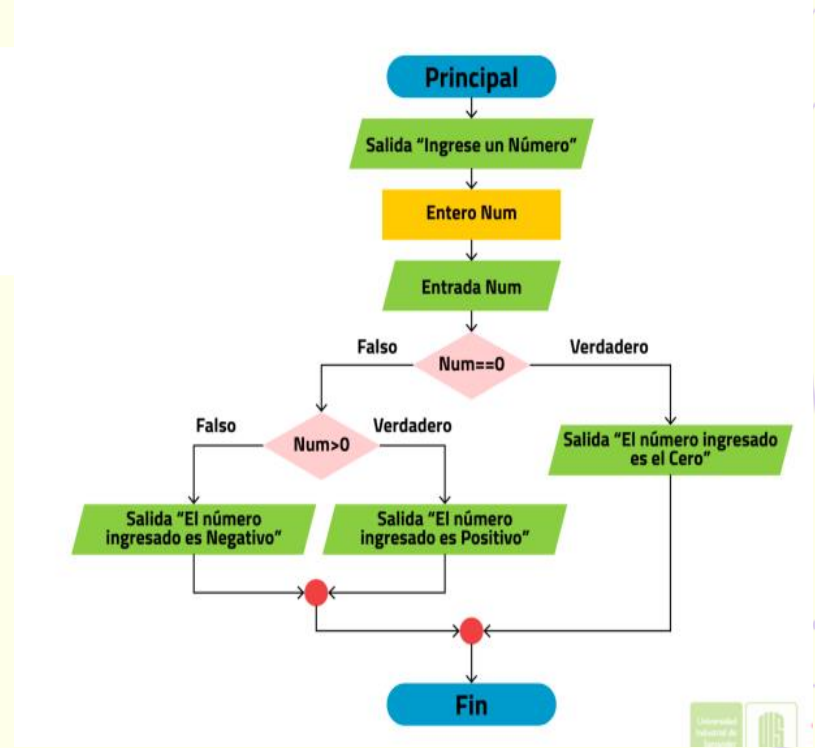
Realizar un algoritmo que determine cual de dos números ingresados es el mayor y cual es el menor:

- Cree una variable llamada numero_1 y numero_2 e ingrese el valor por consola
- Estructure la condición if-else
- Imprima el resultado



Realizar un algoritmo que determine cual de dos números ingresados es el mayor, cual es el menor o si son iguales:

- Cree una variable llamada numero_1 y numero_2 e ingrese el valor por consola
- Estructure la condición elif
- Imprima el resultado



Variables de control

- Banderas
- Contadores
- Acumuladores

Banderas

son variables, normalmente de tipo lógicas (boolean), conservan un estado hasta que un evento requiera cambiarlo y ejecutar otra funcionalidad.

Es importante resaltar que las banderas no siempre son de tipo bool, pueden ser de cualquier tipo, siempre y cuando tenga un significado su estado.

Ejemplo: Se requiere realizar una multiplicación, pero además indicarle al usuario por pantalla cuando la multiplicación ya se ejecutó.

Acumuladores

Un acumulador es una variable que se utiliza para sumar valores. Al igual que los contadores, se utiliza normalmente dentro de un ciclo pero cambiamos su valor sumándole una variable, es decir, no siempre se le suma la misma cantidad.

Ejemplo: Una lista de compras

Contadores

Un contador es una variable que se utiliza para contar algo. Normalmente usamos un contador dentro de un ciclo y cambiamos su valor sumándole o restándole una constante, es decir, siempre se le suma o resta la misma cantidad. El caso más utilizado es incrementar la variable en uno.

Cont=0

Cont=Cont+1

Ejemplo: Una variable contador que vaya de cero a 100

Tipos de Bucles

Determinados (Bucle For)

- Se ejecutan un número determinado de veces.
- Se sabe a priori cuántas veces se va a ejecutar el código del interior del bucle

Indeterminados (Bucle While)

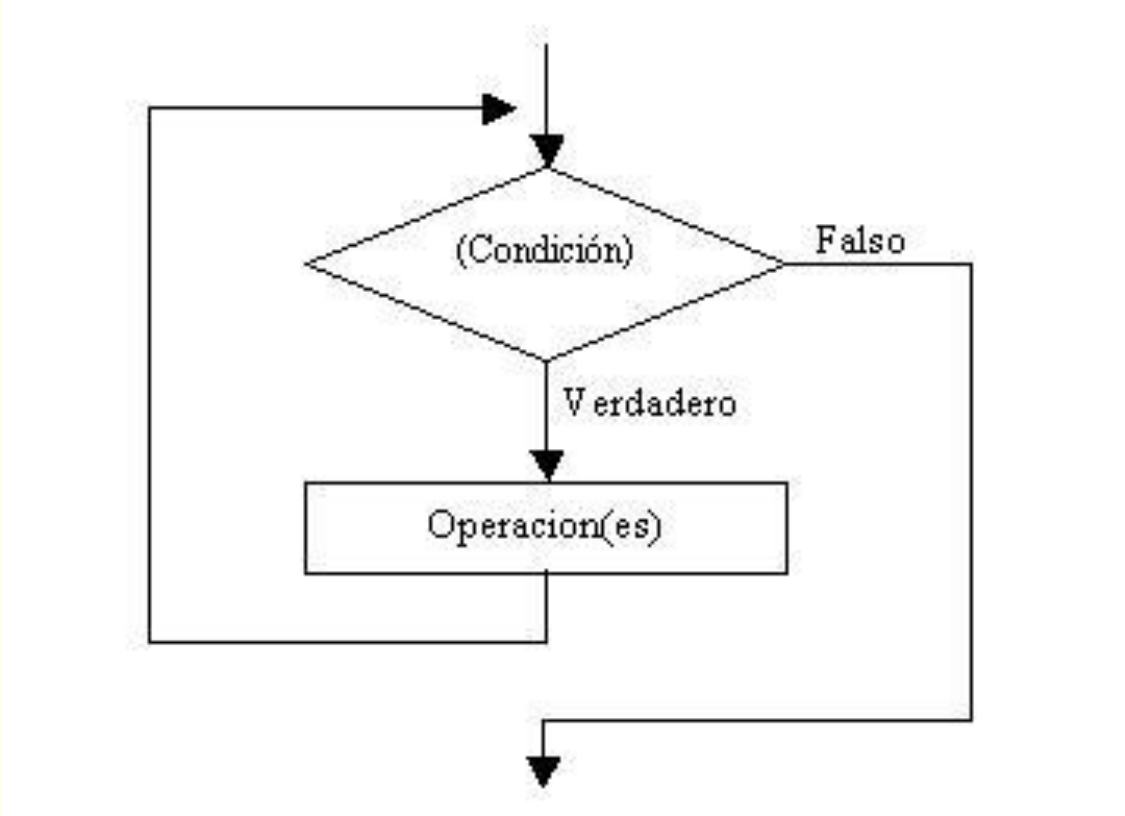
- Se ejecutan un número indeterminado de veces
- No se sabe a priori cuántas veces se va a ejecutar el código del interior del bucle
- El número de veces que se ejecutará dependerá de las circunstancias durante la ejecución del programa

Ciclos con while

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

```
while (condición1):
        …
        Bloque de instrucciones
        …
Código fuera del while
```

Estructura repetitiva ciclo while



Ciclos con for

un ciclo for en Python es una estructura iterativa para ejecutar un mismo segmento de código una cantidad de veces deseada y conocida. Se necesita conocer previamente un valor de inicio, un tamaño de paso y un valor final para el ciclo.

```
for indice in elemento a rrecorrer:
        Cuerpo del bucle
#el elemento a rrecorrer puede ser una lista, una cadena de texto, una tupla,etc.
```

Ciclos con for con range

```
for g in range(valor_inicial,valor_final,incremento):
        .....
        Bloque de instrucciones
        ......
Código fuera del ciclo
```

Ejemplo:

```
for i in range(0,100+1,5):  # en este caso la función va de 0 a 99 que son 100 elementos
        print(i)
```

Como puedes ver, la magia está en la función *range()* de Python, pues con esta definimos el inicio, el final y el incremento.

Funciones en python

Una función es un bloque de código que tiene asociado un nombre y que recibe o no una serie de argumentos como entrada, para luego seguir una serie de instrucciones que tienen como finalidad llevar a cabo una tarea específica y retornar un valor.

En términos técnicos, Una **función es un bloque de código que tiene asociado un nombre**, de manera que **cada vez que se quiera ejecutar el bloque de código basta con invocar el nombre de la función**.

Sentencia Def Funciones

Para declarar una función se utiliza la siguiente sintaxis:

```
def <nombre-función>(<parámetros>):
        bloque código
        return <objeto>
```

```
def nombre(parametros) :
    #código dentro la función
    #código dentro la función
    return  expresión
#código fuera de la función
```

Ejemplo de una función

```python
#Ejemplo de una función

def imprimir_pantalla():
    print("Hola Mundo")

imprimir_pantalla()


def imprimir_nombre(nombre):
    print("El nombre de la persona es:",nombre)

imprimir_nombre("Juan")

```

Funciones con parámetros

Una función puede recibir valores cuando se invoca a través de unas variables conocidas como parámetros o argumentos que se definen entre paréntesis en la declaración de la función.

En el cuerpo de la función **se pueden usar estos parámetros como si fuesen variables**.

```
14  # funciones con parametros o argumentos
15
16  def nombre_funcion(argumento):
17      return ('¡Bienvenido a python',argumento+'!')
18
19  print(nombre_funcion("Pedro"))
20
21  def funcion_suma(a,b):
22      suma=a+b
23      return suma
24
25  print(funcion_suma(5,6))
26
27
```

Funciones con retorno

Una función puede devolver una variable de cualquier tipo tras su invocación. Para ello la variable a devolver debe escribirse después de la palabra reservada return.

Si no se indica ningún objeto, la función no devolverá nada.

```
#Otras funciones con return

#La función es_par() devuelve True si un número es par y False en caso contrario:

def es_par(numero):
    if numero % 2 == 0:
        return True
    else:
        return False

print(es_par(4))
print(es_par(5))
```

Resumen: como definir una función

Las partes de una función son:
→**Nombre:** en este caso bienvenida
→**Argumentos**: que en este caso es nombre
→**Instrucciones:** en este caso solo una la línea 2
→**Retorno:** en este caso mensaje
→**Llamado:** en este caso línea 5

```
1   def bienvenida(nombre):
2       mensaje = '¡Bienvenido a Python', nombre + '!'
3       return mensaje
4
5   print(bienvenida("Pedro"))
```

Resumen: como llamar una función

Como nos dimos cuenta las funciones son llamadas por su nombre seguido de un paréntesis donde puede haber o no argumentos. Y estas funciones pueden ser llamadas cuantas veces las necesitemos.

Esto lo vimos con ejemplos de funciones creadas por nosotros, pero como te lo imaginarás existen funciones que no creamos pero que existen de manera predefinida y podemos usar, y tu ya has usado algunas.

```
52  # funciones por default en python
53  print()
54  input()
55  max() #Devuelve el valor maximo de un arreglo
56  min() #Devuelve el valor minimo de un arreglo
57
```

Resumen: otros tipos de funciones por default

Estas funciones predefinidas las vemos como una caja de negra, es decir, llamamos las funciones obtenemos su resultado, pero no sabemos cómo implementa el código cada una.

```
52   # funciones por default en python
53   print()
54   input()
55   max() #Devuelve el valor maximo de un arreglo
56   min() #Devuelve el valor minimo de un arreglo
57
```

Una lista de las funciones predefinidas con su descripción y uso puede encontrarse aquí: https://docs.python.org/es/3/library/functions.html

Ejercicios

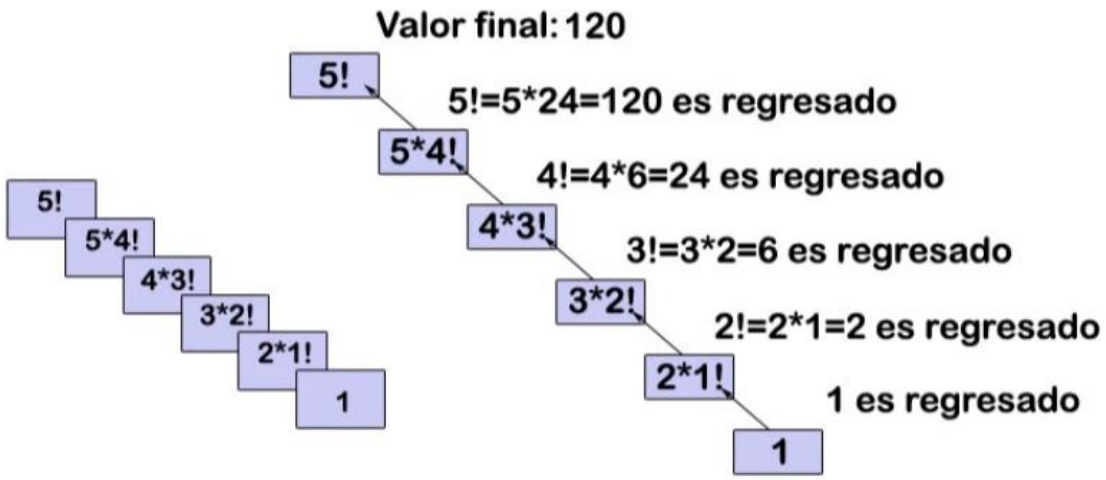
1. Escribir una función que calcule el total de una factura tras aplicarle el IVA. La función debe recibir la cantidad sin IVA y el porcentaje de IVA a aplicar, y devolver el total de la factura. Si se invoca la función sin pasarle el porcentaje de IVA, deberá aplicar un 19%

2. Escribir una función que calcule el área de un círculo y otra que calcule el volumen de un cilindro usando la primera función. Nota: A= π*r2

3. Escribir una función que calcule el factorial de un número.

Planteamiento ejercicio 3

```
# 5! = 5 * 4 * 3 * 2 * 1
# 5! = 5 * 4 * 3 * 2
# 5! = 5 * 4 * 6
# 5! = 5 * 24
# 5! = 120
```



Gracias por su atención

Universidad Industrial de Santander

MisiónTIC2022