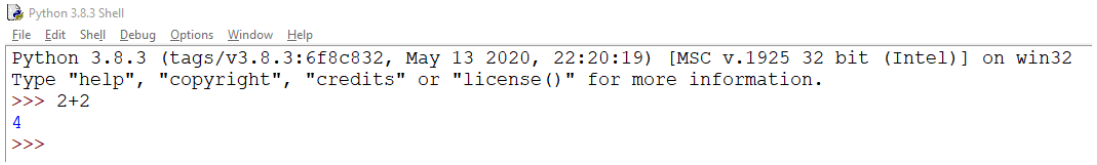
Programación Básica con Python

Interacción con la Shell de Python



Introducción a la Shell Interactiva

Capacidad para realizar operaciones directamente

Uso de expresiones como instrucciones básicas

Concepto de Expresión

Combinación de valores y operadores

Reducción a un valor único

Uso de la Shell Interactiva

Acepta una instrucción a la vez

Permite instrucciones complejas pero individuales

Escribiendo tu Primer Programa en Python

- Saludo Inicial en Python
  - El código comienza con un saludo '¡Hola mundo!'
- Interacción con el Usuario
  - Pide el nombre y lo utiliza para saludar
  - Consulta la edad y calcula la edad futura
- Conversión de Tipos de Datos
  - Convierte la edad a entero para sumar
  - Uso de 'str' para imprimir números

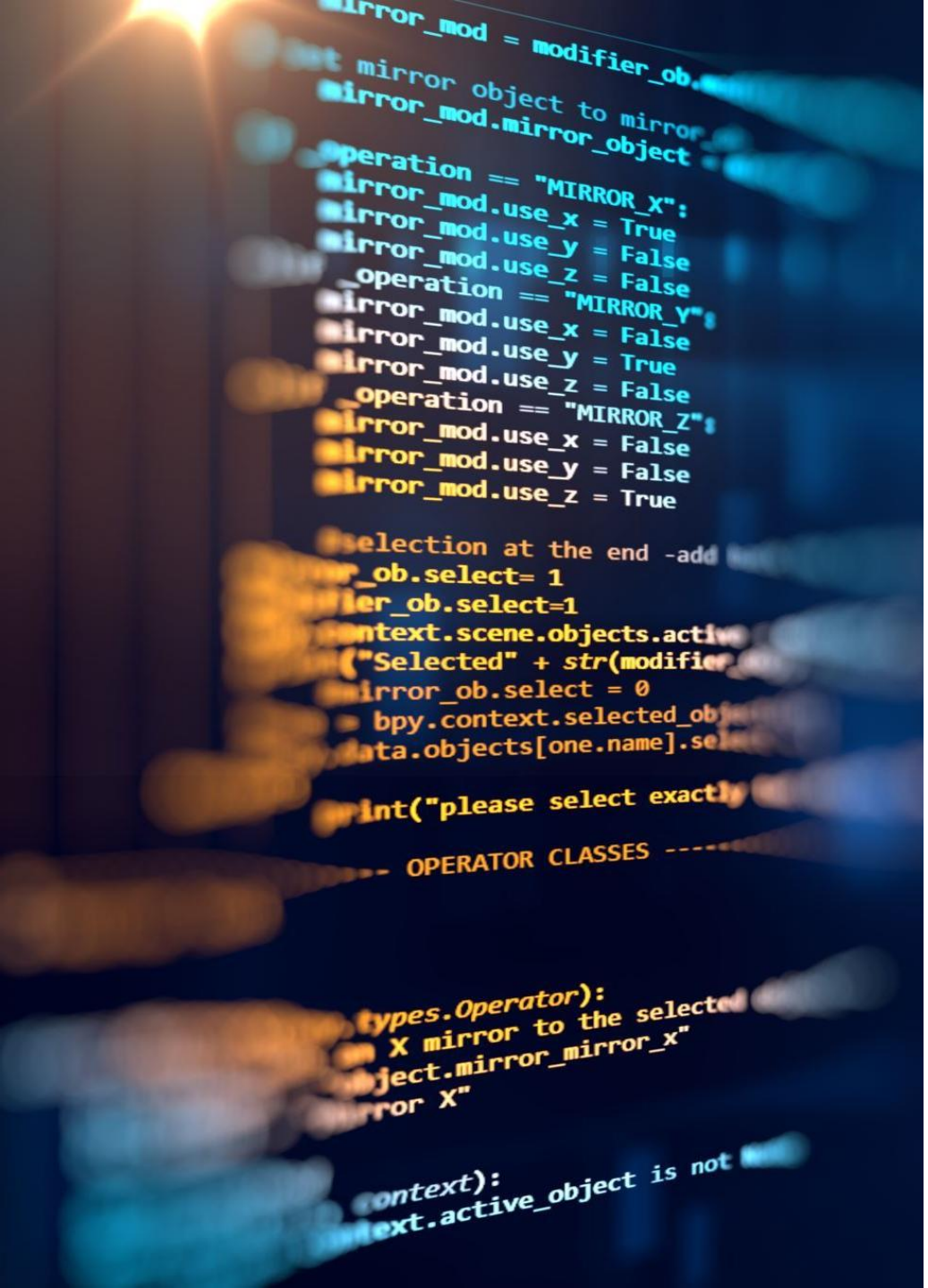
Escribiendo tu Primer Programa en Python

Hola mundo y entrada de datos

[1] ✓ 8s

```
# Este código fuente dice hola mundo y pregunta por tu nombre
print('¡Hola mundo!')
print('¿Cuál es tu nombre?')
mi_nombre = input()
print('Un gusto en conocerte, ' + mi_nombre)
print('¿Cuál es tu edad?')
mi_edad = input()
print('Tendrás ' + str(int(mi_edad) + 1) + ' años dentro de un año.')
```

```
¡Hola mundo!
¿Cuál es tu nombre?
Alumno
Un gusto en conocerte, Alumno
¿Cuál es tu edad?
18
Tendrás 19 años dentro de un año.
```



Variables y Objetos en Python: Variables y su Importancia

Almacenamiento de Valores con Variables

- Las variables permiten reutilizar valores en el código

Python: Un Mundo de Objetos

- Cada elemento en Python es considerado un objeto
- Los objetos poseen características específicas

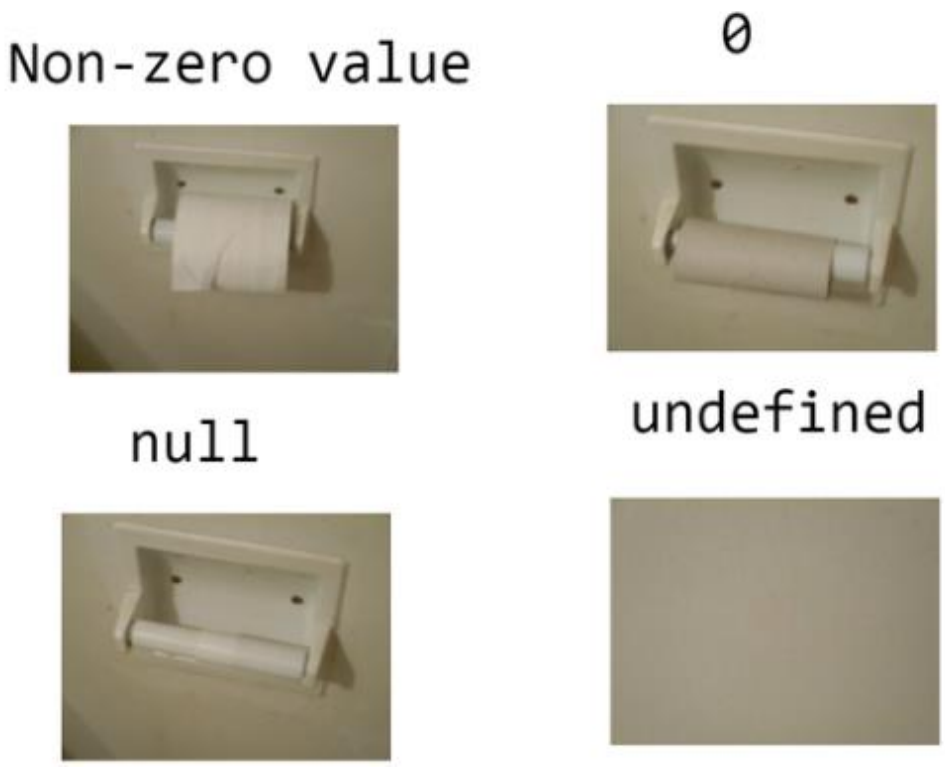
Variables y Objetos en Python: Tabla de Identidad, Tipo y Valor

Identidad	la dirección en memoria
Tipo	Número, secuencia, set, etc.
Valor	el dado por el usuario

Variables y Objetos en Python: Ejemplos de Variables

- Concepto de Variables
  - Elementos fundamentales en programación
  - Almacenan información para su uso
- Variables de Ejemplo
  - 'mi_nombre' y 'mi_edad' como contenedores
  - Recogen datos a través de 'input()'
- Importancia de Nombres Claros
  - Facilitan la comprensión del código
  - Mejoran la mantenibilidad

Variables y Objetos en Python: El Valor None



- Representación de Valor Nulo
  - 'None' se utiliza para indicar la ausencia de un valor.
  - Es un concepto clave en la programación para representar valores nulos.
- Diferenciación de Otros Valores
  - Distinto de cero, que representa una cantidad numérica.
  - No es lo mismo que falso, que indica una condición no verdadera.
  - Diferente de una cadena vacía, que es una secuencia de caracteres sin contenido.

Variables y Objetos en Python: Tipos de Números en Python

Números Enteros

- Sin parte fraccional

Números de Punto Flotante

- Con parte fraccional o punto decimal

Números Complejos

- Con parte real e imaginaria (a + bj)

Secuencias o datos estructurados

- Se explorarán en temas posteriores

Keywords (palabras clave)

and	del	from	not	while	as
elif	gloal	else	if	pass	Yield
break	except	import	class	raise	continue
finally	return	def	for	try	Entre otras

Operadores Aritméticos

+ : suma	• 1 + 2 = 3
- : resta	• 1 – 2 = - 1
* : multiplicación	• 1 * 2 = 2
/ : división	• 10 / 2 = 5
% : módulo	• 5 % 2 = 1
** : exponente	• 5 ** 2 = 25
// : división de piso	• 9 // 2 = 4 11 // 3 = 3

Operadores de asignación

= : igualación	• x = 2 x = 2+5
+= : Suma y	• x+=2 equivale a x = x + 2
-= : resta y	• x-=2 equivale a x = x - 2
*= : multiplica y	• x*=2 equivale a x = x*2
/= : divide y	• x/= 2 equivale a x = x / 2
%= : modulo y	• x%= 2 equivale a x = x % 2
**= : exponente y	• x**=2 equivale a x = x ** 2
// = : división de piso y	• x // = 2 equivale a x = x // 2

Operadores de relación

Jerarquía de operadores

**

* / % //

+ -

<= < > >=

== !=

= %= /= //= -= += *= **=

Operadores en Python: Operadores Lógicos

Operador 'And'	Representado por el símbolo '&' Usado para verificar que dos condiciones sean verdaderas
Operador 'Or'	Representado por el símbolo '|' Usado para verificar que al menos una de las condiciones sea verdadera

Condicionales

- Declaraciones “if”
- Declaraciones “if... else”
- Declaraciones anidadas

Declaraciones if

La sintaxis de la declaración if es la siguiente

if condición:

    Instrucción

    Instrucción

if (condición):

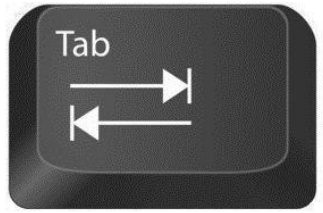
    Instrucción (es)

```
edad = int ( input("Hola, escribe tu edad \n") )


if (edad >= 18):
    print ("Ya eres mayor de edad")
    print ("Naciste en el año: " + str(2024-edad) )
```

Ejemplo de if

Ejemplo de if

```
edad = int ( input("Hola, escribe tu edad \n") )


if (edad >= 18):
    print ("Ya eres mayor de edad")
    print ("Naciste en el año: " + str(2024-edad) )
```



La indentación es importante.
Se hace con la tecla Tab

Las instrucciones dentro de este bloque, son las que se ejecutan si la condición es verdadera.

Estructuras Condicionales: Declaraciones if...elif

Declaraciones condicionales en Python

- La declaración 'if' verifica una condición y ejecuta un bloque de código si es verdadera.
- La declaración 'elif' permite verificar múltiples condiciones después de un 'if'.
- El código indentado se ejecuta solo si la condición correspondiente es verdadera.
- Si todas las condiciones son falsas, no se ejecuta ninguna acción.
- El indentado es crucial para definir bloques de código.

Ejemplo de if - elif

```
edad = int ( input("Hola, escribe tu edad \n") )


if edad >= 18:
    print ("Ya eres mayor de edad")
    print ("Naciste en el año: " + str(2024-edad) )
elif (edad < 18 and edad > 0):
    print ("Es menor de edad")
```

Estructuras Condicionales: Declaraciones if...elif...else

- Declaraciones Condicionales en Python
  - if para condiciones que son verdaderas
  - elif para condiciones adicionales
  - else para condiciones que son falsas
- Ejemplo Práctico
  - Verificación de la mayoría de edad
  - Calculo del año de nacimiento
- Manejo de Edades Desconocidas
  - Uso de else cuando if y elif son falsas
  - Importancia de cubrir todos los casos posibles

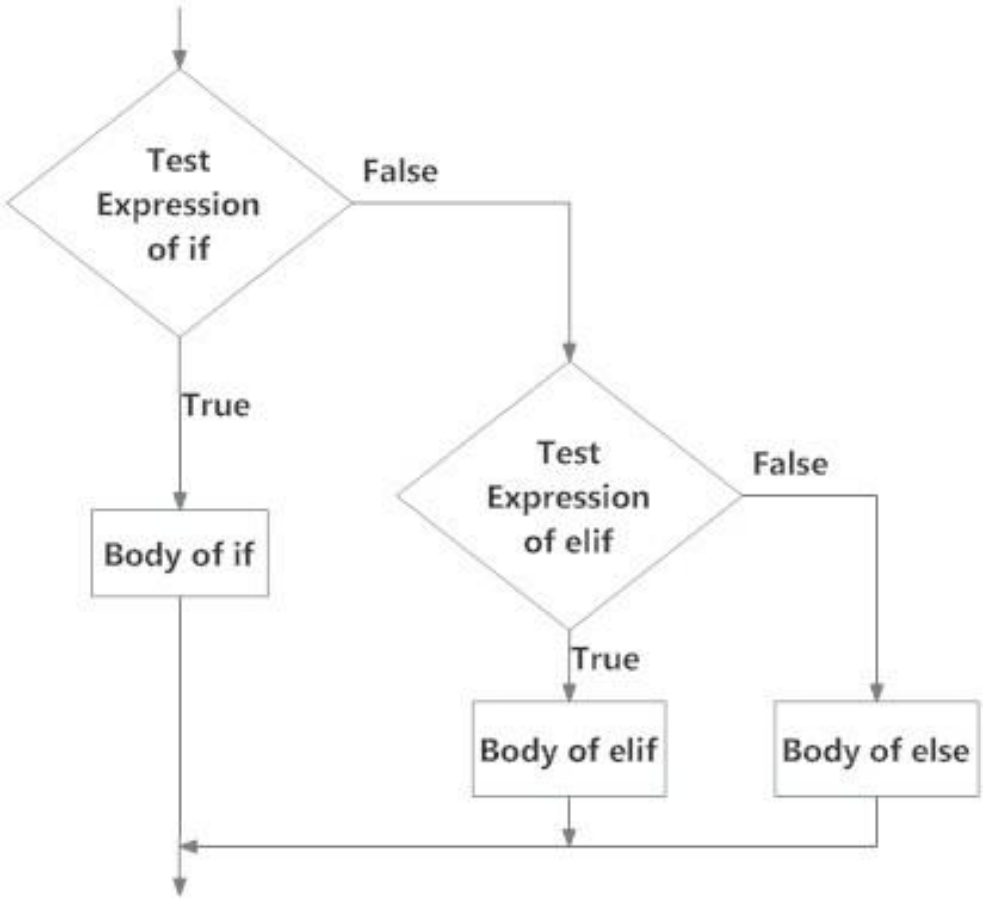


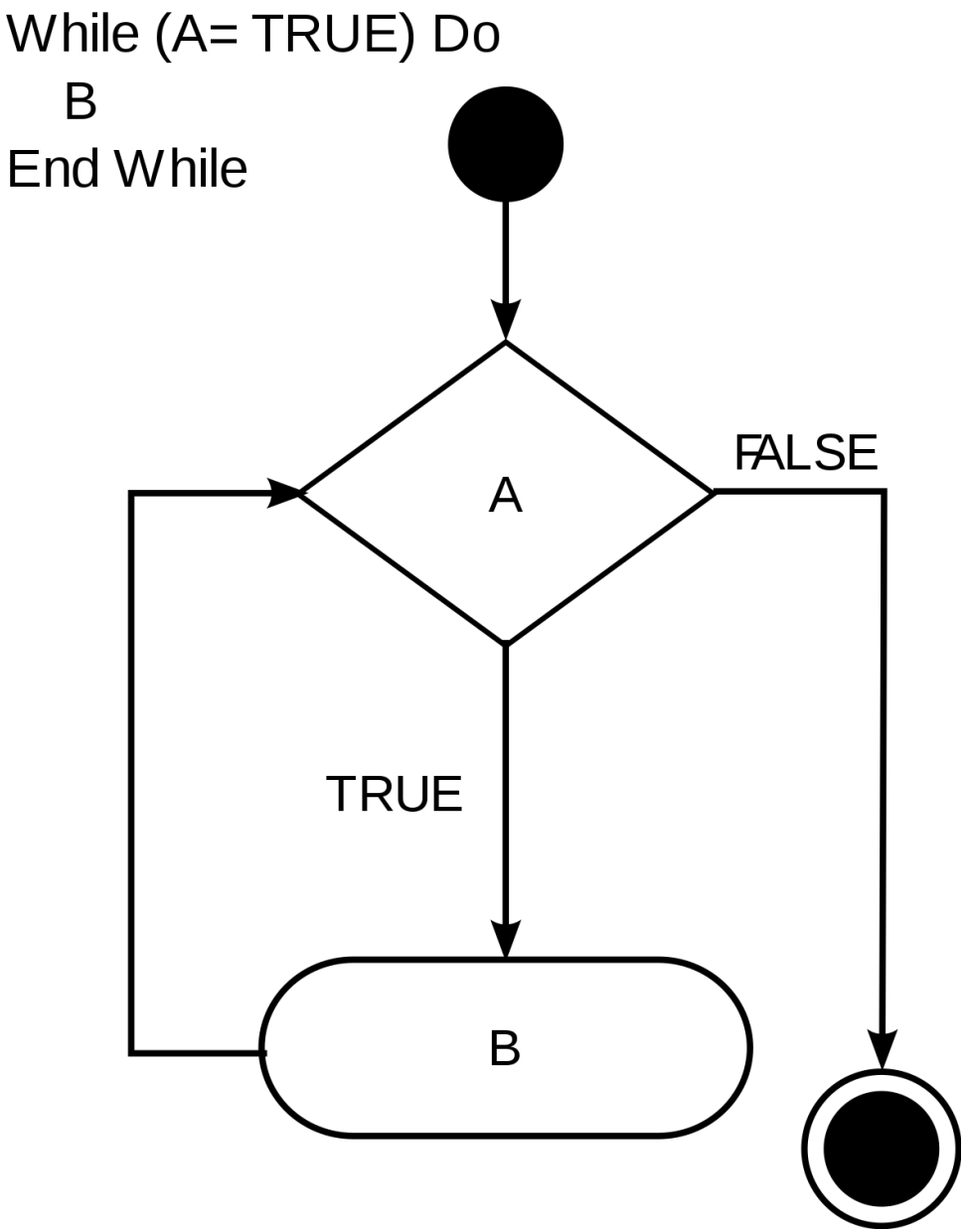
Fig: Operation of if...elif...else statement

Ejemplo if – elif - else

```
edad = int ( input("Hola, escribe tu edad \n") )


if edad >= 18:
    print ("Ya eres mayor de edad")
    print ("Naciste en el año: " + str(2020-edad) )
elif (edad < 18 and edad > 0):
    print ("Es menor de edad")
else:
    print ("Es una edad desconocida")
```

Ciclos y Bucles en Python: Ciclos While

- Sintaxis de la Instrucción 'while'
  - Se inicia con la condición a evaluar
  - Seguido por un bloque de código a ejecutar
- Ejemplo Práctico
  - Variable 'i' comienza en 0
  - Se imprime 'i' mientras sea ≤ 20
  - Incremento de 'i' en 2 en cada iteración
- Finalización del Ciclo
  - El ciclo termina cuando la condición es falsa



Estructuras de control - while

Sintaxis:

while condición:

→ Instrucción 1

→ Instrucción 2

→ ...

→ Instrucción n

[línea vacía]

Notas:

No hay i++, debe hacerse siempre i=i+1

```
In [17]: i=0
         while i<len(palabra):
             print("caracter",i,palabra[i])
             i=i+1
```

```
caracter 0 m
caracter 1 o
caracter 2 n
caracter 3 e
caracter 4 t
caracter 5 i
caracter 6 z
caracter 7 a
caracter 8 c
caracter 9 i
caracter 10 o
caracter 11 n
```

```
In [ ]:
```

Ciclos y Bucles en Python: Ciclos For

- Sintaxis de la Instrucción 'for'
  - Se inicia con 'for' seguido de una condición y dos puntos
  - Se ejecuta un bloque de código para cada elemento que cumple la condición
- Ejemplo con Números
  - Se crea una lista de números del 0 al 10
  - Para cada número en la lista, se imprime su doble

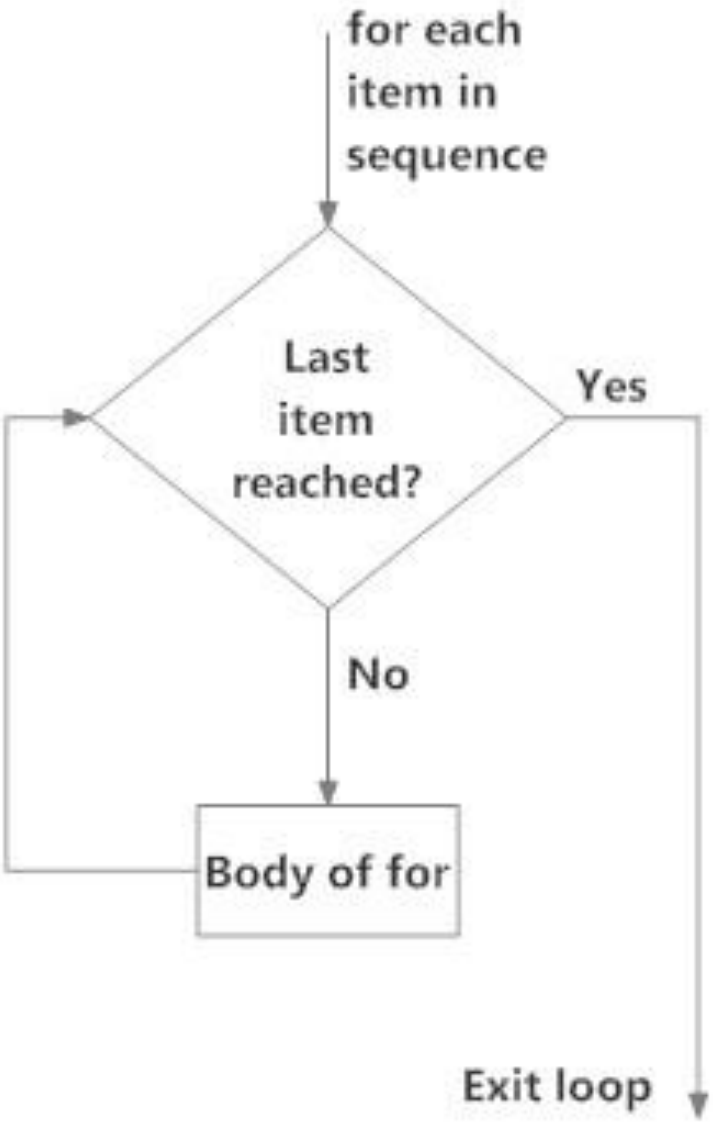


Fig: operation of for loop

Estructuras de control - for

Sintaxis:

for i in range(x)

→ Instrucción 1

→ Instrucción 2

→ ...

→ Instrucción n

[línea vacía]

Notas:

Los rangos son exclusivos en el límite superior:

[0,x)

Se pueden definir incrementos:

range(0,10,2)

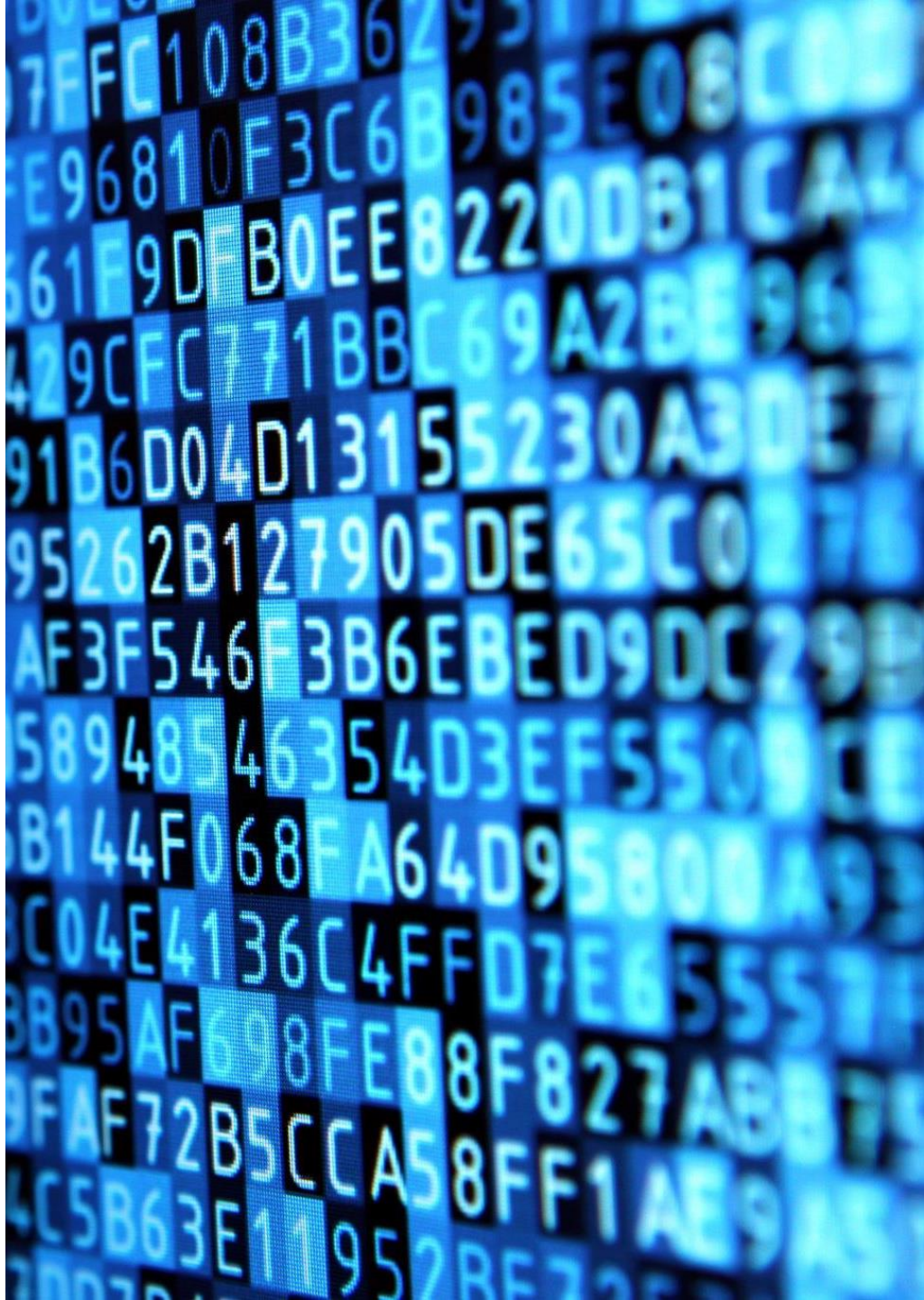
equivale a: 0, 2, 4, 6, 8

```
In [25]: for i in range(len(palabra)):
             print("caracter",i,palabra[i])
```

```
caracter 0 m
caracter 1 o
caracter 2 n
caracter 3 e
caracter 4 t
caracter 5 i
caracter 6 z
caracter 7 a
caracter 8 c
caracter 9 i
caracter 10 ó
caracter 11 n
```

Trabajando con Listas en Python: Sintaxis y Creación de Listas

- Definición de Listas
  - mi_lista = [0,2,4,6,8]
  - otra_lista = [0,'hola',2,4,5,6]
  - lista_de_listas = [[0,2,4,6],[3,6,9,12]]



Acceso y Manipulación de Listas

Creación de Listas

Se escriben entre corchetes ([]).

Acceso a Elementos

Se utiliza el nombre de la lista y el índice entre corchetes ([]).

Índices Negativos

Indican inicio desde el final de la lista.

El índice -1 corresponde al último elemento.

Rangos en Listas

Se definen con dos puntos (:).

El rango 2:4 incluye índices 2 al 3.

Métodos de Listas

Algunos métodos

list.append(elemento)

- pega un elemento al final de la lista

list.insert(pos,elemento)

- inserta un elemento en la posición pos

list.remove(elemento)

- borra la primera instancia del elemento en la lista

el=list.pop(pos)

- borra y regresa el elemento en la posición pos

el=list.pop()

- Borra y regresa el último elemento de la lista

list.clear()

- Borra todos los elementos de la lista

pos=list.index(elemento[,ini,fin])

- Regresa la posición de la primera instancia del elemento