universidad cenfotec
La U de la informática

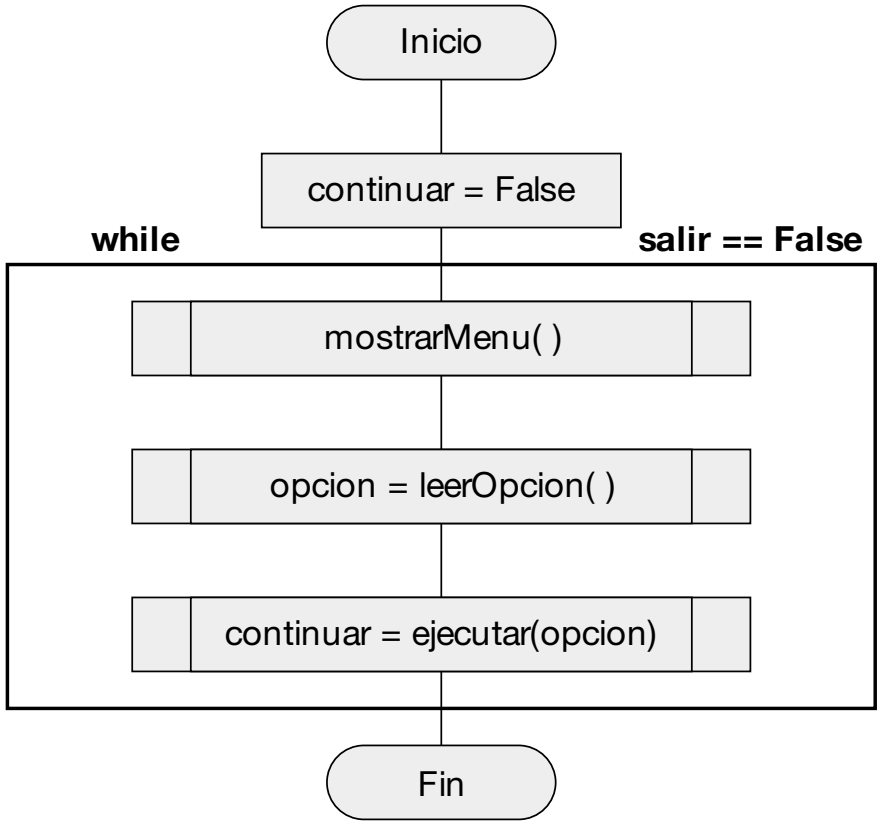
Principios de programación.

Abstracción procedimental..
Creación de un programa de consola con menú.

Objetivos

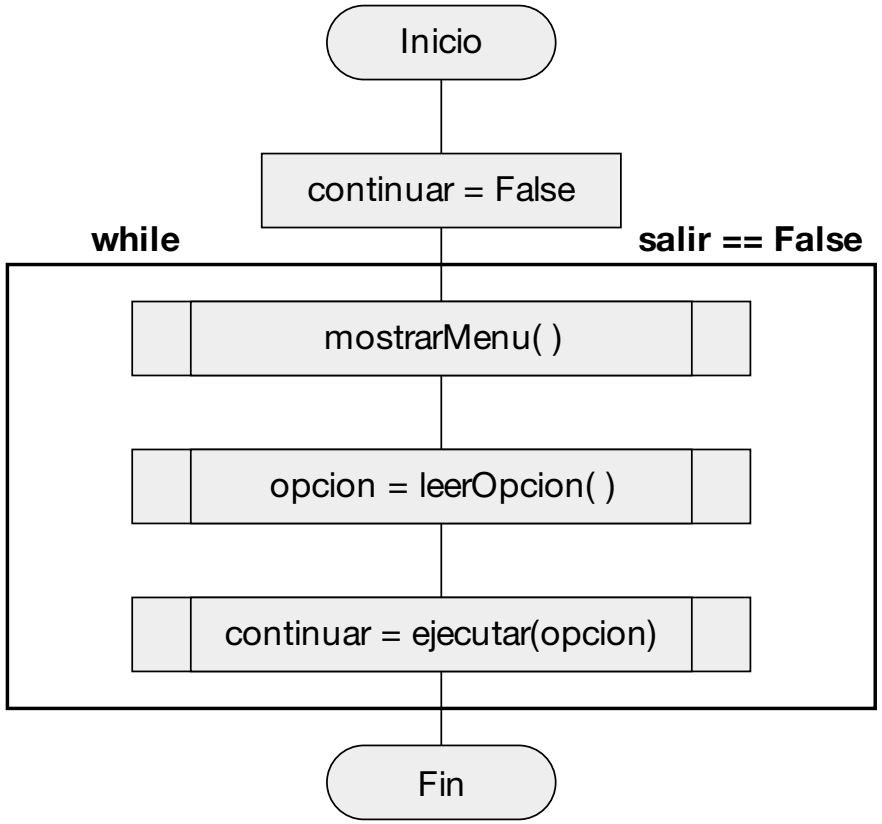
- Reconocer la aplicación de las rutinas en un ejemplo concreto.
- Interpretar las diferentes responsabilidades dentro del proceso para separarlas.
- Implementar el código provisto para conseguir el resultado deseado.
- Valorar el funcionamiento de las instrucciones provistas.
- Probar la ejecución del programa de acuerdo con las explicaciones.

Flujo principal




```
continuar = True
#Aquí van las rutinas
while(continuar == True):
    mostrarMenu()
    opcion =  leerOpcion()
    continuar = ejecutarOpcion(opcion)
```

Responsabilidades del flujo principal

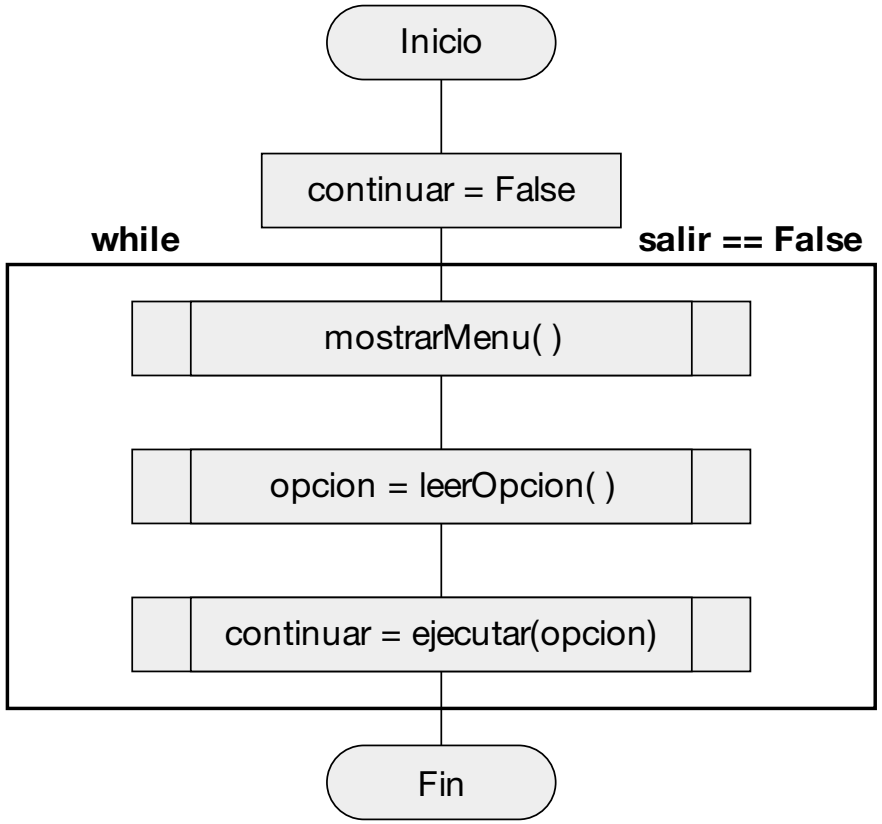


El flujo principal solo tiene la responsabilidad de llevar control del orden de ejecución y por medio del ciclo, cuando llega el momento debe terminar el programa.

La rutina “conoce” que es necesario mostrar el menú, y que una opción se seleccionará para ejecutarla, sin embargo, como se llevan a cabo todos estos procedimientos se escapan a su responsabilidad y fueron delegadas a otras rutinas.

En el caso del flujo principal lo que verdaderamente importa es tener un ciclo que funcione con una condición funcional y una modificación que pueda potencialmente acabar con el programa.

Explicación del diagrama



Con rutinas los diagramas se vuelven más pequeños, cuando se utiliza el símbolo que representa una rutina (el rectángulo con dos bordes) la expresión se parece más al diagrama general donde se enumeran los pasos.

Viendo únicamente este diagrama, se puede recuperar la esencia del funcionamiento del programa:

1. Mostrar las opciones.
2. Seleccionar una opción.
3. Ejecutar la acción correspondiente.

Explicación del código

El flujo principal el código que se ejecuta inmediatamente Python cuando pone a correr el programa.

En este caso consta de un ciclo while que se ejecutará mientras no seleccione salir en las opciones.

El flujo del programa será:

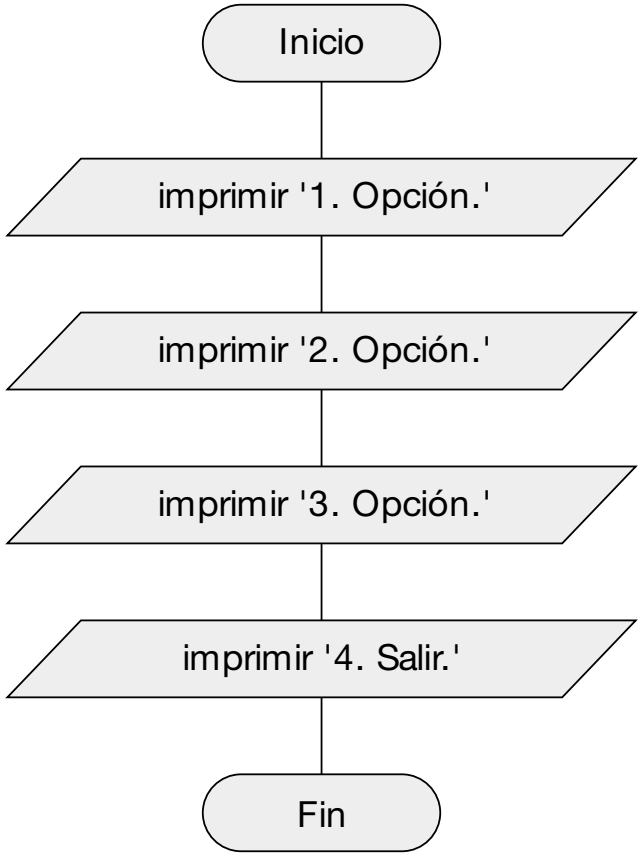
mostrar > leer > ejecutar

Una vez que acabe mientras “continuar” sea verdadero vuelve a empezar.

```
continuar = True
#Aquí van las rutinas
while(continuar == True):
    mostrarMenu()
    opcion =  leerOpcion()
    continuar = ejecutarOpcion(opcion)
```

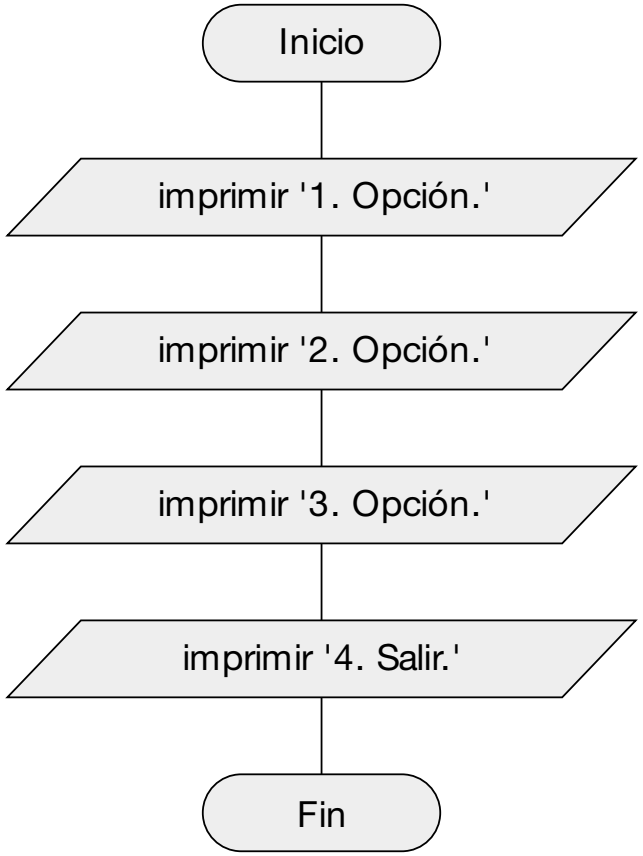
** Las rutinas no forman parte del flujo principal en Python, y deben ser definidas antes de ser llamadas.*

mostrarMenu()



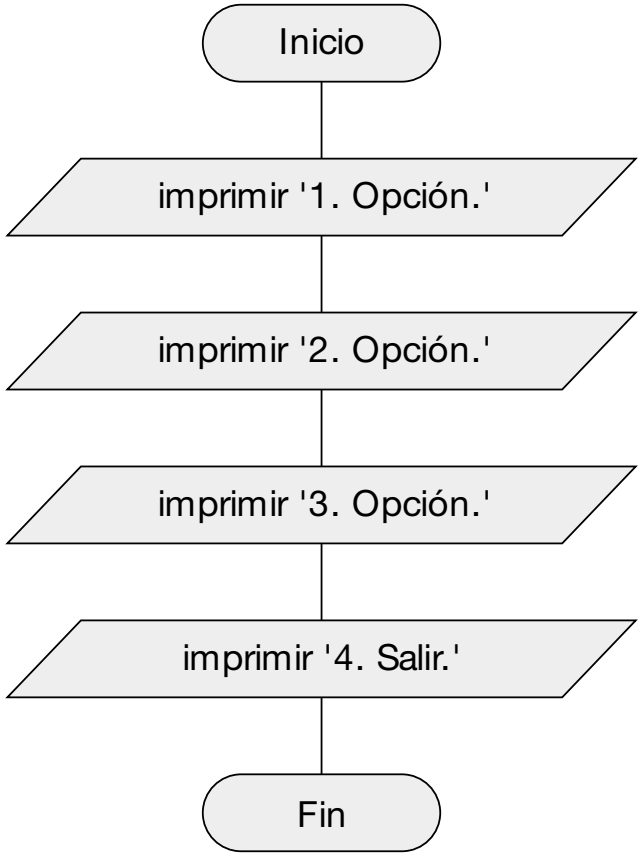

```
def mostrarMenu():
    print('----- Menú -----')
    print('1. Opción. ')
    print('2. Opción. ')
    print('3. Opción. ')
    print('4. Salir. ')
    print()
```

mostrarMenu() responsabilidades



Al analizar el diagrama es fácil reconocer que la única responsabilidad de esta rutina es mostrar las opciones en pantalla, no es saber cuál es la opción que el usuario selecciona, ni cómo se van a ejecutar; solamente imprime las opciones en consola y termina.

`mostrarMenu()` explicación del diagrama



Esta es una rutina que únicamente imprime mensajes, no es nada que no haya hecho antes.

mostrarMenu() explicación del código

universidad cenfotec
La U de la informática

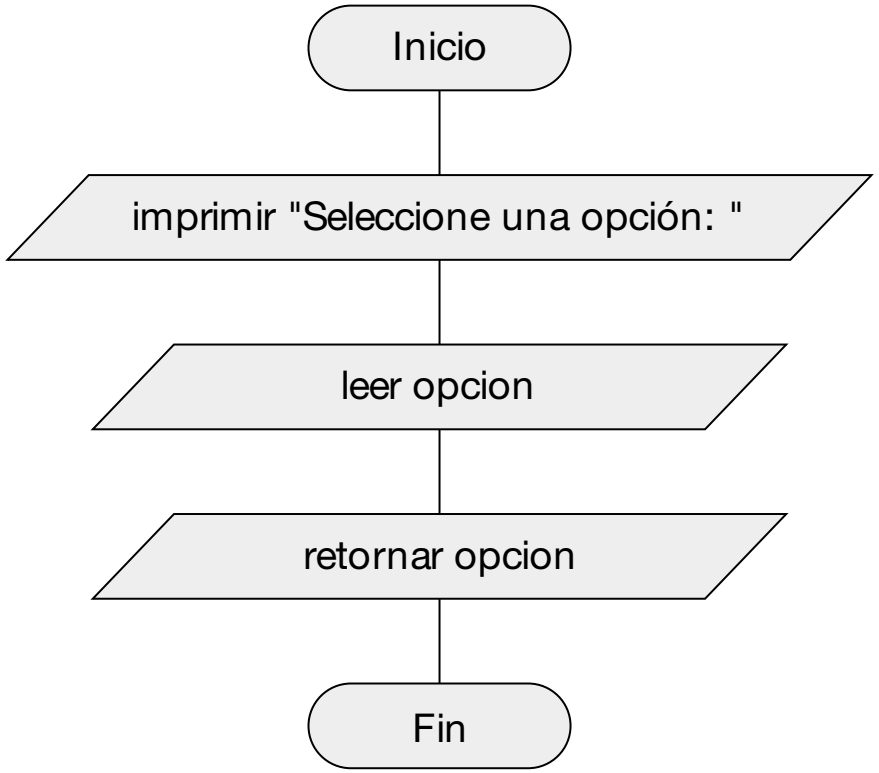
Cada línea de esta rutina es una instrucción de impresión.

Usando el ejemplo de una biblioteca, algunas opciones del menú podrían ser:

- Guardar un nuevo libro.
- Consultar disponibilidad.
- Ver préstamos vencidos.

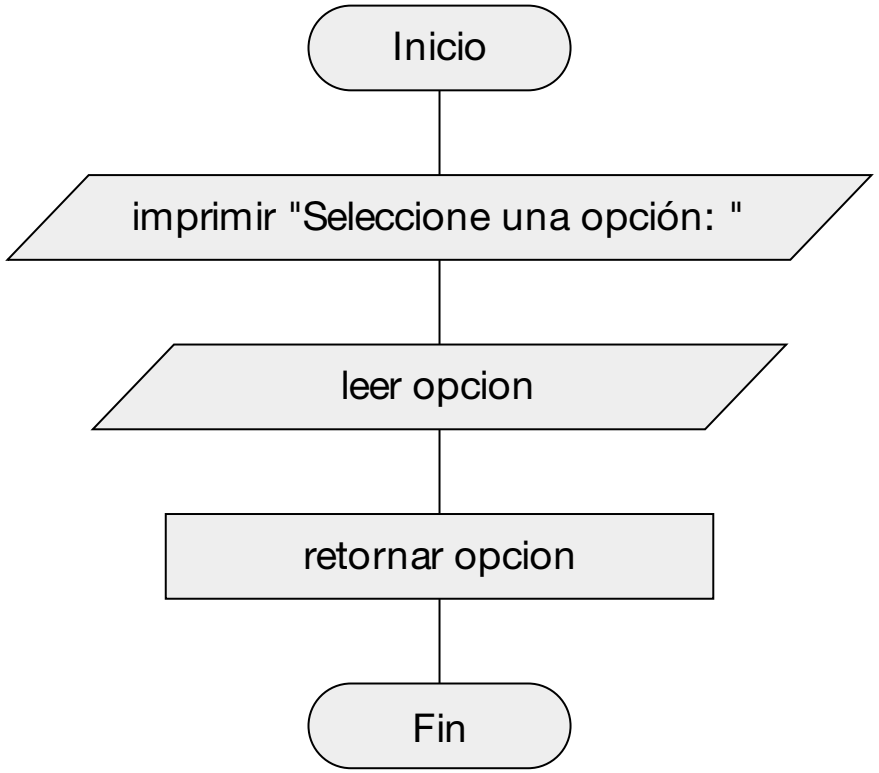
```
def mostrarMenu():
    print('----- Menú -----')
    print('1. Opción. ')
    print('2. Opción. ')
    print('3. Opción. ')
    print('4. Salir. ')
    print()
```

leerOpcion()



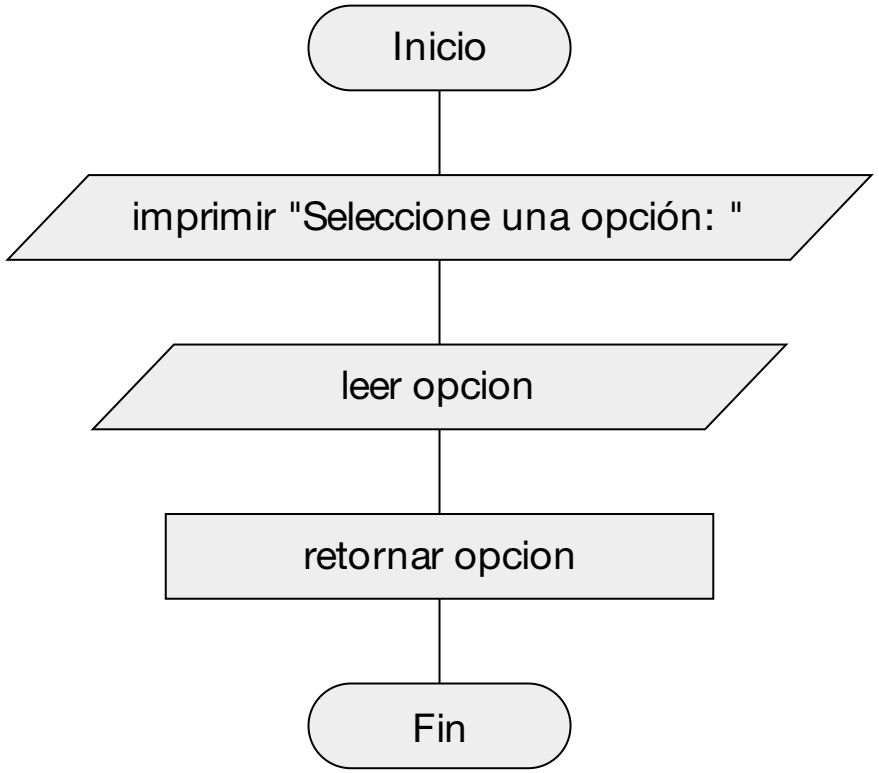

```python
def leerOpcion():
    try:
        opcion = int(input("Seleccione una opción: "))
        print()
        return opcion
    except:
        return -1
```

leerOpcion() responsabilidades



La única responsabilidad de la rutina es asegurarse que el valor de retorno es válido, esta rutina no “conoce” las opciones que fueron impresas pero sabe que el valor de retorno será un entero, en caso que no sea un entero disparará una excepción o retornará -1.

leerOpcion() explicación del diagrama



La rutina solamente imprime un mensaje y lee la variable correspondiente antes de retornarla.

El retorno no es una operación de impresión, por eso se diagrama como un proceso cualquiera.

leerOpcion() explicación del código

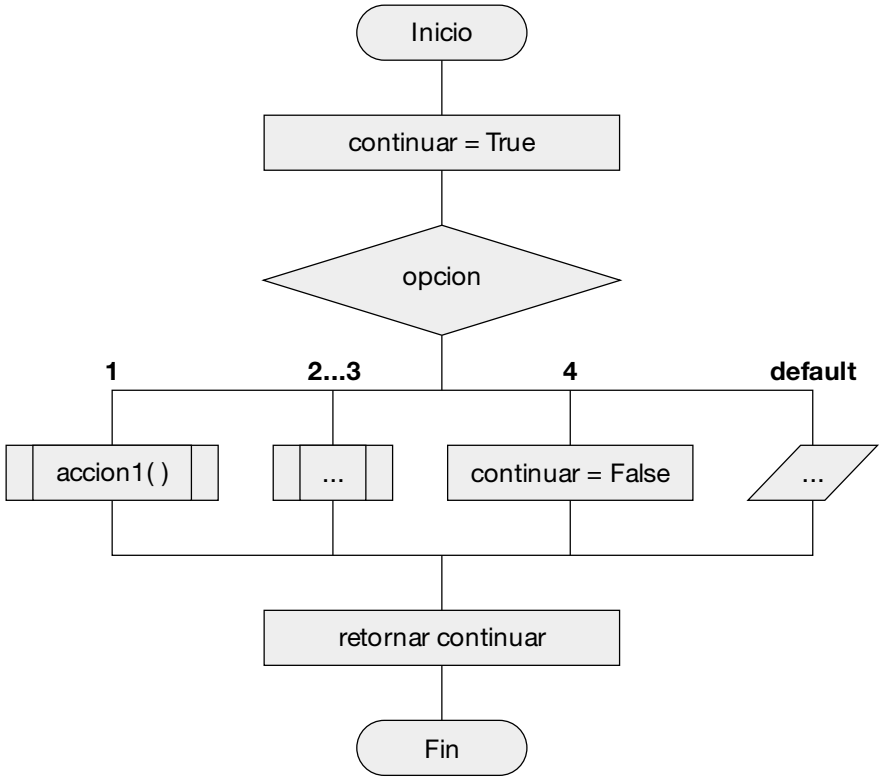
universidad cenfotec
La U de la informática

Las instrucciones que se incluyen en esta rutina no son nada nuevo, imprimir, leer y finalmente retornar, sin embargo, el ejemplo usa algo que no ha visto antes: un bloque que se usa para manejar errores.

Este bloque se llama try-except, en este caso específico significa que si el usuario ingresa algo que no sea un número (causando un error en la conversión) el programa atrapará esta excepción y en su lugar retornará -1.

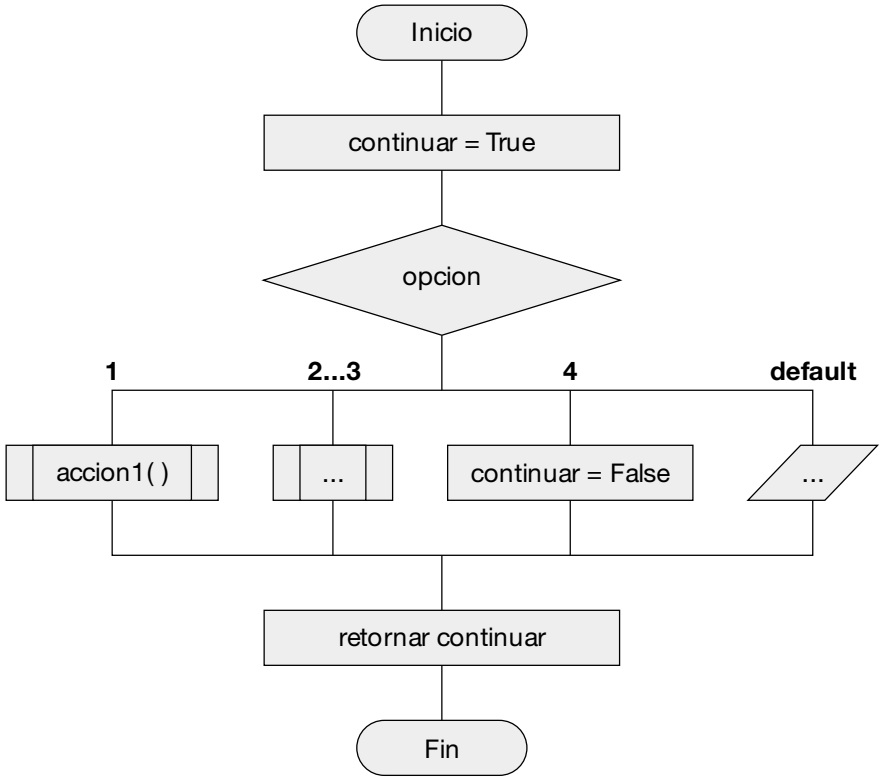
```
def leerOpcion():
    try:
        opcion = int(input("Seleccione una opción: "))
        print()
        return opcion
    except:
        return -1
```

ejecutarOpcion()




```python
def ejecutarOpcion(pOpcion):
    continuar = True
    if(pOpcion == 1):
        accion1()
    elif(pOpcion == 2):
        accion2()
    elif(pOpcion == 3):
        accion3()
    elif(pOpcion == 4):
        continuar = False
    else:
        print("Opción inválida")
    return continuar
```

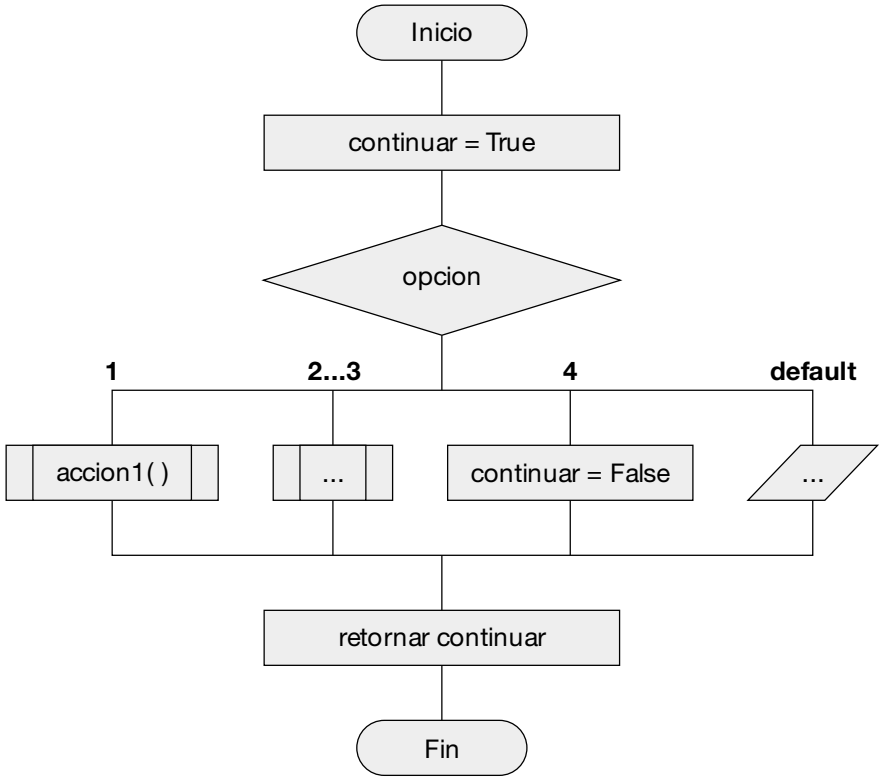
ejecutarOpcion() responsabilidades



La función tiene la responsabilidad de hacer la llamada a la rutina adecuada según la opción seleccionada, esta opción la recibe por parámetro, esto significa que es un requerimiento para funcionar, averiguar esa opción está fuera de sus responsabilidades.

La otra responsabilidad de la rutina es devolver el estado true/false de salir, para qué es necesario a la rutina no le corresponde saberlo.

ejecutarOpcion() explicación del diagrama



ejecutar() usa una condicional múltiple para correr diferentes rutinas de acuerdo con la opción seleccionada, por espacio no se muestran todos los casos ni el default, pero se especifican en el código.

En el caso del ejemplo la opción 4 es para salir del programa y en lugar de ejecutar una rutina solo cambia el valor booleano de la variable para retornar False en lugar de True al terminar.

ejecutar() explicación del código

universidad cenfotec
La U de la informática

Esta función es el cuerpo del programa, aquí es donde el flujo se redirige a la rutina que corresponde a la opción seleccionada.

La función recibe por parámetro un entero llamado “pOpcion” que controla la condicional múltiple.

Cada escenario corresponde a una opción del menú y como la condicional múltiple siempre tiene un caso predeterminado será imprimir un mensaje que la opción indicada no es válida.

```
def ejecutarOpcion(pOpcion):
    continuar = True
    if(pOpcion == 1):
        accion1()
    elif(pOpcion == 2):
        accion2()
    elif(pOpcion == 3):
        accion3()
    elif(pOpcion == 4):
        continuar = False
    else:
        print("Opción inválida")
    return continuar
```

Rutinas varias

universidad cenfotec
La U de la informática

Finalmente se incluyen las rutinas que se encargan de los procesos correspondientes a cada opción.

Los nombres accion1, accion2 y accion3 son pésimos para una rutina, es solamente con carácter ilustrativo.

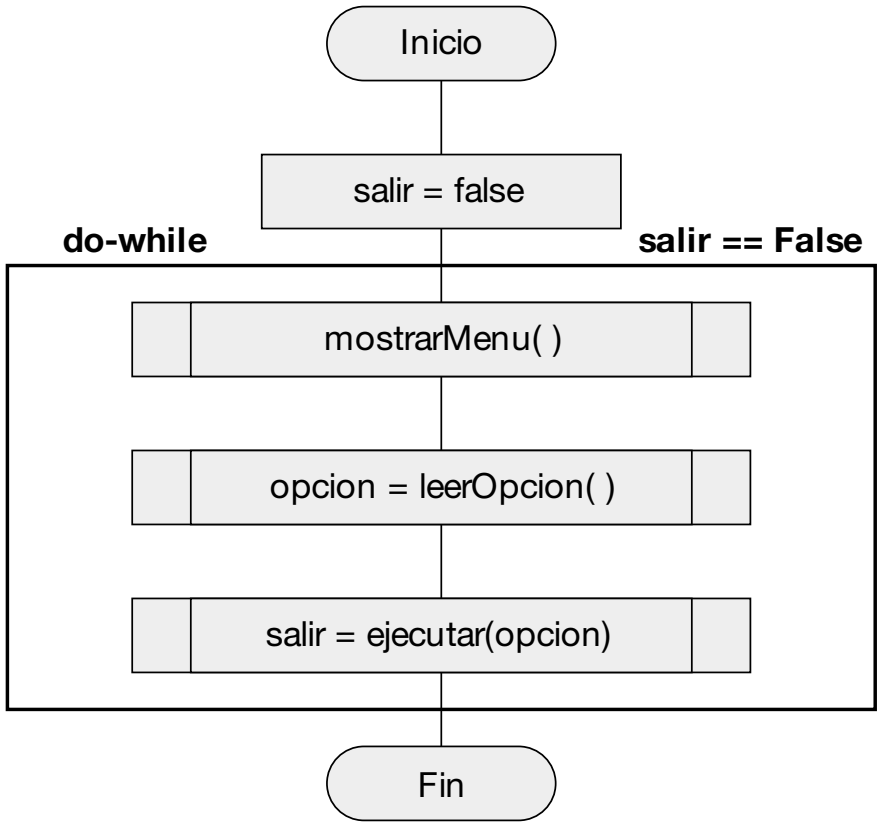
Estas rutinas tendrán a su vez diagramas y código propio que resuelva el procedimiento correspondiente.

```
def accion1():
   #El algoritmo para realizar la opción 1

def accion2():
   #El algoritmo para realizar la opción 2

def accion3():
   #El algoritmo para realizar la opción 3
```

Recapitulación



En el flujo principal podrá entonces reconocer todas las partes y el camino que tomará la ejecución:
El ciclo condicional while se ejecutará al menos una vez, su variable de control es “continuar” y la condición es “continuar == True”, esto significa que aún si selecciona “salir” como primera opción habrá visto el menú al menos una vez.
La variable “opcion” se asigna como resultado de la función leerOpcion para usarla como parámetro en la función ejecutar.
Finalmente ejecutarOpcion retorna el estado de “continuar”, es importante que “continuar” es la variable de control del ciclo entonces esta última llamada es también la modificación que finalmente acabará con el ciclo cuando el usuario decida salir del programa.

Conclusiones

universidad cenfotec
La U de la informática

- El diagrama explicativo ayuda a diferenciar cuáles serán finalmente las rutinas que se crearán para solucionar el algoritmo.

- Mantener las responsabilidades separadas de cada rutina es fundamental cuando es necesario hacer algún cambio.

- Una rutina bien nombrada ayuda a reconocer inmediatamente cuál es su responsabilidad.

- Una rutina no recibirá por parámetro un booleano para empezar con una condicional, esto por definición indicaría que quiere hacer dos cosas diferentes en la rutina, tendría dos responsabilidades diferentes.

- Las variables que se envían por parámetro y los nombres de los parámetros no tienen que coincidir, solo su tipo de dato.


universidad
cenfotec_
La U de la informática