Trabajo Práctico 2025
Preguntas Frecuentes

Algoritmos y Estructuras de Datos

Índice

Cómo armar los menús

Estamos aprendiendo a programar de manera estructurada y secuencial, por ende, se pide que los entregables del TP se ejecuten de esa manera. Evitar que un módulo vuelva a llamarse a si mismo, o que llame a otro módulo que vuelva a llamar al mismo.

Ejemplo de cómo deberían hacerse los menús:

```
#ejemplo de menú anidado para la opción 1
def cartel():
    print("....en construcción...")

def MENU():
    print(".....MENU PRINCIPAL.....")
    print("1- va al submenú de la opcion 1...")
    print("2- va al submenú de la opcion 2...")
    print("3- va al submenú de la opcion 3...")
    print("4- va al submenú de la opcion 4...")
    print("0- Fin DEL PROGRAMA")


def MENU2():
    print("          .....SUB MENU.....")
    print("          1- .....")
    print("          2- .....")
    print("          3- .....")
    print("          4- .....")
    print("          0- Fin del SUBMENU, retorna al menu anterior")

def Submenu():
    opc2 = 1  #asignacion interna para obligar al mientras a que entre aunque sea una vez
    while (opc2!=0):
      MENU2()
      opc2 = int(input("          Ingrese su opcion: "))
      while (opc2<0 or opc2>4):
         opc2 = int(input("Ingreso Invalido - reintente ... "))
      match opc2:
         case 1: cartel() #print("\n se ejecuta la opcion 1 del MENU2")
         case 2: cartel() #print("\n se ejecuta la opcion 2 del MENU2")
         case 3: cartel() #print("\n se ejecuta la opcion 3 del MENU2")
         case 4: cartel() #print("\n se ejecuta la opcion 4 del MENU2")
         case 0: print(' \nRETORNA AL MENU PRINCIPAL!!!')


# acá comienza el Programa Principal
opc = 1  # así lo obligo a entrar al mientras y lo convierto en un Repetir
while (opc!=0):
    MENU()
    opc = int(input("Ingrese su opcion: "))
    while (opc<0 or opc>4):
      opc = int(input("Ingreso Invalido - reintente ... "))
    match opc:
      case 1: Submenu()
      case 2: cartel() #print("\n se ejecuta la opcion 2 del MENU PRINCIPAL")
      case 3: cartel() #print("\n se ejecuta la opcion 3 del MENU PRINCIPAL")
      case 4: cartel() #print("\n se ejecuta la opcion 4 del MENU PRINCIPAL")
      case 0: print('\n\n GRACIAS POR USAR NUESTRO SISTEMA!!!!')
```

En otras palabras, evitar que una función se llame a sí misma, o bien que llame a otra función que luego la vuelva a llamar. Se considerarán inválidas las resoluciones que incluyan recursividad.

Declarativa de variables

Poner la declarativa de variables como comentario arriba de todos los módulos que desarrollemos en Python. La declarativa deberá tener el formato nombre: tipo, pudiendo agruparse las variables del mismo tipo.

Ejemplo trivial de declarativa de variables: supongamos que debemos desarrollar un módulo que realice la división entre 2 enteros llamados n1 y n2, previamente guardando el resultado en otra variable llamada n3. Sabemos que al dividir 2 enteros nuestro resultado puede contener decimales, por lo tanto, vamos a tener 3 variables: n1 y n2 como variables enteras y n3 como flotante.

```
'''
n1, n2: integer
n3: float
'''
def division():
    n1 = int(input("Ingrese el primer número: "))
    n2 = int(input("Ingrese el segundo número: "))
    n3 = n1 / n2
    print(n3)
```

Nombres de variables

Evitar los nombres de variables demasiado genéricos. El enunciado del TP es relativamente largo, por lo cual deberíamos evitar usar nombres como x, y, z o variable1, variable2. Si nuestra variable es la opción de un menú, ponerle de nombre opción u op. Si se trata del email de un usuario, utilizar email o email_usuario. Evitar los nombres en mayúsculas, y utilizar formato camelCase o separar las palabras con guiones bajos.

Recordemos que Python no acepta espacios en los nombres de variable, ni tampoco se acepta que las variables comiencen con números.

Evitar métodos de manejo de strings

No se podrán usar métodos de manejo de strings tales como find, join, split, strip, así como tampoco expresiones regulares, etc.

Evitar librerías externas (salvo las nombradas acá)

Evitar importar librerías de Python, excepto las que sean estrictamente necesarias para resolver el enunciado: random o math, date o datetime, os (opcional), getpass o similar.

Enviar la misma resolución

Todos los integrantes deben enviar LA MISMO resolución a través del CVG. Subir el archivo .py a la consigna. Si uno de los integrantes del equipo no envía la resolución del TP no se le podrá pasar la nota.

Defensa del TP

Todos los integrantes del equipo deben estar presentes en la defensa. A su vez, la nota del TP es individual. Si un integrante del equipo está ausente el o los días de la defensa, se le pasará ausente como nota final.

Por más que la resolución del TP sea correcta, el hecho de no estar presente o no saber responder a las preguntas de la defensa puede ser causa de desaprobar la entrega.

Evitar el uso excesivo de banderas

Evitar utilizar banderas para cualquier cosa, especialmente si están anidadas. Usar el ejemplo de este documento para generar los menús, donde las condiciones de salida están dentro de los bucles.

Restricciones del Paradigma Estructurado

La materia adopta el paradigma de programación estructurada, por lo que queda estrictamente prohibido el uso de instrucciones que rompan el flujo normal de ejecución dentro de estructuras iterativas, tales como:

- La instrucción return dentro de bucles.
- El uso de break para salir de un ciclo antes de que se cumpla la condición de finalización.
- La función exit() o cualquier otro mecanismo que termine abruptamente la ejecución del programa.

Respetar los tiempos de entrega del TP

El campus virtual (CVG) maneja las entregas de manera automática. No se aceptarán entregas en una fecha y hora posterior a la estipulada en el CVG. Incluso, si sólo uno de los integrantes del equipo entrega la resolución luego de la fecha permitida, esa persona tendrá un desaprobado de manera automática.

Respetar el tamaño de los equipos

Si el enunciado estipula equipos de 4 personas, respetar lo que dice el mismo. No crear equipos de 3 o de 5 personas, ni mucho menos equipos unipersonales.

No utilizar estructuras de datos que no se hayan dictado en clases

Cualquier resolución que incluya estructuras de datos no dadas en clase (ejemplo: listas para el TP1 o diccionarios, conjuntos, pilas, colas, etc) será tomada como incorrecta, y el equipo que utilice dichas estructuras será desaprobado. Respetar la consigna y resolver los problemas con las herramientas provistas en clase.

La resolución DEBE poder ejecutarse

La entrega del TP debe ser un programa en Python funcional. No se admitirán programas que no funcionen o con errores de sintaxis. Utilizar las clases de consulta de cada profesor para poder evacuar dudas ANTES de la fecha de entrega del TP. Les recordamos que pueden ir a la consulta

de cualquier profesor, las mismas se encuentran listadas en el sitio web de la facultad: https://www.frro.utn.edu.ar/horarios_consulta_dptoisi2023.php?cont=349&subc=26 .

Sobre la interacción con el usuario y validaciones y control de errores.

El programa debe ser amigable con el usuario, mostrando mensajes claros y evitando que se cierre abruptamente en caso de errores.

En caso de ingresar datos inválidos, el programa debe solicitar nuevamente el ingreso de los datos.

Todas las entradas de datos deben ser validadas para asegurar que cumplan con los requisitos especificados.