Problemas de estructuras repetitivas

1) Programa que permite calcular el factorial de un número entero.

Matemáticamente el factorial de un número se representa así: 3!, que se lee factorial de 3, lo cual equivale a lo siguiente:

3! = 1*2*3 = 6, por lo tanto el factorial de 3 es 6, otros ejemplos

4! = 1*2*3*4 = 24

5! = 1*2*3*4*5 = 120

Aquí conocemos la cantidad de veces que ciclo se va ejecutar en el caso de 3! el ciclo se ejecutara 3 veces, usamos un **for.** El **código** en Python es:

```
"""
    Programa que permite calcular el factorial
    de un numero entero.
"""
# Inicializacion de variables
num = 0
fact = 1
# Entrada datos
num = int(input("De numero entero: "))
# Proceso
for i in range(1, num + 1):
    fact *= i
print("\nEl factorial de ", num, " es: ", fact)
```

Nota: El factorial de **8** es: **40320**, mientras más grande es el numero cuyo factorial queremos calcular su factorial crece en forma exponencial por lo que es conveniente que la variable en algunos casos sea **float** y no **int.**

2) Programa que lee una cantidad de valores enteros y un valor b. Determine e imprima el porcentaje de valores leídos menores que b.

De acuerdo al enunciado se va procesar una lista de valores enteros cuyo tamaño es conocido, por lo tanto el ciclo a usar será un ciclo **for**, el programa tendrá como entrada la cantidad de valores a leer y valor de **b**, cual nos servirá para determinar el porcentaje de los valores leídos que so menores que el.

Código:

```
"""
   Programa que lee ua lista de  valores eteros y un valor b
   Determine e imprima el porcentaje de valores leidos
   menores que b.
"""
# Inicializar variables
cant = 0  # Cantidad de enteros a leer.

valor = 0  # Guarda el valor leido.
porc=0.0 # Para guardar el porcentaje.
b=0 # Valor a comparar
cont=0 # Cuenta valores menores que comp.
```

```
# Entrada de datos
cant=int(input("Indique cuantos valores seran leidos: "))
b=int(input("Indique de b: "))
# Ciclo de lectura y Proceso
for i in range(1,cant+1):
    valor=int(input("De un valor entero: "))
    if (valor<b):
        cont+=1
# Calculos finales salida resultados
porc=cont/cant*100
print("\nPorcentaje = ",porc)
```

Ejecución:

```
Indique cuantos valores seran leidos: 10
Indique de b: 8
De un valor entero: 5
De un valor entero: 12
De un valor entero: 7
De un valor entero: 9
De un valor entero: 8
De un valor entero: 25
De un valor entero: 4
De un valor entero: 16
De un valor entero: 12
De un valor entero: 67

Porcentaje =  30.0
```

3) Si desconociéramos el tamaño de la lista no podríamos usar un ciclo for, sino un ciclo while con centinela.

Código:

```
"""
   Programa que lee ua lista de  valores enteros y un valor b
   Determine e imprima el porcentaje de valores leidos
   menores que b.
"""
# Inicializar variables
resp = "s"  # variable centinela
valor = 0  # Guarda el valor leido.
cant = 0  # Catidad de valores leidos.
porc = 0.0  # Para guardar el porcentaje.
b = 0  # Valor a comparar
cont = 0  # Cuenta valores menores que comp.
# Entrada de datos y Ciclo leyendo y procesando la lista
b = int(input("Indique de b: "))
```

```
15  # Ciclo de lectura y Proceso
16  while (resp.upper() == "S"):
17      valor = int(input("De un valor entero: "))
18      cant += 1
19      if (valor < b):
20          cont += 1
21      resp = input("Lee mas valore Si-->S, No-->N ")
22  # Calculos finales salida resultados
23  porc = cont / cant * 100
24  print("\nPorcentaje = ", porc)
```

Línea 16: aquí usamos la función de cadena **upper** aplicada a la variable centinela, la razón, esto nos da la posibilidad de ingresar el carácter **"s"** en mayúscula o minúscula, ya que **upper** convierte en mayúscula el carácter leído, otra forma seria escribir el código así:

```
while(resp=="s" or resp=="S"):
```

Aquí evaluamos si el carácter leído esta en minúscula o mayúscula, esto logar el mismo efecto que al usar **upper**. En el programa nos vemos en la necesidad de contar la cantidad de caracteres leídos, ya que no conocemos el tamaño de la lista, línea 20, en la 21 controlamos el valor del centinela introduciendo una **s** si queremos continuar o una n si queremos detener el proceso de lectura de los valores de la lista.
Datos de prueba:

```
Indique de b: 12
De un valor entero: 3
Lee mas valore Si-->S, No-->N s
De un valor entero: 34
Lee mas valore Si-->S, No-->N s
De un valor entero: 3
Lee mas valore Si-->S, No-->N s
De un valor entero: 41
Lee mas valore Si-->S, No-->N n

Porcentaje =  50.0
```

4) Programa que lee una lista de N enteros y determina cual de los enteros leídos es el mayor y cual el menor. Aquí usaremos un **for** es conocida el tamaño de la lista.

Para procesar la lista debemos usar una variable bandera con la finalidad de determinar si el primer valor de la lista ya fue procesado, el cual usaremos como valor de referencia de partida para comparar con valores siguientes de la lista y determinar el mayor y el menor valor de la lista. Les recuerdo que una bandera es una variable que procesa dos valores, cierto o falso, 0 o 1 etc., al pasar de un valor a otro nos está indicando que el proceso o evento que queremos controlar ya tuvo lugar.

Código:

```
1   """
2       Programa que lee una lista de N enteros y determina
3       cual de los enteros leidos es el mayor y cual
4       el menor.
5   """
6   # Inicializar variables
7   cant = 0   # Cantidad de enteros a leer.
8   band = 0   # Variable que verifica si el 1er valor fue leido. Bandera
9   mayor = menor = 0   # Guardan el mayor o el menor valor procesado en la list
10  valor = 0   # Guarda el valor leido.
11  # Entrada de datos
12  cant = int(input("Cantidad de numeros a leer: "))
13  # Ciclo de lectura y Proceso datos.
14  for i in range(1, cant + 1):
15      valor = int(input("De un numero entero: "))
16      if (band == 0):
17          # Fijamos el 1er valor leido como el mayor y el menor
18          mayor = valor
19          menor = valor
20          band = 1 # Cambia la bandera indicando que el 1er valor fue procesado
21      elif (mayor < valor):
22          mayor = valor
23      elif (menor > valor):
24          menor = valor
25  # Salida de resultados
26  print("\nEl mayor valor leido fue: ", mayor)
27  print("El menor valor leido fue: ", menor)
```

Una vez procesado el 1er valor de la lista, líneas desde la 16 a la 20, considerando este valor el mayor y el menor de la lista, los siguientes valores se comparan contra las variables **mayor** y **menor**, línea 21, aquí verificamos si el valor leído es mayor que el valor que considerábamos el mayor de la lista, si esto se cumple, debemos cambiar el contenido de **mayor** al nuevo valor encontrado, de no cumplirse, se procede con la línea 23, aquí verificamos si el valor leído es menor que el valor que considerábamos el menor de la lista, si esto se cumple, debemos cambiar el contenido de **menor** al nuevo valor encontrado, de no cumplirse, se procesa el siguiente valor de la lista. Al terminar el ciclo en mayor tendremos el mayor de la lista y en menor el menor de la lista. **Ejecución**:

```
Cantidad de numeros a leer: 7
De un numero entero: 3
De un numero entero: 15
De un numero entero: 10
De un numero entero: 1
De un numero entero: 25
De un numero entero: -3
De un numero entero: 12

El mayor valor leido fue:  25
El menor valor leido fue:  -3
```

5) Programa que lea un entero positivo y determina si el entero leído es primo o no.

Consideraciones:

a) Un entero positivo es primo si es divisible por1 y el mismo.
b) Un número primo es impar.
c) El procedimiento a seguir es hallar un entero entre 2 y $\sqrt{N}$ que lo divida exactamente, si lo hay *no es primo*, si no lo hay *es primo*. Pasos a seguir
d) Hallar la $\sqrt{N}$.
e) Obtener la parte entera P del paso anterior.
f) Hallar un entero entre 2 y P que divida exactamente a N.
g) Si encontramos un entero *no es primo*, de lo contrario *es primo*.

Código:

```
# Importamos la funcion sqrt
from math import sqrt
# Inicializar variables
pnum = 0 # Valor a evaluar si es primo o no
raiz=0.0 # Guarda la raiz dl numero
eraiz=0 # Guarda la parte entera de la raiz
i=0 # Variable controladora del ciclo
# Entrada dato
pnum=int(input("De un entero positivo: "))
if (pnum%2==0):
    print("No es primo")
else:
    raiz = sqrt(pnum) #Obtenemos la raiz cuadrada
    eraiz = int(raiz) # Obtemos la parte entera
    i = 2
    # Ciclo para encontrar un numero i que divida exactamemte
    # el numero leido.
    while (pnum % i != 0 and i < eraiz):
        i += 1
    # Se verifica si i divide exactamente al numero leido
    # si lo no hay es primo, de lo cotrari es primo.
    if (pnum % i != 0):
        print("Es primo")
    else:
        print(" No Es primo")
```

Ejecución:

```
De un entero positivo: 17
Es primo
```

```
De un entero positivo: 721
 No Es primo
```

6) Hacer un programa en Python 3.6 que lea dos valores enteros, el primero leído debe ser menor que el segundo en caso de no cumplirse esto emitir un mensaje indicando el error (aquí validamos el valor de entrada), en caso de cumplirse determine para cada numero incluido en el intervalo definido, incluso los extremos, los valores que lo dividen exactamente, la cantidad de los valores que lo dividen y su suma. Al final se debe indicar el número del intervalo que obtuvo la mayor cantidad de divisores. La salida debe quedar así:

```
De el valor inicial: 14
De el valor final: 25
14  = 1  2  7   Divisores =  3  Suma =  10
15  = 1  3  5   Divisores =  3  Suma =  9
16  = 1  2  4  8   Divisores =  4  Suma =  15
17  = 1   Divisores =  1  Suma =  1
18  = 1  2  3  6  9   Divisores =  5  Suma =  21
19  = 1   Divisores =  1  Suma =  1
20  = 1  2  4  5  10   Divisores =  5  Suma =  22
21  = 1  3  7   Divisores =  3  Suma =  11
22  = 1  2  11   Divisores =  3  Suma =  14
23  = 1   Divisores =  1  Suma =  1
24  = 1  2  3  4  6  8  12   Divisores =  7  Suma =  36
25  = 1  5   Divisores =  2  Suma =  6

El numero con mayor cantidad de divisores es:  24  con total de:  7

Fin del Programa
```

En programación, la validación de **datos** es el proceso de asegurar que un **programa** funcione con **datos** limpios, correctos y útiles. ... La validación de **datos** compruebe que los **datos** están aptos para el propósito deseado, válidos, razonables y seguros antes de su procesamiento, permiten asegurar que los valores con los que se vaya a operar estén dentro de determinado rango. Por ejemplo se puede comprobar el contenido de una variable sea de un tipo en particular; o que el dato tenga determinada característica.

También se debe tener en cuenta qué hará nuestro código cuando una validación falle, ya que queremos darle información al usuario que le sirva para procesar el error. En cualquier caso, lo importante es que el resultado generado por nuestro código cuando funciona correctamente y el resultado generado cuando falla debe ser claramente distinto. Para los efectos de este curso estos procesos de validación los llevaremos a cabo usando la sentencia IF.

Código:

```
1  # Inicializar variables
2  valorInicial = valorFinal = 0  # establece los limites del ciclo o de los valores a evaluar
3  numerador = denominador = 0 # variables controladora de los ciclos
4  resto = 0 # guarda el resto de la division entre el numerador y el denominador
5  mayorCant = 0 # guarda la maypr cantidad de denominadores
6  cantDiv = 0 # guarda la cantidad de divisores de un numero
7  mnum = 0 # Guarda el numero con la mayor cantidad de divisores
8  band = 0 # para controlar el 1er elemento de la lista
9  sumaDiv = 0 # Guarda la suma de todos los divisores
10 # entrada de datos
11 valorInicial = int(input("De el valor inicial: "))
12 valorFinal = int(input("De el valor final: "))
13 # verificamos si el valor inicial es menor que el valor final
14 if (valorInicial < valorFinal):
15     for numerador in range(valorInicial, valorFinal + 1):
16         print(numerador, " = ", end="")
17         sumaDiv = 0
18         cantDiv = 0
19         for denominador in range(1, numerador):
20             resto = numerador % denominador
21             if resto == 0:
22                 print(denominador, " ", end="")
23                 cantDiv += 1
24                 sumaDiv += denominador
25         print(" Divisores = ", cantDiv, " Suma = ", sumaDiv)
26         if (band == 0):
27             mayorCant = cantDiv
28             mnum = numerador
29             band = 1
30         elif (mayorCant < cantDiv):
31             mayorCant = cantDiv
32             mnum = numerador
33     print("\nEl numero con mayor cantidad de divisores es: ",mnum," con total de: ",mayorCant)
34 else:
35     print("\nValores Incorrectos")
36 print("\nFin del Programa")
```

Verificamos que el valor inicial del intervalo es menor que el valor final

Codigo que se ejecutara en el caso de que la comprobacion sea la correcta

En el caso de que la verificacio sea incorrecta se ejecutara el siguiente codigo

Líneas 1 a 9: Inicializamos todas las variables que usara el programa.
Líneas 11 a 12: se procede a dar entrada al valor inicial y final del intervalo a evaluar.
Línea 14: verificamos si el valor inicial es menor que el valor final, de cumplirse ejecutamos las intrusiones del bloque formado por las líneas 15 hasta la 33, de no cumplirse se ejecuta la línea 35 mostrando un mensaje de error.
Líneas 15 a 33:

Línea 15: ciclo que permite recorrer todos los valores de intervalo definido, por ejemplo si el intervalo está entre 14 y 25 la variable numerador tomara todos estos valores, dentro del cuerpo de este ciclo tenemos:

Línea 16. Se imprime un valor del intervalo, note que la sentencia print tiene end = “”, esto impide el salto de línea, manteniendo el cursor en la línea de impresión.
Líneas 17 a 18: se inicializan las variables sumaDiv y cantDiv a cero, esto debido a que cada vez que se procese un nuevo valor del intervalo estas se den inicializar a cero de no hacerlo ellas acumularían los valores del numerador anterior dando resultados erróneas.
Línea 19: ciclo que permite generar todos los valores por los cuales se dividirá el numerador en proceso, ejemplo si el numerador es 14 este ciclo generara valores entre 1 y 13 los cuales dividirán a 14.
Línea 20: se realiza la división entera entre el numerador entre el denominador para obtener el resto de esta división (% operador resto).
Línea 21: se verifica si el resto de esta división igual a cero de ser así se ejecutan:

Línea 22: se imprime el denominador que da resto cero, ya que divide exactamente al numerador.
Línea 23: se incrementa en 1 la variable ya que hemos encontrado un valor que divide exactamente al numerador de turno.
Línea 24: se acumula este valor que divide al numerador de turno en la variable sumaDiv.

Línea 25: se sale del ciclo for controlado por la variable denominador y se imprime la cantidad de valores que dividen exactamente al numerador en turno y la suma de los mismos.
Línea 26 a 32: se determina cual de los valores procesados del intervalo tiene la mayor cantidad de valores que lo dividen exactamente.
Línea 33: marca el fin del ciclo controlado por la variable numerador y se imprime el numero con la mayor cantidad de valores que lo dividieron.