In [1]:
```
#MAESTRÍA EN INTELIGENCIA ARTIFICIAL APLICADA
#Pruebas de software y aseguramiento de la calidad
#Dr. Gerardo Padilla Zárate

#Actividad 4.2. Análisis de requisitos
#CARLOS ENRIQUEZ GORGONIO
#A01793102
#20 de febrero de 2024
```

In [2]:
```
'''Ejercicio 1. Compute statistics
Detalles:
Req1. The program shall be invoked from a command line. The program shall receive a file as parameter. The file will contain a list of items (presumable numbers).
Req 2. The program shall compute all descriptive statistics from a file containing numbers. The results shall be print on a screen and on a file named StatisticsResults.txt. All co
be calculated using the basic algorithms, not functions or libraries.The descriptive statistics are mean, median, mode, standard deviation, and variance.
Req 3. The program shall include the mechanism to handle invalid data in the file. Errors should be displayed in the console and the execution must continue.
Req 4. The name of the program shall becomputeStatistics.py
Req 5. The minimum format to invoke the program shall be as follows:python computeStatistics.py fileWithData.txt
Req 6. The program shall manage files having from hundreds of items to thousands of items.
Req 7. The program should include at the end of the execution the time elapsed for the execution and calculus of the data. This
number shall be included in the results file and on the screen.
Req 8. Be compliant with PEP8.
'''
```

Out[2]:
```
'Ejercicio 1. Compute statistics\nDetalles:\nReq1. The program shall be invoked from a command line. The program shall receive a file as parameter. The file will contain a list of
items (presumable numbers).\nReq 2. The program shall compute all descriptive statistics from a file containing numbers. The results shall be print on a screen and on a file named
StatisticsResults.txt. All computation MUST \nbe calculated using the basic algorithms, not functions or libraries.The descriptive statistics are mean, median, mode, standard devi
ation, and variance.\nReq 3. The program shall include the mechanism to handle invalid data in the file. Errors should be displayed in the console and the execution must continue.
\nReq 4. The name of the program shall becomputeStatistics.py\nReq 5. The minimum format to invoke the program shall be as follows:python computeStatistics.py fileWithData.txt\nRe
q 6. The program shall manage files having from hundreds of items to thousands of items.\nReq 7. The program should include at the end of the execution the time elapsed for the ex
ecution and calculus of the data. This \nnumber shall be included in the results file and on the screen.\nReq 8. Be compliant with PEP8.\n'
```

In [3]:
```
!pip install pylint
!pip install pylint[spelling]
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pylint in c:\programdata\anaconda3\lib\site-packages (2.16.2)
Requirement already satisfied: platformdirs>=2.2.0 in c:\programdata\anaconda3\lib\site-packages (from pylint) (3.10.0)
Requirement already satisfied: astroid<=2.16.0-dev0,>=2.14.2 in c:\programdata\anaconda3\lib\site-packages (from pylint) (2.14.2)
Requirement already satisfied: isort<6,>=4.2.5 in c:\programdata\anaconda3\lib\site-packages (from pylint) (5.9.3)
Requirement already satisfied: mccabe<0.8,>=0.6 in c:\programdata\anaconda3\lib\site-packages (from pylint) (0.7.0)
Requirement already satisfied: tomlkit>=0.10.1 in c:\programdata\anaconda3\lib\site-packages (from pylint) (0.11.1)
Requirement already satisfied: dill>=0.3.6 in c:\programdata\anaconda3\lib\site-packages (from pylint) (0.3.6)
Requirement already satisfied: colorama>=0.4.5 in c:\programdata\anaconda3\lib\site-packages (from pylint) (0.4.6)
Requirement already satisfied: lazy-object-proxy>=1.4.0 in c:\programdata\anaconda3\lib\site-packages (from astroid<=2.16.0-dev0,>=2.14.2->pylint) (1.6.0)
Requirement already satisfied: wrapt<2,>=1.14 in c:\programdata\anaconda3\lib\site-packages (from astroid<=2.16.0-dev0,>=2.14.2->pylint) (1.14.1)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pylint[spelling] in c:\programdata\anaconda3\lib\site-packages (2.16.2)
Requirement already satisfied: platformdirs>=2.2.0 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (3.10.0)
Requirement already satisfied: astroid<=2.16.0-dev0,>=2.14.2 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (2.14.2)
Requirement already satisfied: isort<6,>=4.2.5 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (5.9.3)
Requirement already satisfied: mccabe<0.8,>=0.6 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (0.7.0)
Requirement already satisfied: tomlkit>=0.10.1 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (0.11.1)
Requirement already satisfied: dill>=0.3.6 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (0.3.6)
Requirement already satisfied: colorama>=0.4.5 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (0.4.6)
Requirement already satisfied: pyenchant~=3.2 in c:\users\traba\appdata\roaming\python\python311\site-packages (from pylint[spelling]) (3.2.2)
Requirement already satisfied: lazy-object-proxy>=1.4.0 in c:\programdata\anaconda3\lib\site-packages (from astroid<=2.16.0-dev0,>=2.14.2->pylint[spelling]) (1.6.0)
Requirement already satisfied: wrapt<2,>=1.14 in c:\programdata\anaconda3\lib\site-packages (from astroid<=2.16.0-dev0,>=2.14.2->pylint[spelling]) (1.14.1)
```

In [4]:
```
import sys
import time
```

In [7]:
```
#La función promedio opera con la formula:(suma de todos los valores)/(numero total de valores)
def promedio(valores):
    calculo_promedio = sum(valores) / len(valores)
    return calculo_promedio
```

In [8]:
```
#La función mediana devuelve el valor central de una lista ordenada de numeros, si es impart el promedio de los 2 centrales
def mediana(valores):

    #Primero se crea una variable para ordenar los numeros de menor a mayor
    lista_ordenada = sorted(valores)
    #Creamos una variable para obtener la longitud de la lista, para saber si es par o impar
    longitud = len(lista_ordenada)
    calculo_mediana = None

    #Si la cantidad de valores es un numero par, promediamos los dos valores centrales de la lista
    if longitud % 2 == 0:
        media1 = lista_ordenada[longitud // 2 - 1]
        media2 = lista_ordenada[longitud // 2]
        calculo_mediana = (media1 + media2) / 2
    #Si la cantidad de numeros es impar, simplemente obtenemos el valor en el centro de la lista
    else:
        # If the length of the array is odd, return the middle element
        calculo_mediana = lista_ordenada[longitud // 2]

    return calculo_mediana
```

In [17]:
```
#La función moda devuelve el valor que se repite el mayor numero de veces
def moda(valores):

    #Si nuestra cadena esta vacía lo indicamos
    if not valores:
        return "Sin valores"

    #Creamos un arreglo para almacenar las repeticiones del valor
    frecuencia = {}
    for valor in valores:
        frecuencia[valor] = frecuencia.get(valor, 0) + 1

    #Obtenemos el valor maximo dentro de nuestro arreglo
    maximo = max(frecuencia.values())

    #Ubicamos el valor de maxima repetición dentro de la lista
    calculo_moda = [key for key, value in frecuencia.items() if value == maximo]

    return "Sin valores" if len(calculo_moda) == len(frecuencia.values()) else calculo_moda[0]
```

In [18]:
```
#La funcion desviacion estandar nos indica que tan dispersos estan los datos con relación a la media
def desviacion_estandar(valores):

    # Obtenemos el promedio
    promedio = sum(valores) / len(valores)

    # Obtenemos la diferencia cuadratica del promedio
    diferencia2 = [(x - promedio) ** 2 for x in valores]

    # Obtenemos la varianza
    varianza = sum(diferencia2) / len(valores)

    # Obtenemos la desviación estandar a traves de la raiz cuadrada de la varianza
    calculo_desviacion = varianza ** 0.5

    return calculo_desviacion
```

In [19]:
```python
def varianza(valores):

    longitud = len(valores)

    # Obtenemos el promedio
    promedio = sum(valores) / longitud

    # Obtenemos la diferencia cuadratica del promedio
    diferencia2 = sum((x - promedio) ** 2 for x in valores)

    # Obtenemos la varianza
    calculo_varianza = diferencia2 / (longitud - 1)

    return calculo_varianza
```

In [43]:
```python
#Función para imprimir las estadisticas descriptivas, considerando el tiempo de ejecución
def impresora(ruta):
    try:
        start_time = time.time()
        #Abrimos el archivo
        with open(ruta, 'r', encoding="utf-8") as archivo:
            renglones = archivo.readlines()

        with open(ruta, 'r', encoding="utf-8") as archivo:
            # Read the numbers from the file and convert them to a list
            valor_origen = []
            for index, renglon in enumerate(archivo):
                try:
                    valor_origen.append(float(renglon.strip()))
                except ValueError:
                    print(f"Error: Se encontro en el renglon {index+1} un valor no numerico")

            imp_promedio = promedio(valor_origen)
            imp_mediana = mediana(valor_origen)
            imp_moda = moda(valor_origen)
            imp_desviacion = desviacion_estandar(valor_origen)
            imp_varianza = varianza(valor_origen)

            resultado = (
                f"Metricas.\n SUMATORIA: {len(renglones)}\n PROMEDIO: {imp_promedio}\n"
                f" MEDIANA: {imp_mediana}\n MODA: {imp_moda}\n"
                f" DESVIACIÓN ESTANDAR: {imp_desviacion}\n VARIANZA: {imp_varianza}\n"
            )

            end_time = time.time()
            elapsed_time_ms = (end_time - start_time) * 1000

            print(resultado)
            print("\n")
            execution_time_result = f"Tiempo de ejecución: {elapsed_time_ms:.6f} milisegundos"
            print(execution_time_result)
            with open("StatisticsResults.txt", "w", encoding="utf-8") as file:
                # guardmos los resultados en el archivo
                print(resultado, file=file)
                print("\n", file=file)
                print(execution_time_result, file=file)

    except FileNotFoundError:
        print(f"Error: El archivo '{ruta}' No se encuentra.")
```

In [44]:
```python
#Para fines de observar resultados invocamos el archivo desde una ruta local, posteriormente queda la opción de invocarlo desde consola
impresora("C:/Users/traba/Downloads/TC1.txt")
#impresora("C:\Users\traba\Downloads\TC1.txt")
```

```
Metricas.
 SUMATORIA: 400
 PROMEDIO: 242.32
 MEDIANA: 239.5
 MODA: 393.0
 DESVIACIÓN ESTANDAR: 145.25810683056557
 VARIANZA: 21152.79959899749



Tiempo de ejecución: 0.000000 milisegundos
```

In [30]:

```python
if __name__ == "__main__":
    # Verificamos los parametros en la linea de comando, en caso de estar vacía, solicitamos la información
    if len(sys.argv) != 2:
        print("Introduce la ruta como se muestra: python compute_statistics.py P1/TC2.txt")
        sys.exit(1)

    # obtenemos la ruta de los archivos
    ruta_archivo = sys.argv[1]

    # Invocamos nuestra función principal para imprimir las estadisticas
    impresora(ruta_archivo)
```

```
Introduce la ruta como se muestra: python compute_statistics.py P1/TC2.txt
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
```