```
# info o warstwach w TOC


from PyQt5.QtGui import *
from PyQt5.QtCore import *
from qgis.core import *
from qgis.utils import *
from csv import *

mapa = iface.mapCanvas()
warstwy = mapa.layers()   #lista warstw z TOC
csv_file=open("B:/Python_QGIS/moje/Dane_proby/output.csv",'w')  #otwarcie/utworzenie pliku
csv do zapisu

for ww in warstwy:

    name_war = ww.name()
    #metoda type() obiektu QgsMapLayer zwraca 0 - wektor; 1 - raster
    if ww.type()==1:
        wiersze = ww.height()
        kolumny = ww.width()
        print("Nazwa warstwy = {}, typ = Raster, wierszy = {}, kolumn ={}
        ".format(name_war,wiersze,kolumny))
        wynik_txt = "Nazwa warstwy = {}, typ = Raster, wierszy = {}, kolumn ={}
        ".format(name_war,wiersze,kolumny)
    elif ww.type()==0:
        l_feat = ww.featureCount()  #liczba obiektów w warstwie; metoda featureCount() na
        QgsVectorLayer
        l_ver = 0
        # metoda geometryType() klasy QgsVectorLayer
        if ww.geometryType() == QgsWkbTypes.PolygonGeometry:
            typ_wek = "poligony"
        elif ww.geometryType() == QgsWkbTypes.LineGeometry:
            typ_wek = "linie"
        elif ww.geometryType() == QgsWkbTypes.PointGeometry:
            typ_wek = "punkty"

        #wczytanie pierwszego rekordu by dostać geometrię do określenia Multi czy Single part
        feat_first = ww.getFeature(1)  #uważaj czy jest obiekt o takim id - gdy bylo 0 nie
        dzialalo
        geomF_first = feat_first.geometry()
        MultiPF = geomF_first.isMultipart()
        if MultiPF == True:
            parts = "multipart"
        else:
            parts = "singlepart"

        featIT = ww.getFeatures()  # do iteracji po wszystkich rekordach warstwy

        if typ_wek == "punkty":
            if MultiPF == False:
                l_ver = l_feat
            else:

                for feat in featIT:
                    geomF = feat.geometry()
                    ptG = geomF.asMultiPoint()
                    l_ver = len(ptG)

        elif typ_wek == "linie":

            if MultiPF == False:

                for feat in featIT:
                    geomF = feat.geometry()
                    ptG = geomF.asPolyline()
```

```
                    l_ver = len(ptG)


            else:
                for feat in featIT:
                    geomF = feat.geometry()
                    ptG = geomF.asMultiPolyline()
                    l_parts = len(ptG)
                    l_ver_parts = 0
                    for i in range(l_parts):
                       l_ver_parts = l_ver_parts + len(ptG[i])
                    l_ver = l_ver+ l_ver_parts

        elif typ_wek == "poligony":
            if MultiPF == False:
                for feat in featIT:
                    geomF = feat.geometry()
                    ptG = geomF.asPolygon()
                    l_parts = len(ptG)
                    l_ver_parts = 0
                    for i in range(l_parts):
                       l_ver_parts = l_ver_parts + len(ptG[i])
                    l_ver = l_ver+ l_ver_parts
            else:
                for feat in featIT:
                    geomF = feat.geometry()
                    ptG = geomF.asMultiPolygon()

                    l_parts = len(ptG)
                    l_ver_parts = 0
                    for i in range(l_parts):
                        l_ver_rings = 0
                        l_rings = len(ptG[i])
                        for j in range(l_rings):
                            l_ver_rings = l_ver_rings + len(ptG[i][j])
                        l_ver_parts = l_ver_parts + l_ver_rings
                    l_ver = l_ver + l_ver_parts

        # print(MultiPF,typ_wek,l_feat,l_ver)

        wynik_txt = "Nazwa warstwy = {}, typ = Wektor - {} {}, liczba obiektow = {},liczba
        werteksow = {}\n".format(name_war, typ_wek,parts, l_feat, l_ver)
        print(wynik_txt)
    csv_file.write(wynik_txt)  #wpisanie stringu do pliku

csv_file.close()  # zamknięcie pliku
```