01

INITIATION À PYTHON APPLIQUÉ À LA GÉOMATIQUE

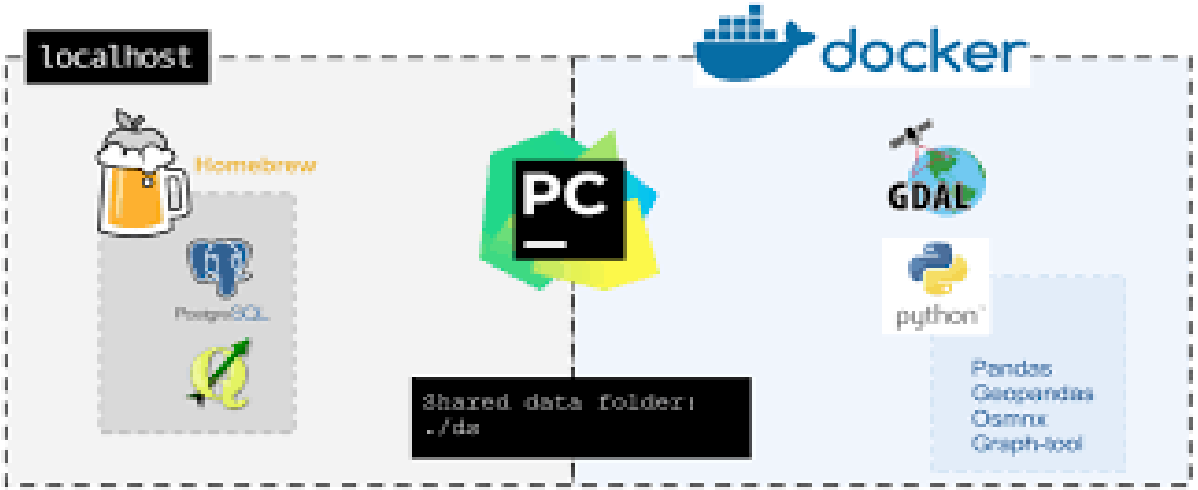
Moumouni Guingarey
Technologie spatiale et Drone,
Fondateur de MGC_SIG Consulting

MGC SIG Consulting

Python

Python est un langage de programmation généraliste interprété de haut niveau!

- ✓ **Années 1980**
- ✓ **Guido van Rossum**



03 Qu’est-ce qu’un programme ?

- Une série d'ordres indiquant à un ordinateur ce qu'il doit faire, pour atteindre un but spécifique

Exemple de programme :

  - un programme pour calculer la distance entre deux coordonnées indiquées par l’utilisateur
  - un programme pour trouver la moyenne des températures mesurées dans une région donnée

- un programme comporte une ou plusieurs séquences d'instructions
- les instructions sont codées (écrites) dans un langage spécifique (langage de programmation)
- l’ensemble d’instructions codées est appelé code source
- le code source est enregistré dans un fichier texte, nommé fichier source

04

Types d'instructions typiques dans un programme ?

- Afficher un message à l'écran
- Lire une entrée au clavier d'un utilisateur
- Lire des données dans un fichier
- Assigner des valeurs à des variables
- Effectuer des calculs
- Ecrire des informations dans un fichier

MGC SIG Consulting

Ce qu'il faut retenir…

Symboles et d'une **syntaxe** utiles pour créer des programmes informatiques

Symboles

- Mots réservés : utilisés pour effectuer des actions spécifiques. Ex : import, return
- Opérateurs : utilisés pour effectuer des opérations sur des données. Ex : +, =
- Symboles de ponctuation : utilisés pour les règles de syntaxes du langage. Ex : ., ( ), ' ' , { }

Syntaxe

Ensemble de règles à suivre lorsqu'on écrit un programme

print("Bonjour MGC !")

variable = input("Bonjour MGC !")

Commentaires

Les commentaires sur une seule ligne

"""

Les chaînes multilignes peuvent être écrites en utilisant trois ", et sont souvent utilisées comme documentation.

"""

```
#Les commentaires sur une seule ligne

"""
Les chaînes multilignes peuvent être écrites en
utilisant trois ", et sont souvent utilisées
comme documentation.
"""
```

Types de données et opérateurs primitifs

Opérateurs primitifs

+, -, =, /

True # => True
False # => False

 # nier avec not
not **True** # => False
not **False** # => True

```
# Les mathématiques sont ce à quoi vous vous attendriez
print(1 + 1)  # => 2
print(8 - 1)  # => 7
print(10 * 2)  # => 20
print(35 / 5)  # => 7.0
```

```
# nier avec not
print(not True)  # => False
print(not False)  # => True
```

MGC SIG Consulting

Les opérateurs

L'égalité est: ==

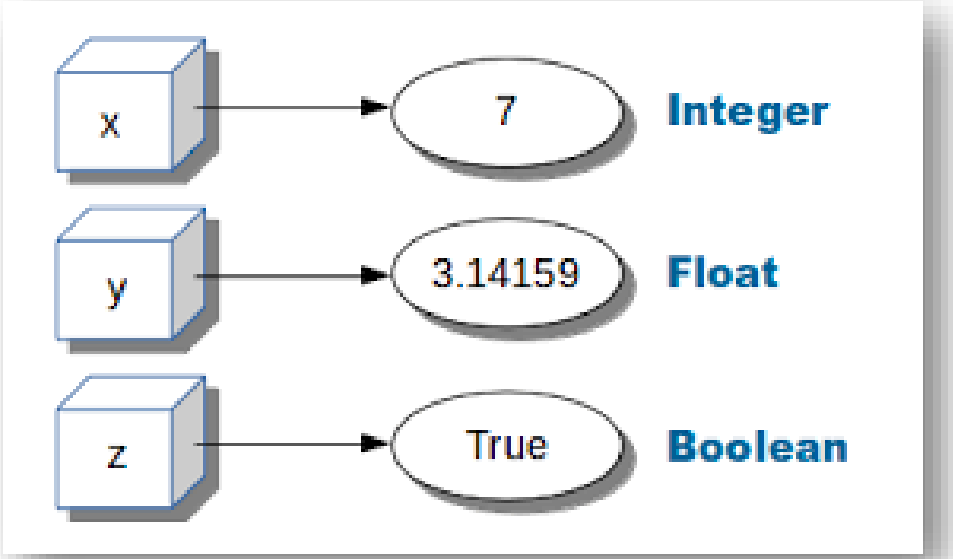
Inférieur : <
Supérieur: >
Inférieur ou égal : <=
Supérieur ou égal : >=

```
# Plus de comparaisons
print(1 < 10)  # => Vrai
print(1 > 10)  # => Faux
print(2 <= 2)  # => Vrai
print(2 >= 2)  # => Vrai
```

Types de données

La fonction type()
Le type int (entier) : 1
Le type float (flottant) : 1.0
Le type str (chaîne de caractères): '1'
Le type bool (booléen): True
Le type list (liste): ['Latitude','Longitude',4329]

La variable




```
x = 75
X = "Longitude"
print(x, X)
```

x = 75 **X = ‘Latitude’**

Chaînes de caractères

 # Les chaînes sont créées avec " ou ‘

" Ceci est une chaîne. "
' Ceci est aussi une chaîne.'

```
# Des chaînes peuvent également être ajoutées
print("Hello " + "world!")
print("Hello "  "world!")
```

Les listes

Une liste en Python est une structure de données qui peut contenir n'importe quel type de données (nombre entier, nombre flottant, chaîne de caractères, liste). Les éléments d'une liste sont rangés et séparés par des virgules.

Le premier élément de la liste L a pour rang 0; il est noté L[0].

Les listes

Les listes stockent les séquences

li = [ ]

 # Vous pouvez commencer avec une liste

préremplie!: other_li = [4 , 5 , 6]

li.pop()

li.insert(index, valeur)

```
li.insert(2, 8)
print(li)
```

```
li = []
# Ajout à la fin d'une ligne
li.append(1)
print(li)
li.append(2)
print(li)
li.append(4)
print(li)
li.append(3)
print(li)
```

Code

https://docs.python.org/fr/3/tutorial/datastructures.html

https://python.doctor/page-apprendre-listes-list-tableaux-tableaux-liste-array-python-cours-debutant

Les listes : Fonctions importantes

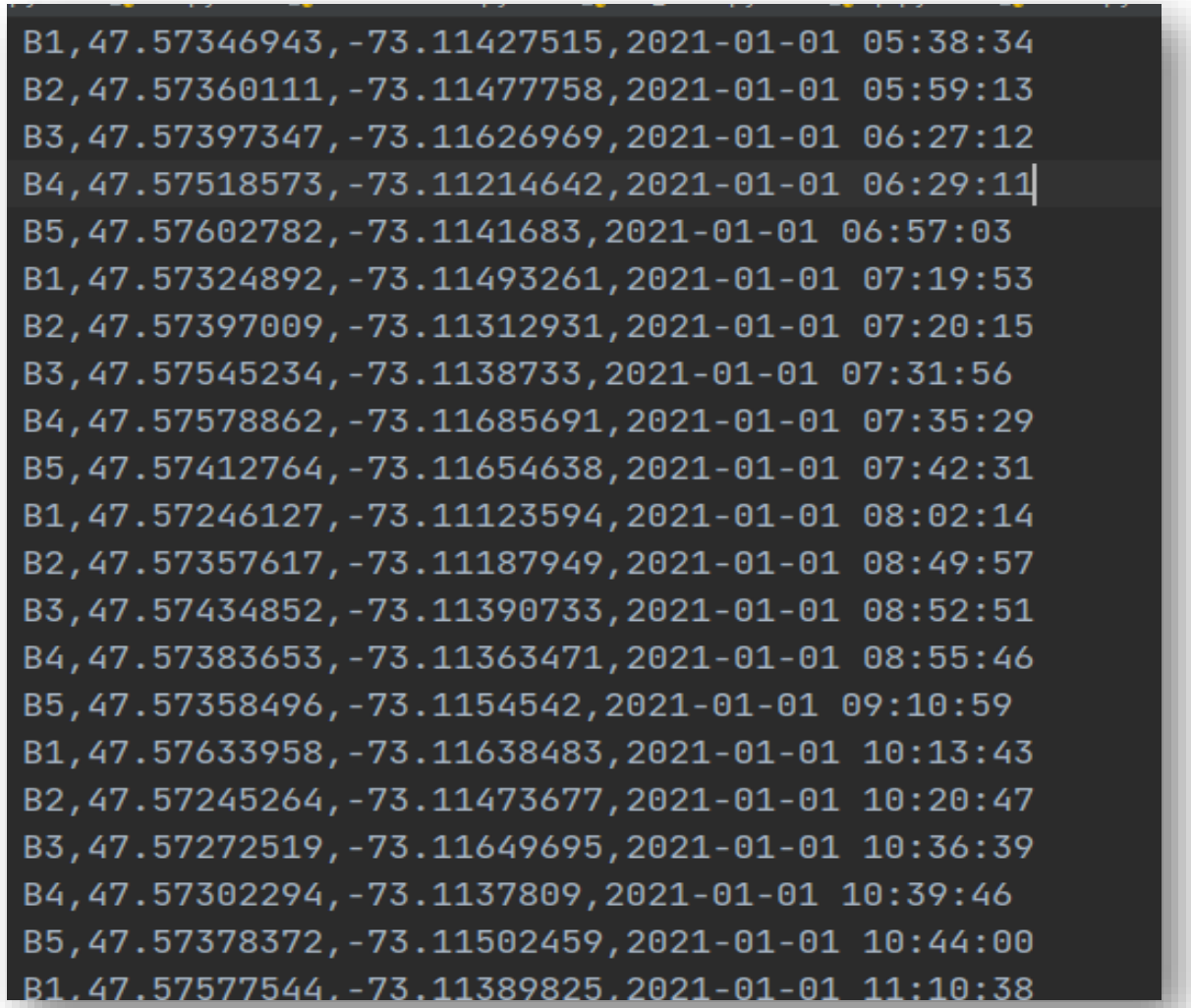
```
append
clear
copy
count
extend
index
insert
pop
remove
reverse
sort
```

Méthodes spécifiques aux listes

L.append(élément)
L.insert(index, valeur)
L.remove(valeur)
L.index(valeur)
L.pop(index)
L.sort()
⇔ del L[index] mais retourne la valeur de l'élément

len(liste)

Les fichiers



```
B1,47.57346943,-73.11427515,2021-01-01 05:38:34
B2,47.57360111,-73.11477758,2021-01-01 05:59:13
B3,47.57397347,-73.11626969,2021-01-01 06:27:12
B4,47.57518573,-73.11214642,2021-01-01 06:29:11
B5,47.57602782,-73.1141683,2021-01-01 06:57:03
B1,47.57324892,-73.11493261,2021-01-01 07:19:53
B2,47.57397009,-73.11312931,2021-01-01 07:20:15
B3,47.57545234,-73.1138733,2021-01-01 07:31:56
B4,47.57578862,-73.11685691,2021-01-01 07:35:29
B5,47.57412764,-73.11654638,2021-01-01 07:42:31
B1,47.57246127,-73.11123594,2021-01-01 08:02:14
B2,47.57357617,-73.11187949,2021-01-01 08:49:57
B3,47.57434852,-73.11390733,2021-01-01 08:52:51
B4,47.57383653,-73.11363471,2021-01-01 08:55:46
B5,47.57358496,-73.1154542,2021-01-01 09:10:59
B1,47.57633958,-73.11638483,2021-01-01 10:13:43
B2,47.57245264,-73.11473677,2021-01-01 10:20:47
B3,47.57272519,-73.11649695,2021-01-01 10:36:39
B4,47.57302294,-73.1137809,2021-01-01 10:39:46
B5,47.57378372,-73.11502459,2021-01-01 10:44:00
B1,47.57577544,-73.11389825,2021-01-01 11:10:38
```

Une grande partie de l'information en géomatique est stockée sous forme de texte dans des fichiers.

Pour traiter cette information, vous devez le plus souvent lire ou écrire dans un ou plusieurs fichiers.

Python possède pour cela de nombreux outils qui vous simplifient la vie.

MGC SIG Consulting

Les fichiers

Lecture dans un fichier:

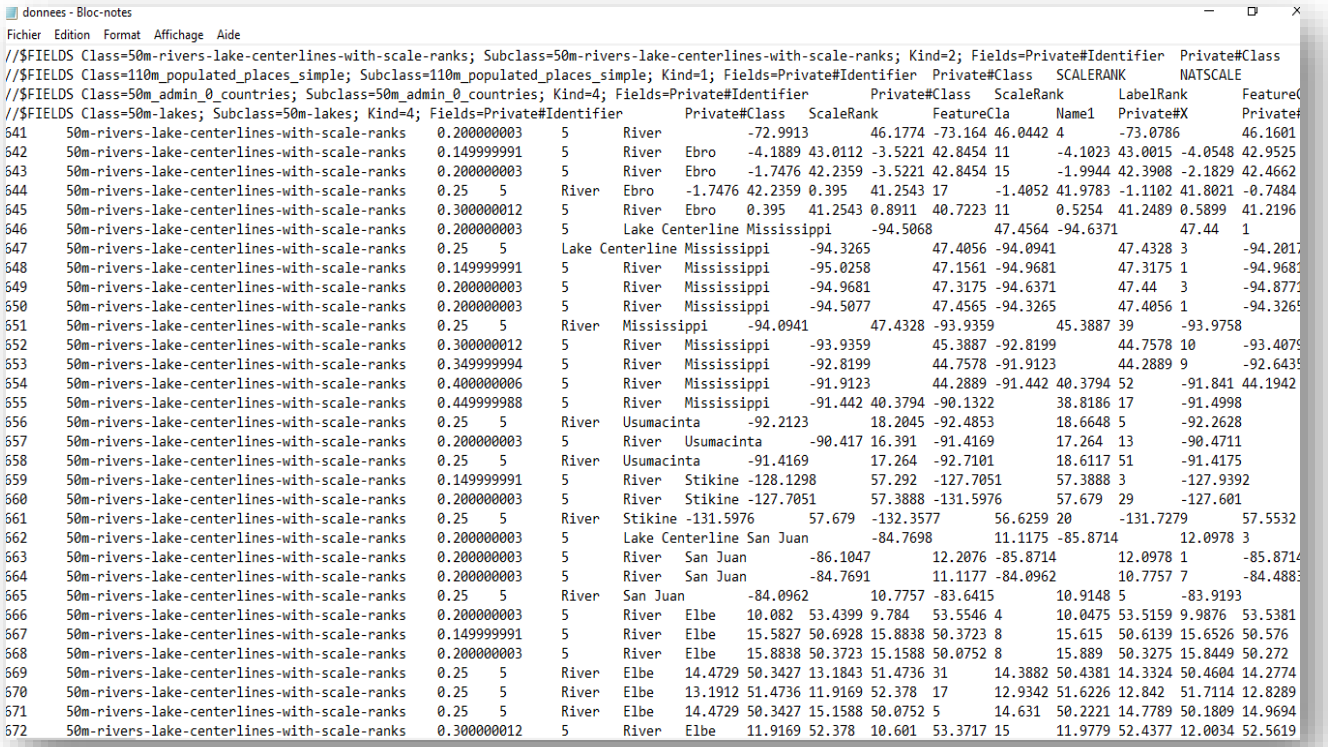
```
coucheRoutes = open('routes.txt','r')

coucheRoutes.close()
```

Écriture dans un fichier:

```
fichierPosition = open('positions.txt','w')

fichierPosition.close()
```


donnees - Bloc-notes

Fichier Edition Format Affichage Aide

```
//$FIELDS Class=50m-rivers-lake-centerlines-with-scale-ranks; Subclass=50m-rivers-lake-centerlines-with-scale-ranks; Kind=2; Fields=Private#Identifier  Private#Class
//$FIELDS Class=110m_populated_places_simple; Subclass=110m_populated_places_simple; Kind=1; Fields=Private#Identifier  Private#Class   SCALERANK       NATSCALE
//$FIELDS Class=50m_admin_0_countries; Subclass=50m_admin_0_countries; Kind=4; Fields=Private#Identifier        Private#Class   ScaleRank       LabelRank       Feature
//$FIELDS Class=50m-lakes; Subclass=50m-lakes; Kind=4; Fields=Private#Identifier        Private#Class   ScaleRank       FeatureCla      Name1   Private#X       Private
641     50m-rivers-lake-centerlines-with-scale-ranks    0.200000003     5       River           -72.9913        46.1774 -73.164 46.0442 4       -73.0786        46.1601
642     50m-rivers-lake-centerlines-with-scale-ranks    0.149999991     5       River   Ebro    -4.1889 43.0112 -3.5221 42.8454 11      -4.1023 43.0015 -4.0548 42.9525
643     50m-rivers-lake-centerlines-with-scale-ranks    0.200000003     5       River   Ebro    -1.7476 42.2359 -3.5221 42.8454 15      -1.9944 42.3908 -2.1829 42.4662
644     50m-rivers-lake-centerlines-with-scale-ranks    0.25    5       River   Ebro    -1.7476 42.2359 0.395   41.2543 17      -1.4052 41.9783 -1.1102 41.8021 -0.7484
645     50m-rivers-lake-centerlines-with-scale-ranks    0.300000012     5       River   Ebro    0.395   41.2543 0.8911  40.7223 11      0.5254  41.2489 0.5899  41.2196
646     50m-rivers-lake-centerlines-with-scale-ranks    0.200000003     5       Lake Centerline Mississippi     -94.5068        47.4564 -94.6371        47.44   1
647     50m-rivers-lake-centerlines-with-scale-ranks    0.25    5       Lake Centerline Mississippi     -94.3265        47.4056 -94.0941        47.4328 3       -94.201
648     50m-rivers-lake-centerlines-with-scale-ranks    0.149999991     5       River   Mississippi     -95.0258        47.1561 -94.9681        47.3175 1       -94.968
649     50m-rivers-lake-centerlines-with-scale-ranks    0.200000003     5       River   Mississippi     -94.9681        47.3175 -94.6371        47.44   3       -94.877
650     50m-rivers-lake-centerlines-with-scale-ranks    0.200000003     5       River   Mississippi     -94.5077        47.4565 -94.3265        47.4056 1       -94.326
651     50m-rivers-lake-centerlines-with-scale-ranks    0.25    5       River   Mississippi     -94.0941        47.4328 -93.9359        45.3887 39      -93.9758
652     50m-rivers-lake-centerlines-with-scale-ranks    0.300000012     5       River   Mississippi     -93.9359        45.3887 -92.8199        44.7578 10      -93.407
653     50m-rivers-lake-centerlines-with-scale-ranks    0.349999994     5       River   Mississippi     -92.8199        44.7578 -91.9123        44.2889 9       -92.643
654     50m-rivers-lake-centerlines-with-scale-ranks    0.400000006     5       River   Mississippi     -91.9123        44.2889 -91.442 40.3794 52      -91.841 44.1942
655     50m-rivers-lake-centerlines-with-scale-ranks    0.449999988     5       River   Mississippi     -91.442 40.3794 -90.1322        38.8186 17      -91.4998
656     50m-rivers-lake-centerlines-with-scale-ranks    0.25    5       River   Usumacinta      -92.2123        18.2045 -92.4853        18.6648 5       -92.2628
657     50m-rivers-lake-centerlines-with-scale-ranks    0.200000003     5       River   Usumacinta      -90.417 16.391  -91.4169        17.264  13      -90.4711
658     50m-rivers-lake-centerlines-with-scale-ranks    0.25    5       River   Usumacinta      -91.4169        17.264  -92.7101        18.6117 51      -91.4175
659     50m-rivers-lake-centerlines-with-scale-ranks    0.149999991     5       River   Stikine -128.1298       57.292  -127.7051       57.3888 3       -127.9392
660     50m-rivers-lake-centerlines-with-scale-ranks    0.200000003     5       River   Stikine -127.7051       57.3888 -131.5976       57.679  29      -127.601
661     50m-rivers-lake-centerlines-with-scale-ranks    0.25    5       River   Stikine -131.5976       57.679  -132.3577       56.6259 20      -131.7279       57.5532
662     50m-rivers-lake-centerlines-with-scale-ranks    0.200000003     5       Lake Centerline San Juan        -84.7698        11.1175 -85.8714        12.0978 3
663     50m-rivers-lake-centerlines-with-scale-ranks    0.200000003     5       River   San Juan        -86.1047        12.2076 -85.8714        12.0978 1       -85.871
664     50m-rivers-lake-centerlines-with-scale-ranks    0.200000003     5       River   San Juan        -84.7691        11.1177 -84.0962        10.7757 7       -84.488
665     50m-rivers-lake-centerlines-with-scale-ranks    0.25    5       River   San Juan        -84.0962        10.7757 -83.6415        10.9148 5       -83.9193
666     50m-rivers-lake-centerlines-with-scale-ranks    0.200000003     5       River   Elbe    10.082  53.4399 9.784   53.5546 4       10.0475 53.5159 9.9876  53.5381
667     50m-rivers-lake-centerlines-with-scale-ranks    0.149999991     5       River   Elbe    15.5827 50.6928 15.8838 50.3723 8       15.615  50.6139 15.6526 50.576
668     50m-rivers-lake-centerlines-with-scale-ranks    0.200000003     5       River   Elbe    15.8838 50.3723 15.1588 50.0752 8       15.889  50.3275 15.8449 50.272
669     50m-rivers-lake-centerlines-with-scale-ranks    0.25    5       River   Elbe    14.4729 50.3427 13.1843 51.4736 31      14.3882 50.4381 14.3324 50.4604 14.2774
670     50m-rivers-lake-centerlines-with-scale-ranks    0.25    5       River   Elbe    13.1912 51.4736 11.9169 52.378  17      12.9342 51.6226 12.842  51.7114 12.8289
671     50m-rivers-lake-centerlines-with-scale-ranks    0.25    5       River   Elbe    14.4729 50.3427 15.1588 50.0752 5       14.631  50.2221 14.7789 50.1809 14.9694
672     50m-rivers-lake-centerlines-with-scale-ranks    0.300000012     5       River   Elbe    11.9169 52.378  10.601  53.3717 15      11.9779 52.4377 12.0034 52.5619
```

Les fichiers

Lecture dans un fichier:

```
coucheRoutes = open('routes.txt','r')

coucheRoutes.readline()        #Lecture de la première ligne
coucheRoutes.readline()        #Lecture de la deuxième ligne
coucheRoutes.readlines()       #Lecture de toutes les lignes

coucheRoutes.close()
```

Écriture dans un fichier:

```
fichierPosition = open('positions.txt','w')
        fichierPosition.write('pays' )          #Ecriture de la première ligne
        fichierPosition.write('continent' )         #Ecriture de la deuxième ligne
        fichierPosition.write('ville' )         #Ecriture de la deuxième ligne
fichierPosition.close()
```

Les fichiers

Lecture dans un fichier:

```
villesPeuple = open('files/villesPeuples.txt', 'r')

ligne1 = villesPeuple.readline()

print(ligne1)  # Lecture de la première ligne
ligne2 = villesPeuple.readline()
print(ligne2)  # Lecture de la deuxième ligne

resteDesLignes = villesPeuple.readlines()
print(resteDesLignes)

villesPeuple.close()
```

Code

Écriture dans un fichier:

```
fichierInformations = open("informations.txt", "w")

fichierInformations.write("pays")  # Ecriture de la première ligne
fichierInformations.write("continent")  # Ecriture de la deuxième ligne
fichierInformations.write("ville")  # Ecriture de la troisième ligne

fichierInformations.close()
```

Code

Les fichiers

Écriture dans un fichier: ‘\n’

```
fichierInformations = open("informations.txt", "w")

fichierInformations.write("pays\n")  # Ecriture de la première ligne
fichierInformations.write("continent\n")  # Ecriture de la deuxième ligne
fichierInformations.write("ville\n")  # Ecriture de la troisième ligne

fichierInformations.close()
```

Code

Les boucles

for i in iterable :
instructions

```
fichierInformations = open("informations.txt", "r")
information = fichierInformations.readlines()
for ligne in information:
    print(ligne)
fichierInformations.close()
```

for k in iterable :
instructions

```
fichierInformations = open("informations.txt", "r")
information = fichierInformations.readlines()
for geomadev in information:
    print(geomadev)
fichierInformations.close()
```

Code

Autre boucle : while

MGC SIG Consulting

- if condition :
  - *instructions 1*
- elif condition:
  - *instructions 2*
- elif condition:
  - *instructions 3*
- else:
  - *instructions*

20

INTRODUCTION À PYTHON POUR ARCGIS

Moumouni Guingarey
Technologie spatiale et Drone,
Fondateur de MGC_SIG
Consulting

MGC SIG Consulting

ARCPY

Définir un espace de travail :

arcpy.env.workspace = r'C:\Users\Lenovo\PycharmProjects\pythonMGC\data\testpython.gdb'

Comment lister des tableaux contenu dans une base :

tableaux = arcpy.ListFeatureClasses()
tableaux

Comment ajouter un champ dans la table :

arcpy.AddField_management('Regions', 'area', 'short', 20, field_alias='superficie')

Comment ajouter un champ dans plusieurs tables :

for table in tableaux :
arcpy.AddField_management(table, 'area', 'short', 20, field_alias='superficie')

ARCPY

Comment multiplier des données :

On crée un autre géodatabase qui servira a accueillir les nouvelles données

On définit un environnement de travail qui contient les données:

arcpy.env.workspace = r'C:\Users\Lenovo\PycharmProjects\pythonMGC\data\testpython.gdb'

On définit une variable de sorti :

outws = r'C:\Users\Lenovo\PycharmProjects\pythonMGC\data\testpython2.gdb'

On crée une variable de recuperation des tables dans le premier géodatabase

Tables = arcpy.ListFeatureClasses()

Tables

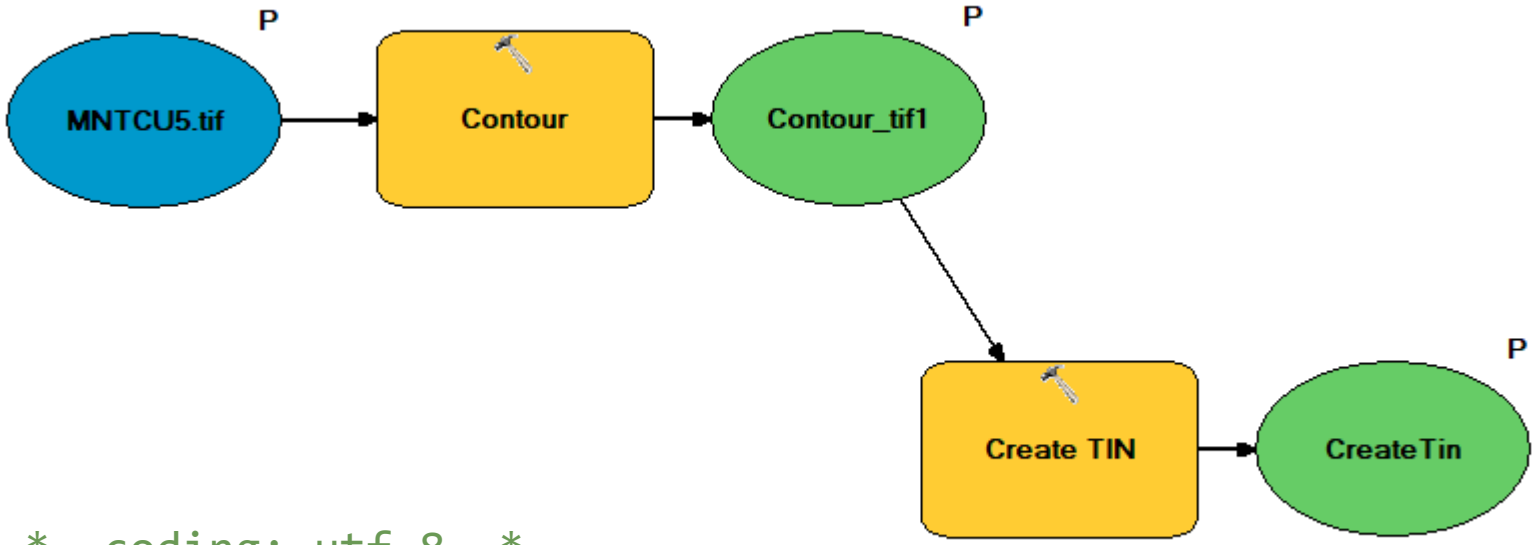
On exporte 100 fois la table vers une autres base :

```
for i in range(0,100):
...     arcpy.FeatureClassToFeatureClass_conversion(Tables[0], outws, Tables[0] + str(i))
```

Exercice

Ajouter un champ a toute les tables crée par la boucle

Model Bulder




```
# -*- coding: utf-8 -*-

# Import arcpy module
import arcpy

# Script arguments
MNTCU5_tif = arcpy.GetParameterAsText(0)
if MNTCU5_tif == '#' or not MNTCU5_tif:
    MNTCU5_tif = "MNTCU5.tif" # provide a default value if unspecified
```

Model Builder

```
Contour_tif1 = arcpy.GetParameterAsText(1)
if Contour_tif1 == '#' or not Contour_tif1:
    Contour_tif1 = "C:\\Users\\Lenovo\\Documents\\ArcGIS\\Default.gdb\\Contour_tif1" # provide
a default value if unspecified

CreateTin = arcpy.GetParameterAsText(2)
if CreateTin == '#' or not CreateTin:
    CreateTin = "C:\\Users\\Lenovo\\Documents\\ArcGIS\\CreateTin" # provide a default value if
unspecified

# Local variables:

# Process: Contour
arcpy.gp.Contour_sa(MNTCU5_tif, Contour_tif1, "10", "0", "1", "CONTOUR", "")

# Process: Create TIN
arcpy.CreateTin_3d(CreateTin, "",
"C:\\Users\\Lenovo\\Documents\\ArcGIS\\Default.gdb\\Contour_tif1 <None> Hard_Line <None>",
"DELAUNAY")
```

END

Guingarey

MERCI !

Moumouni Guingarey

dg@mgcsigconsultingniger.com

Moumouni Guingarey Chamssoudina

- COLLECTES ET ANALYSES DES DONNÉES
- DIGITALISATION-GÉODATAMANATEMENT
- PRODUCTION DE CARTE THEMATIQUE
- SUIVI ET ÉVALUATION ENVIRONNEMENTAL
- FORMATION ET ACCOMPAGNEMENT SCIENTIFIQUE
- INTÉGRATION DES SYSTÈMES SIG
- WEBMAPPING