

Προγραμματισμός Η/Υ

Περιεχόμενα

- 1. Εισαγωγή στον προγραμματισμό
- 2. Τιμές, τύποι και μεταβλητές. Συμβολοσειρές
- 3. Λίστες. Εκφράσεις, τελεστές
- 4. Πλειάδες και λεξικά
- 5. Έλεγχος ροής εκτέλεσης
- 6. Συναρτήσεις (functions)
- 7. Ανάγνωση & εγγραφή αρχείων, μετονομασία, αναζήτηση, αντιγραφή μετακίνηση αρχείων και καταλογών
- 8. Ανάγνωση αρχείων csv, η βιβλιοθήκη pandas
- 9. Γεωεπεξεργασία διανυσματικών δεδομένων
- 10. Διαγράμματα
- 11. Η βιβλιοθήκη numpy
- Indexing and Slicing
- 12. Άσκηση επεξεργασίας και ανάλυσης γεωχωρικών δεδομένων

Τμήμα: Μηχανικών Χωροταξίας, Πολεοδομίας και Περιφερειακής Ανάπτυξης

Τίτλος Επιστημονικού Πεδίου: Στατιστική και Πληροφορική

Κωδικός Μαθήματος: ΜΕ0200

Τίτλος Μαθήματος: Προγραμματισμός Η/Υ

Κατηγορία Μαθήματος: Επιλογής

Εξάμηνο: Εαρινό

Περίγραμμα του μαθήματος

Το μάθημα εισάγει τους φοιτητές στις βασικές έννοιες και αρχές του προγραμματισμού και είναι προσαρμοσμένο στις ανάγκες και στο υπόβαθρο των χωροτακτών. Στόχος του μαθήματος είναι να αποκτήσουν οι φοιτητές τις αναγκαίες γνώσεις για την επεξεργασία και ανάλυση δεδομένων, την αυτοματοποίηση διαδικασιών και την σύνταξη σεναρίων (scripts) για την αναπαραγωγή σιμότητα της ερευνάς τους. Πέρα από τις βασικές αρχές προγραμματισμού η διδασκαλία επεκτείνεται σε εξειδικευμένες θεματικές ενότητες που αφορούν τα γεωχωρικά δεδομένα και την ανάλυσή τους μέσω προγραμματισμού. Η διδασκαλία θα στηριχθεί στην [Python](#), μια σύγχρονη, ευρέως διαδεδομένη και υψηλού επιπέδου γλώσσα προγραμματισμού.

Το πρόγραμμα των διαλέξεων καθώς και η προτεινόμενη βιβλιογραφία παρατίθενται στην συνέχεια.

Αξιολόγηση

Η αξιολόγηση γίνεται:

- 50% από ατομικές ασκήσεις στην διάρκεια του εξαμήνου.
- 50% από τελικές εξετάσεις.

Περιεχόμενα

1. Εισαγωγή στον προγραμματισμό

Ενότητες του μαθήματος

Το μάθημα χωρίζεται στις ακόλουθες ενότητες:

1. Εισαγωγή στον προγραμματισμό

Κατά την διάρκεια της διάλεξης διευκρινίζεται ο σκοπός του μαθήματος και περιγράφονται συνοπτικά οι ενότητες που θα διδαχθούν οι φοιτητές κατά την διάρκεια του εξαμήνου. Διατυπώνονται συγκεκριμένοι ορισμοί που αφορούν τον προγραμματισμό Η/Υ και αναπτύσσονται έννοιες για την επιστήμη των υπολογιστών. Στην συνέχεια εγκαθίσταται στους υπολογιστές των φοιτητών η γλώσσα προγραμματισμού Python μαζί με το απαραίτητο λογισμικό για την συγγραφή και αποσφαλμάτωση του κώδικα. Ακολουθεί εξοικείωση με το περιβάλλον εργασίας.

2. Τιμές, τύποι και μεταβλητές

Περιγραφή της έννοιας των μεταβλητών, των σταθερών, τύποι δεδομένων, εκχώρηση τιμών στις μεταβλητές, κανόνες ονοματοδοσίας των μεταβλητών.

3. Εκφράσεις, τελεστές

Ορισμός εκφράσεων, τι είναι τελεστές, ποια είναι η προτεραιότητα των τελεστών, πως εισάγουμε σχόλια στον κώδικα και γιατί είναι σημαντική πρακτική.

4. Έλεγχος ροής εκτέλεσης

Η λογική Boolean, Εκτέλεση υπό συνθήκη, αλυσιδωτές και εμφωλευμένες συνθήκες, βρόχος και οι εντολές επανάληψης for και while.

5. Συναρτήσεις

Ορισμός και κλήση συνάρτησης, παράμετροι συναρτήσεων, εμβέλεια μεταβλητών, αναδρομή.

6. Συμβολοσειρές/Δομές Δεδομένων

Προσπέλαση συμβολοσειρών, χαρακτήρες διαφυγής, υποσύνολα συμβολοσειράς, συγκρίσεις και ιδιότητες, μέθοδοι συμβολοσειρών. Λίστες, Πλειάδες, Λεξικά.

7. Ανάγνωση & εγγραφή αρχείων, φάκελοι

Ανάγνωση και εγγραφή σε αρχείο, σειριοποίηση (serialization) αντικειμένου, διαχείριση φακέλων και αρχείων.

8. Πίνακες και διαγράμματα

Ανάγνωση αρχείων csv ή excel, pandas dataframes

9. Πίνακες και διαγράμματα

Πίνακες στην βιβλιοθήκη numpy, διαγράμματα με την βιβλιοθήκη seaborn.

10. Γεωεπεξεργασία διανυσματικών δεδομένων

Ανάγνωση και εγγραφή διανυσματικών δεδομένων, μετα-δεδομένα, φιλτράρισμα, αλλαγή προβολικού συστήματος.

11. Ανάλυση διανυσματικών δεδομένων

Χωρικές σχέσεις, στατιστικά ομαδοποιήσεων, οπτικοποίηση διανυσματικών δεδομένων.

12. Γεωεπεξεργασία ψηφιδωτών δεδομένων

Ανάγνωση και εγγραφή διανυσματικών δεδομένων, μετα-δεδομένα, ορισμός μάσκας/αποκοπή περιοχής, αλλαγή τιμών, επαναταξινόμηση, αλλαγή προβολικού συστήματος.

13. Ανάλυση ψηφιδωτών δεδομένων

Αλγεβρα ψηφιδωτών αρχείων, στατιστικά ζωνών, ιστόγραμμα συχνοτήτων.

Ορισμοί

Definition 1

«Αλγόριθμο» ονομάζουμε κάθε πεπερασμένη και αυστηρά καθορισμένη σειρά βημάτων (οδηγιών) για την επίλυση ενός προβλήματος. [Αγγελιδάκης, 2015]

Ένας αλγόριθμος είναι μια αυστηρά καθορισμένη διαδικασία που λαμβάνει μια τιμή ή ένα σύνολο τιμών εισόδου και αποδίδει μια ή περισσότερες τιμές εξόδου. Είναι κατά συνέπεια μια ακολουθία υπολογιστικών βημάτων που μετατρέπει την είσοδο δεδομένων σε έξοδο αποτελεσμάτων [Cormen, 2009].

Για παράδειγμα η αύξουσα (ή φθίνουσα) ταξινόμηση μιας λίστας αριθμών είναι ένα χαρακτηριστικό παράδειγμα αλγορίθμου. Οπότε με τιμές εισόδου {31, 41, 59, 26, 41, 58}, ο αλγόριθμος ταξινόμησης επιστρέφει ως τιμές εξόδου {26, 31, 41, 41, 58, 59}.

Definition 2

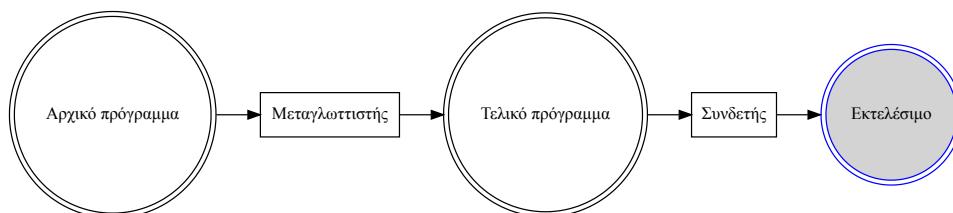
Ως «Πρόγραμμα» ορίζεται ένας αλγόριθμος γραμμένο σε γλώσσα κατανοητή για τον υπολογιστή και περιέχει εντολές (οδηγίες) που κατευθύνουν με κάθε λεπτομέρεια τον υπολογιστή, για να εκτελέσει μια συγκεκριμένη εργασία και να επιλύσει ένα πρόβλημα [Αγγελιδάκης, 2015]. Ένα Πρόγραμμα αναγνώσιμο από τον άνθρωπο ονομάζεται «πηγαίος κώδικας».

Definition 3

«Προγραμματισμός» ονομάζεται η διαδικασία συγγραφής προγραμμάτων και περιλαμβάνει τη διατύπωση των κατάλληλων εντολών προς τον υπολογιστή με τη χρήση τεχνητών γλωσσών, των γλωσσών προγραμματισμού [Αγγελιδάκης, 2015].

Ο προγραμματισμός είναι μια διαδικασία που απαιτεί μια σειρά εργαλείων και διαδικασιών τα οποία συνήθως ενσωματώνονται όλα μαζί σε ένα ενοποιημένο περιβάλλον που αποκαλείται «ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών» (Integrated Development Environment, IDE).

Η διαδικασία μετατροπής του πηγαίου κώδικα σε εκτελέσιμο αρχείο περιγράφεται στο παρακάτω διάγραμμα:



Εικ. 1 Η ροή εκτέλεσης του κώδικα σε εκτελέσιμο.

Τα εργαλεία προγραμματισμού τα οποία κάνουν την μεταγλωτιστή του πηγαίου προγράμματος σε εκτελέσιμο πρόγραμμα είναι τα εξής:

- ο επεξεργαστής κειμένου με την βοήθεια οποίου συντάσσεται ο πηγαίο κώδικάς του προγράμματος.
- ο μεταγλωτιστής ή διερμηνευτής οι οποίοι χρησιμοποιούνται για την μετατροπή του πηγαίου κώδικα σε γλώσσα μηχανής η οποία είναι απαραίτητη για την αναγνώριση και εκτέλεση των εντολών από τον Η/Υ. Τα παραγόμενο πρόγραμμα από την μεταγλώτιση ονομάζεται αντικείμενο πρόγραμμα (object). Ο διερμηνευτής διαβάζει διαδοχικά τις εντολές και για κάθε εντολή που διαβάζει, εκτελεί αμέσως μια ισοδύναμη ακολουθία εντολών μηχανής. Από την άλλη, ο μεταγλωτιστής δέχεται στην είσοδο ένα πρόγραμμα γραμμένο σε γλώσσα υψηλού επιπέδου (πηγαίο κώδικα) και παράγει ισοδύναμο πρόγραμμα σε γλώσσα μηχανής (αντικείμενο).

- ο συνδέτης - φορτωτής (linker- loader) ο οποίος συνδέει το αντικείμενο πρόγραμμα με άλλα τμήματα του προγράμματος ή απαραίτητες βιβλιοθήκες που διατίθεται από την γλώσσα προγραμματισμού. Το τελικό πρόγραμμα που προκύπτει από την μεταγλώτιση και την σύνδεση των τμημάτων του προγράμματος είναι το εκτελέσιμο πρόγραμμα (executable) το οποίο μπορεί να διαβάσει και να εκτελέσει ο υπολογιστής.
- τα εργαλεία αποσφαλμάτωσης με την βοήθεια των οποίων δοκιμάζεται η εκτέλεση και η ορθότητα του πηγαίου κώδικα και εντοπίζονται λάθη σε αυτόν.

Τα λάθη στον κώδικα συνοψίζονται σε τρείς βασικές κατηγορίες:

1. **σφάλμα μεταγλώπτισης** τα οποία προκύπτουν κατά την λανθασμένη συγγραφή του πηγαίου κώδικα. Ο μεταγλωπτιστής δεν επιτρέπει την μετάφραση του πηγαίου κώδικα σε γλώσσα μηχανής αν προηγουμένος δεν έχει διορθωθεί το συντακτικό λάθος. Συντακτικά λάθη συμβαίνουν συνήθως όταν δεν ακολοθούνται οι κανόνες σύνταξης μια γλώσσας (π.χ. μια παρένθεση που δεν έχει κλείσει, ένα ξεχασμένο εισαγωγικό ή κόμμα κτλ.).

Το παρακάτω είναι ένα παράδειγμα συντακτικού σφάλματος και το μήνυμα που επιστρέφει ο μεταγλωπτιστής της Python. Η αιτία του σφάλματος είναι η ξεχασμένη παρένθεση στην συνάρτηση (function) `print`

```
print("an example")
```

```
Input In [2]
    print("an example"
      ^
SyntaxError: '(' was never closed
```

1. **σφάλμα εκτέλεσης** (run-time errors) τα οποία συμβαίνουν κατά την εκτέλεση του προγράμματος παρότι δεν υπάρχουν σφάλματα σύνταξης. Χαρακτηριστικά παραδείγματα τέτοιων λαθών είναι η διαίρεση με το μηδέν, η πρόσβαση σε ένα στοιχείο μιας λίστας εκτός του εύρους της, η ανάγνωση ενός αρχείου το οποίο δεν υπάρχει, ή η πρόσβαση σε ένα ανύπαρκτο object. Τα σφάλματα εκτέλεσης έχουν επικρατήσει να αναφέρονται και ως «bugs» [1]. Παρακάτω δίνεται ένα σφάλμα που προκύπτει από την διαίρεση ενός ακέραιου με το μηδέν.

```
1/0
```

```
-----
ZeroDivisionError                         Traceback (most recent call last)
Input In [3], in <cell line: 1>()
----> 1 1/0

ZeroDivisionError: division by zero
```

1. **σφάλμα λογικής**, κατά το οποίο το πρόγραμμα εκτελείται κανονικά χωρίς σφάλματα αλλά δεν συμπεριφέρεται όπως έχει σχεδιαστεί να συμπεριφέρεται. Αυτά τα σφάλματα δεν σταματούν την εκτέλεση του προγράμματος αλλά το αποτέλεσμα της εκτέλεσης δεν είναι το αναμενόμενο.

```
x = 6
y = 4

z = x+y/2
print('Ο μέσος όρος των δύο αριθμών είναι:',z)
```

```
Ο μέσος όρος των δύο αριθμών είναι: 8.0
```

Το παραπάνω είναι σφάλμα λογικής γιατί έπρεπε να γραφτεί ως εξής (δώστε προσοχή στις παρενθέσεις που δίνουν προτεραιότητα στις πράξεις):

```
x = 6
y = 4

z = (x+y)/2
print('Ο μέσος όρος των δύο αριθμών είναι:',z)
```

```
Ο μέσος όρος των δύο αριθμών είναι: 5.0
```

Όλες οι παραπάνω μορφές σφαλμάτων εντοπίζονται μέσω της αποσφαλμάτωσης, της συστηματικής δηλαδή διαδικασίας εντοπισμού και επιδιόρθωσης σφαλμάτων. Η αποσφαλμάτωση συνοψίζεται στα εξής βήματα:

- Επανάληψη του προβλήματος
- Απομόνωση του σημείου που εμφανίζεται το σφάλμα
- Αναγνώριση της αιτίας που το προκαλεί
- Διόρθωση του σφάλματος

- Επιβεβαίωση της διόρθωσης

Οι εντολές των προγραμμάτων γράφονται από τους προγραμματιστές σε τεχνητές γλώσσες που ονομάζονται «γλώσσες προγραμματισμού». Μια γλώσσα προγραμματισμού θα πρέπει να έχει αυστηρά ορισμένη σύνταξη και σημασιολογία. Η σύνταξη καθορίζει αν μια σειρά από σύμβολα αποτελούν «νόμιμες» εντολές ενός προγράμματος γραμμένου σε μια συγκεκριμένη γλώσσα προγραμματισμού και η σημασιολογία καθορίζει τη σημασία του προγράμματος, δηλαδή τις υπολογιστικές διαδικασίες που υλοποιεί. [\[Αγγελιδάκης, 2015\]](#).

Η γλώσσα προγραμματισμού Python

Η Python είναι μια ευρέως διαδομένη, αντικειμενοστραφής, υψηλού επιπέδου γλώσσα προγραμματισμού γενικής χρήσης. Η Python είναι μια γλώσσα που εκτελεί τις εντολές στον διερμηνέα, που όπως αναφέρθηκε, διαβάζει τον πηγαίο κώδικα γραμμή προς γραμμή και το μετατρέπει σε γλώσσα μηχανής. Αυτός ο τρόπος λειτουργίας της Python την καθιστά πιο αργή σε σύγκριση με άλλες γλώσσες μεταγλωπιστού όπως η C. Η Python είναι διαδραστική υπό την έννοια ότι ο χρήστης εκτελεί εντολές μέσω της γραμμή εντολών της Python, εκτελείται άμεσα και λαμβάνει το αποτέλεσμα εξόδου.

Δημιουργήθηκε από τον Guido van Rossum και πρωτοκυκλοφόρησε στις 20 Φεβρουαρίου του 1991. Το όνομά της, αν και παρεπέμπει, δεν έχει σχέση με το φίδι Πύθωνα αλλά προέρχεται από την γνωστή κωμική σειρά του BBC, Monty Python's Flying Circus. Αν και αρχικά αναπτύχθηκε σαν μεμονωμένη ατομική προσπάθεια στην συνέχεια υποστηρίχθηκε από μια παγκόσμια κοινότητα προγραμματιστών και χρηστών. Στις 6 Μαρτίου 2001 ιδρύθηκε το αμερικανικό μη κερδοσκοπικό ίδρυμα *Python Software Foundation (PSF)*, το οποίο στόχο έχει την διάδοση και υποστήριξη της Python μέσω της διοργάνωσης συνεδρίων, την ανάπτυξη κοινοτήτων χρηστών, την υποστήριξη προσπαθειών μέσω υποτροφιών και την διασφάλιση οικονομικών πόρων για την ανάπτυξη της γλώσσας. Το ίδρυμα κατέχει τα πνευματικά δικαιώματα της γλώσσας και διασφαλίζει ότι αυτή θα διατίθεται με όρους ελεύθερου λογισμικού προς το ευρύτερο κοινό.

Οι βασικοί στόχοι που έθεσε ο δημιουργός κατά την ανάπτυξή της είναι να είναι εύκολη και κατανοητή με ισχυρές δυνατότητες εφάμιλλες των ανταγωνιστικών γλωσσών. Ταυτόχρονα έθεσε το όρο να είναι ανοιχτού κώδικα (*open source*) για να μπορεί εύκολα να αναπτύσσεται από τους ενδιαφερόμενους προγραμματιστές και να έχει πρακτική αξία σε καθημερινές εργασίες ρουτίνας. Την τρέχουσα περίοδο (02/2023) κατατάσσεται ως η κορυφαία γλώσσα προγραμματισμού σύμφωνα με την κοινότητα προγραμματιστών [TIOBE](#) αλλά και τον δείκτη [PopularitY of Programming Language Index \(PYPL\)](#).

Feb 2023	Feb 2022	Change	Programming Language	Ratings	Change
1	1		Python	15.49%	+0.16%
2	2		C	15.39%	+1.31%
3	4	▲	C++	13.94%	+5.93%
4	3	▼	Java	13.21%	+1.07%
5	5		C#	6.38%	+1.01%
6	6		Visual Basic	4.14%	-1.09%
7	7		JavaScript	2.82%	+0.70%
8	10	▲	SQL	2.12%	+0.58%
9	9		Assembly language	1.38%	-0.21%
10	8	▼	PHP	1.29%	-0.49%
11	11		Go	1.11%	-0.12%
12	13	▲	R	1.08%	-0.04%
13	14	▲	MATLAB	0.99%	-0.04%
14	15	▲	Delphi/Object Pascal	0.95%	+0.05%
15	12	▼	Swift	0.93%	-0.25%
16	16		Ruby	0.83%	-0.06%
17	19	▲	Perl	0.79%	-0.01%
18	22	▲	Scratch	0.76%	+0.13%
19	17	▼	Classic Visual Basic	0.74%	-0.09%
20	24	▲	Rust	0.70%	+0.16%

Εικ. 2 Η κατάταξη σύμφωνα με την κοινότητα TIOBE (02/2023)

Η Python πλέον είναι μια ώριμη γλώσσα προγραμματισμού με εφαρμογές στην ανάπτυξη διαδικτυακών εφαρμογών και υπηρεσιών, την εκπαίδευση, την ανάλυση δεδομένων, την τηλεπισκόπηση και τα ΣΓΠ, την δημιουργία γραφικών, την διαχείριση συστημάτων, τα παιχνίδια, το εμπόριο και την επιχειρηματικότητα, τους μικροελεγκτές και το Internet of Things (IOT).

Η φιλοσοφία της Python ως προς την μεθοδολογία ανάπτυξης και προγραμματισμού συνοψίζεται σε 20 αρχές, οι οποίες εκτυπώνονται μέσω της γλώσσας με την παρακάτω εντολή:

```
import this
```

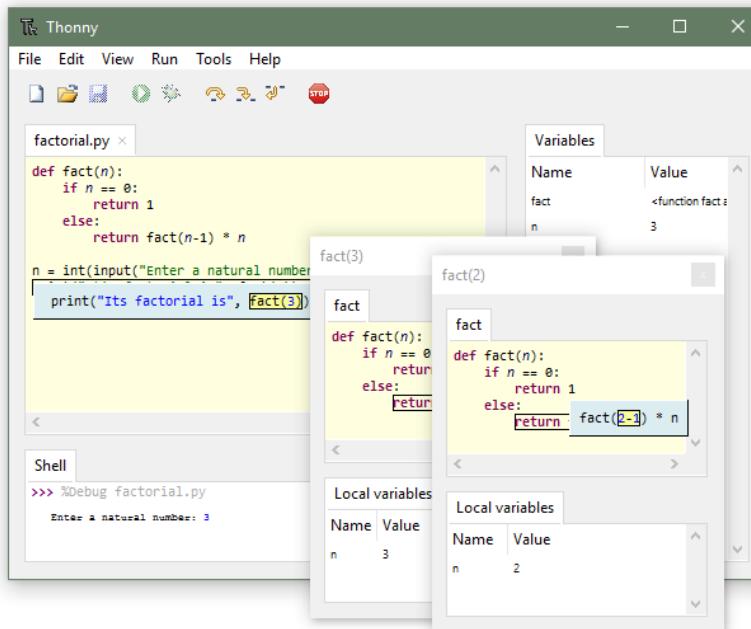
```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Το ολοκληρωμένο περιβάλλον ανάπτυξης thonny

Η συγγραφή κώδικα θα γίνει στο ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment, IDE) [thonny](#). Δεν απαιτείται η προεγκατάσταση της Python καθώς το thonny έρχεται με ενσωματωμένη την γλώσσα προγραμματισμού Python 3.7 και διατίθεται για Windows, Mac και Linux. Το thonny αποτελεί εκπαιδευτικό περιβάλλον για την συγγραφή και αποσφαλμάτωση κώδικα Python. Για τον λόγο αυτό διαθέτει πολύ συγκεκριμένες αλλά ζωτικές λειτουργίες και δεν είναι επιφορτισμένο με δυνατότητες που απαιτούνται από προχωρημένους προγραμματιστές. Το Γραφικό Περιβάλλον Χρήστη (GUI, Graphical User Interface) είναι λιτό ώστε να μην αποπροσανατολίζει τον αρχάριο χρήστη. Το thonny παρέχει βοηθητικές λειτουργίες για τον χρήστη όπως είναι η σήμανση συντακτικών λαθών, η αυτόματη συμπλήρωση κώδικα και η ευκολία στην επέκταση των λειτουργίων της Python με την εγκατάσταση συμπληρωματικών πακέτων.

Ενναλακτικά, στους χρήστες παρέχεται online περιβάλλον ανάπτυξης που βασίζεται στο [Jupyter Lab](#). Η χρήση του δεν απαιτεί την εγκατάσταση λογισμικού παρά μόνον έναν απλό φιλομετρητή (προτείνεται Chrome, Safari ή Firefox). Το online περιβάλλον Jupyter είναι προσβάσιμο από εδώ: <https://kokkytos.github.io/programming>



Εικ. 3 Το περιβάλλον εργασία thonny.

Εκτέλεση εντολών στο περιβάλλον thonny

Στην παρακάτω ενότητα παρουσιάζονται μερικά εισαγωγικά παραδείγματα από εντολές της Python. Δεν θα επικεντρωθούμε σε λεπτομέρειες ούτε θα αναλύσουμε τις εντολές που διατυπώνονται στα αρχεία. Παρουσιάζονται σαν μια μορφή συνοπτικής επίδειξης των δυνατοτήτων που διαθέτει η γλώσσα και θα περιγράψουμε σε επόμενα μαθήματα.

Δοκιμάστε να τρέξετε την παρακάτω εντολή στην γραμμή εντολών της Python στο thonny:

```
25+30
```

```
55
```

Τι παρατηρείτε; Η Python λειτούργησε σαν μια απλή αριθμομηχανή.

Στην συνέχεια δοκιμάστε να τρέξετε την παρακάτω εντολή γραμμή προς γραμμή:

```
number1 = 25
number2 = 30
number3 = number1+number2
number3
```

```
55
```

Το αποτέλεσμα είναι το ίδιο με το προηγούμενο.

Δώστε την παρακάτω εντολή. Αντικαταστήστε την συμβολοσειρά *AnyName* με το ονομά σας.

```
name="AnyName"
for i in range(10):
    print("Εκτύπωση", i, ":", name)
```

```
Εκτύπωση 0 : AnyName
Εκτύπωση 1 : AnyName
Εκτύπωση 2 : AnyName
Εκτύπωση 3 : AnyName
Εκτύπωση 4 : AnyName
Εκτύπωση 5 : AnyName
Εκτύπωση 6 : AnyName
Εκτύπωση 7 : AnyName
Εκτύπωση 8 : AnyName
Εκτύπωση 9 : AnyName
```

Όπως βλέπετε η Python επανέλαβε την εκτύπωση του ονόματός σας 10 φορές. Από που ξεκινά όμως η αρίθμηση της πρώτης εκτύπωσης;

Στο επόμενο παράδειγμα η Python θα σας ενημερώσει αν τρέχετε γρήγορα ή αργά ή αν είστε ακίνητος:

```
speed=70
if speed>50:
    if speed>=100:
        print("Τρέχεις πολύ γρήγορα")
    else:
        print("Τρέχεις γρήγορα")
else:
    if speed==0:
        print("Είσαι ακίνητος".upper())
    if speed>0:
        print("Τρέχεις αργά")
```

```
Τρέχεις γρήγορα
```

Έστω ότι κινείσθε σε αυτοκινητόδρομο με 120 km/h. Αν ορίσετε την ταχύτητα (speed) στον κώδικα, τι θα σας απαντήσει η Python; Αν κινείστε με μηδενική ταχύτητα (speed=0) τι μήνυμα θα λάβετε; Υπάρχουν περιπτώσεις που η Python, και δικαιολογημένα, αγνοεί να απαντήσει με μήνυμα στην ταχύτητα που ορίζεται. Μπορείτε να εντοπίσετε σε ποιές περιπτώσεις;

[1] Η πρώτη περίπτωση *bug* σε υπολογιστή καταγράφεται το 1947 από τον Grace Murray Hopper και πρόκειται για την κυριολεκτική έννοια του όρου. Στο ημερολόγιό του καταγράφει προβλήματα στην λειτουργία του υπολογιστή του Harvard, Mark II, από την ύπαρξη ενός εντόμου στο εσωτερικό του κύκλωμα.

2. Τιμές, τύποι και μεταβλητές. Συμβολοσειρές

Σταθερές (Constants)

Η Python δεν διαθέτει προκαθορισμένες σταθερές όπως άλλες γλώσσες προγραμματισμού. Όμως κατά σύμβαση και όχι κατά κανόνα έχει συμφωνηθεί οι σταθερές να ονοματίζονται με κεφαλαίους χαρακτήρες. Η αδυναμία της Python στην περίπτωση της δήλωσης σταθερών είναι ότι επιτρέπεται η αλλαγή των τιμών τους Παρακάτω παρατίθεται ένα παράδειγμα δήλωσης σταθερών.

```
RATIO_FEET_TO_METERS = 3.281
RATIO_LB_TO_KG = 2.205
PI = 3.14
```

Κυριολεκτικές σταθερές (literal constants)

Η κυριολεκτική σταθερά ή τιμή είναι ένας αριθμός, ή χαρακτήρας ή μά συμβολοσειρά. Για παράδειγμα τα παρακάτω αποτελούν τιμές: 3.25 (στην python η υποδιαστολή ορίζεται με . και όχι ,), «ένα τυχαίο κείμενο», 5.25e-1. Αυτές οι τιμές δεν μεταβάλλονται κατά τη διάρκεια εκτέλεσης του προγράμματος γι” αυτό και λέγονται σταθερές. Μπορούν να εκχωρηθούν σε μεταβλητές και να χρησιμοποιηθούν σαν τελεστέοι σε λογικές εκφράσεις ή σαν παραμέτροι σε συναρτήσεις.

Τύποι δεδομένων

Οι τιμές ανήκουν σε τρεις τύπους δεδομένων (data types) ή κλάσσεις (class):

- τους ακέραιους αριθμούς (integer) π.χ. το 15
- τους αριθμούς κινητής υποδιαστολής (floating point) π.χ. το 201.25)
- τις συμβολοσειρές (string) π.χ. το «Time is money»

Με την εντολή `type` ο διερμηνευτής μας απαντάει με τον τύπο της τιμής, όπως παρακάτω:

```
type("No news, good news.")
```

```
str
```

Η Python είναι *Dynamic typing* δηλαδή δεν ο τύπος των μεταβλητών δεν προκαθορίζεται κατά την συγγραφή αλλά κατά την εκτέλεση.

Κανόνες ονοματοδοσίας μεταβλητών

Τα ονόματα των μεταβλητών στην Python υπακούουν στους παρακάτω κανόνες:

- Το όνομα μίας μεταβλητής μπορεί να ξεκινά από ένα γράμμα ή από κάτω πάυλα.
- Το όνομα μίας μεταβλητής δεν μπορεί με αριθμό.
- Το όνομα μίας μεταβλητής μπορεί να περιέχει μόνο αλφαριθμητικούς χαρακτήρες.
- Στα ονόματα των μεταβλητών γίνεται διάκριση ανάμεσα σε πεζά και κεφαλαία (case sensitive).
- Οι δεσμευμένες λέξεις της Python (keywords) δεν μπορούν να χρησιμοποιηθούν σε ονόματα μεταβλητών.

Συμβολοσειρές (Strings)

Μια συμβολοσειρά είναι μια ακολουθία από χαρακτήρες όπως το "Το πεπρωμένον φυγείν αδύνατον.". Μπορεί να είναι σε κάθε γλώσσα που υποστηρίζεται από το πρώτυπο Unicode. Οι συμβολοσειρές περικλείονται σε μονά, διπλά ή τριπλά εισαγωγικά. Με τριπλά εισαγωγικά μπορούν να ενσωματωθούν με ευκολία συμβολοσειρές σε πολλές γραμμές και πολλαπλά εισαγωγικά εντός αυτόν. Ακολουθούν παραδείγματα συμβολοσειρά.

Χαρακτήρες διαφυγής, κενά, νέες γραμμές

Μπορούμε να σπάσουμε μια συμβολοσειρά κατά την συγγραφή σε νέα γραμμή με τον χαρακτήρα \ και κατά την εκτέλεση με τον χαρακτήρα \n π.χ.

```
message = 'There is no smoke \
without fire'

print(message)
```

```
There is no smoke without fire
```

```
message = 'There is no smoke \nwithout fire'

print(message)
```

```
There is no smoke
without fire
```

Ή να ορίσουμε κενά με το `\t`

```
message = 'There is no smoke \twithout fire'

print(message)
```

```
There is no smoke      without fire
```

Ο χαρακτήρας `\v` είναι χαρακτήρας διαφυγής που απενεργοποιεί την ειδική λειτουργία των παραπάνω ή την παράθεση εισαγωγικών μεσα σε εισαγωγικά.

```
print('There is no smoke \\n without fire')
```

```
There is no smoke \n without fire
```

```
print('Where there\v's a will, there\v's a way')
```

```
Where there's a will, there's a way
```

Ανεπεξέργαστες συμβολοσειρές (Raw Strings)

Παρόμοιο αποτέλεσμα με τα παραπάνω πετυχαίνουμε τις ανεπεξέργαστες συμβολοσειρές οι οποίες ορίζονται με ένα `r` σαν πρόθεμα

```
print(r"It was made by \n συνέχεια")
```

```
It was made by \n συνέχεια
```

Αφαίρεση κενών

Σε αρκετές περιπτώσεις οι συμβολοσειρές περιέχουν κενά είτε στην αρχή είτε στο τέλος. Για παράδειγμα οι παρακάτω συμβολοσειρές δεν είναι το ίδιες για την Python. Και επιβεβαιώνεται σε μέσω ελέγχου ισότητας.

```
departmentA='ΤΜΧΠΑ'
departmentB = ' ΤΜΧΠΑ '
print(departmentA == departmentB) #not equal
```

```
False
```

Για την αφαίρεση των κένων αριστερά, δεξιά ή ταυτόχρονα και στις δύο πλευρές της συμβολοσειράς χρησιμοποιούμε την μέθοδο `strip` και τις παραλλαγές της `rstrip` και `lstrip`

```
print(departmentB.rstrip())
print(departmentB.lstrip())
print(departmentB.strip())
```

Συνένωση (Concatenation) συμβολοσειρών

Η απλή παράθεση συμβολοσειρών οδηγεί στην συνένωσή τους δηλ.

```
message = "Curiosity " "killed " 'the' ''cat''  
print(message)
```

```
Curiosity killed the cat
```

Συνένωση συμβολοσειρών και μεταβλητών

Η συνένωση μεταβλητών και συμβολοσειρών γίνεται με τον τελεστή `+`.

```
city='Βόλος'  
perifereia='Θεσσαλία'  
  
print('Ο '+city+' είναι πόλη της Ελλάδα στην ' +perifereia)
```

```
Ο Βόλος είναι πόλη της Ελλάδα στην Θεσσαλία
```

Η μέθοδος `format`

Άλλη μια πιο πρακτική μέθοδος κατά την συνένωση μεταβλητών και συμβολοσειρών είναι η μέθοδος `format`.

```
print('Ο {0} έχει υψόμετρο {1} μέτρα'.format("Ολυμπος", 2918))  
print('Ο {0} έχει υψόμετρο {1} μέτρα'.format("Ολυμπος", 2918))  
print('Ο {name} έχει υψόμετρο {height} μέτρα'.format(name="Σμόλικας", height= 2637))
```

```
Ο Όλυμπος έχει υψόμετρο 2918 μέτρα  
Ο Όλυμπος έχει υψόμετρο 2918 μέτρα  
Ο Σμόλικας έχει υψόμετρο 2637 μέτρα
```

Δεσμευμένες λέξεις (reserved words)

Ορισμένες λέξεις έχουν ιδιαίτερη σημασία για την Python και δεν μπορούν να χρησιμοποιηθούν σαν ονόματα μεταβλητών. Τα παρακάτω κομμάτια κώδικα θα εκδηλώσουν σφάλμα μεταγλώπισης.

Πρόκειται για 33 λέξεις στην τρέχουσα έκδοση της Python. Μπορούμε να δούμε ποιές είναι αυτές οι δεσμευμένες λέξεις με την παρακάτω εντολή:

```
help("keywords")
```

```
Here is a list of the Python keywords. Enter any keyword to get more help.
```

False	class	from	or
None	continue	global	pass
True	def	if	raise
and	del	import	return
as	elif	in	try
assert	else	is	while
async	except	lambda	with
await	finally	nonlocal	yield
break	for	not	

Η εντολή `help`

Γενικά με την εντολή `help` καλούμε για βοήθεια και πληροφορίες την Python:

```
help(print)
```

```
Help on built-in function print in module builtins:  
  
print(...)  
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)  
  
    Prints the values to a stream, or to sys.stdout by default.  
    Optional keyword arguments:  
        file: a file-like object (stream); defaults to the current sys.stdout.  
        sep:   string inserted between values, default a space.  
        end:   string appended after the last value, default a newline.  
        flush: whether to forcibly flush the stream.
```

```
help(abs)
```

```
Help on built-in function abs in module builtins:  
  
abs(x, /)  
    Return the absolute value of the argument.
```

```
help(max)
```

```
Help on built-in function max in module builtins:  
  
max(...)  
    max(iterable, *, default=obj, key=func) -> value  
    max(arg1, arg2, *args, *, key=func) -> value  
  
    With a single iterable argument, return its biggest item. The  
    default keyword-only argument specifies an object to return if  
    the provided iterable is empty.  
    With two or more arguments, return the largest argument.
```

Αλλαγή Πεζών Κεφαλαίων (Convert case)

Μπορούμε να κάνουμε αλλαγή ανάμεσα σε κεφαλαία και πεζά με τις παρακάτω μεθόδους συμβολοσειρών: `upper()`, `title()`, `lower()`. Αξίζει να σημειώσουμε ότι οι μέθοδοι αυτές δεν έχουν επίδραση στην μεταβλητή που τις καλούμε αλλά πρέπει να επαναεκχωρήσουμε το αποτέλεσμα της μεθόδου στην μεταβλητή με το ίδιο όνομα.

```
agios="άγιος νικόλαος"  
  
print(agios.upper())  
print(agios) # ο agios παραμένει "άγιος νικόλαος"  
  
print(agios.title())  
print('ΑΓΙΑ ΕΛΕΝΗ'.lower())  
  
agios = agios.upper()  
print(agios) # ο agios μετά την εκχώρηση στην ίδια μεταβλητή γινεται ΑΓΙΟΣ  
ΝΙΚΟΛΑΟΣ
```

```
ΑΓΙΟΣ ΝΙΚΟΛΑΟΣ  
άγιος νικόλαος  
Άγιος Νικόλαος  
αγία ελένη  
ΑΓΙΟΣ ΝΙΚΟΛΑΟΣ
```

Οι συμβολοσειρές είναι μη μεταβαλλόμενη δομή δεδομένων

Οι συμβολοσειρές αποτελούνται από ακολουθίες χαρακτήρων με σταθερό μέγεθος και μη μεταβαλλόμενα περιεχόμενα. Αυτό σημαίνει ότι δεν είναι δυνατόν να προστίθενται ή να αφαιρούνται χαρακτήρες, ούτε να τροποποιούνται τα περιεχόμενα του αλφαριθμητικού. Πρόκειται για μια μη μεταβαλλόμενη (immutable) δομή της Python. Η αρίθμηση των χαρακτήρων σε ένα αλφαριθμητικό ξεκινάει από το 0.

Έτσι στην συμβολοσειρά `country = Ελλάδα` έχουμε:

```
country[0] → Ε (η αρίθμηση ξεκινά από το 0)
```

```
country[1] → λ
```

```
country[2] → λ
```

```
country[3] → á
```

```
country[4] → δ
```

```
country[5] → α
```

Η παραπάνω συμβολοσειρά έχει μήκος 6 χαρακτήρες.

Μήκος συμβολοσειράς

Μέσω της συνάρτησης `len` η Python μας επιστρέφει το μήκος συμβολοσειράς δηλαδή το πλήθος των χαρακτήρων (μαζί με τα κενά) από τους οποιούς αποτελείται.

```
message = 'Η τώρα ή ποτέ.'  
len(message)
```

14

Η μέθοδος `find`

Η μέθοδος `find` μας επιτρέπει να αναζητήσουμε μια συμβολοσειρά μέσα σε μια άλλη συμβολοσειρά. Η μέθοδος μας επιστρέφει την τοποθεσία από την ξεκινάει η αναζητούμενη συμβολοσειρά δηλαδή τον δείκτη (`index`) στην οποία εντοπίζεται ο πρώτος χαρακτηρας της αναζητούμενης συμβολοσειράς μέσα στα περιεχόμενα της αρχικής συμβολοσειράς. Στην παρακάτω συμβολοσειρά θα αναζητήσουμε την λέξη `ποτέ`.

```
stixos = 'Η Ελλάδα ποτέ δεν πεθαίνει'  
index = stixos.find('ποτέ')
```

Κανονικά αν πάμε στον χαρακτήρα με ευρετηρίο (`index`) 9 πρέπει να εντοπίσουμε τον πρώτο χαρακτήρα της συμβολοσειράς που είναι το `π`. Πράγματι:

```
stixos[index]
```

'π'

Αν δεν εντοπιστεί η λέξη που αναζητούμε στην συμβολοσειρά η Python θα επιστρέψει: `-1`

```
stixos.find('πάντα')
```

-1

Η αναζήτηση είναι case sensitive δηλαδή γίνεται διάκριση ανάμεσα σε πεζά και κεφαλαία.

```
stixos.find('Ελλάδα') # επιστρέφει τον δείκτη 2 γιατί εντοπίστηκε η λέξη κλειδί
```

2

```
stixos.find('ΕΛΛΑΔΑ') # επιστρέφει -1 γιατί δεν εντοπίστηκε η λέξη κλειδί
```

-1

Μια άλλη σημαντική μέθοδος των συμβολοσειρών είναι η μέθοδος `replace` κατά την οποία μπορούμε να αντικαταστήσουμε τα περιεχόμενα μιας συμβολοσειράς. Στην πρώτη παράμετρο ορίζουμε την συμβολοσειρά που θέλουμε να αντικαταστήσουμε με την δεύτερη παράμετρο.

```
stixos.replace('ποτέ', 'πάντα')
```

3. Λίστες. Εκφράσεις, τελεστές

Στην τρέχουσα ενότητα γίνεται αναφορά σε μια από τις πιο δυνατές και χρήσιμες δομές της Python, τις λίστες. Οι λίστες μας επιτρέπουν να αποθηκεύομε σύνολα πληροφορίας σε ένα μέρος είτε πρόκειται για ένα είτε για εκατομμύρια στοιχεία. Οι λίστες αποτελούνται από μία σειρά από στοιχεία, καθένα από τα οποία μπορεί να ανήκει σε διαφορετικό τύπο δεδομένων.

Definition 4

Μία **λίστα** (list) είναι μια διατεταγμένη συλλογή τιμών, οι οποίες αντιστοιχίζονται σε δείκτες. Οι τιμές που είναι μέλη μιας λίστας ονομάζονται **στοιχεία** (elements). Τα στοιχεία μιας λίστας δεν χρειάζεται να είναι ίδιου τύπου και ένα στοιχείο σε μία λίστα μπορεί να υπάρχει περισσότερες από μία φορές. Μία λίστα μέσα σε μία άλλη λίστα ονομάζεται **εμφωλευμένη λίστα** (nested list). Επιπρόσθετα, τόσο οι λίστες όσο και οι συμβολοσειρές, που συμπεριφέρονται ως διατεταγμένες συλλογές τιμών, ονομάζονται **ακολουθίες** (sequences). Τα στοιχεία μιας λίστας διαχωρίζονται με κόμματα και περικλείονται σε τετράγωνες αγκύλες ([και]). Μία λίστα που δεν περιέχει στοιχεία ονομάζεται άδεια λίστα και συμβολίζεται με [] [Αγγελιδάκης, 2015].

Παρακάτω δίνονται μερικά παραδείγματα από λίστες

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
colors = ["red", "green", "black", "blue"]
scores = [10, 8, 9, -2, 9]
myList = ["one", 2, 3.0]
languages=[[['English'], ['Gujarati'], ['Hindi'], 'Romanian', 'Spanish']] # εμφωλευμένη λίστα (nested list)
list_A = [] # άδεια λίστα
```

Οι λίστες είναι ταξινομημένες συλλογές δεδομένων. Η πρόσβαση στα στοιχεία της λίστας γίνεται μέσω του δείκτη ή της θέσης του κάθε στοιχείου. Το σημαντικό που πρέπει να συγκρατηθεί είναι ότι η αριθμηση των στοιχείων σε μια λίστα ξεκινάει από το μηδεν. Το πρώτο στοιχείο έχει δείκτη 0, το δεύτερο 1 κ.ο.κ. Το τελευταίο στοιχείο στην λίστα έχει τον δείκτη -1, το δεύτερο στοιχείο από το τέλος των δείκτη -2 κ.ο.κ. Δείτε στο παρακάτω παράδειγμα τι εκτυπώνεται με βάση την θέση που δηλώνουμε στην λίστα.

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles[0])
print(bicycles[1])
print(bicycles[-1])
print(bicycles[1:3])
print(bicycles[1:3])
```

```
trek
cannondale
specialized
['cannondale', 'redline']
['cannondale', 'redline']
```

Παράδειγμα με εμφωλευμένη λίστα (nested list):

```
two_by_two = [[1, 2], [3, 4]]
print(two_by_two[0][1])
print(two_by_two[1][1])
```

```
2
4
```

Οι λίστες που περιέχουν συνεχόμενους ακέραιους αριθμούς μπορούν εύκολα να δημιουργηθούν ως εξής:

```
mylist = list(range(1,20))
print(mylist)

mylist1 = list(range(10))
print(mylist1)

mylist2 = list(range(1,20,4))
print(mylist2)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 5, 9, 13, 17]
```

Η μέθοδο index() μιας λίστας επιστρέφει το ευρετήριο (index) μιας τιμής:

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles.index('redline'))
```

```
2
```

Σε αντίθεση με τις συμβολοσειρές (strings) οι λίστες είναι μεταβαλλόμενες δομές δεδομένων. Αυτό σημαίνει ότι μπορούμε να τροποποιήσουμε τα στοιχεία της λίστα, να προσθέσουμε νέα ή να αφαιρέσουμε.

Για παράδειγμα μπορούμε να τροποποιήσουμε τα στοιχεία μιας λίστας ως εξής:

```
colors=['caramel','gold','silver','occur']
colors[3]='bronze'
print(colors)
```

```
['caramel', 'gold', 'silver', 'bronze']
```

```
colors=['caramel','gold','silver','occur']
colors[2:]=[ 'bronze','silver']
print(colors)
```

```
['caramel', 'gold', 'bronze', 'silver']
```

```
colors=['caramel','gold','silver','occur']
colors[2:3]=[ 'bronze','silver']
print(colors)
```

```
['caramel', 'gold', 'bronze', 'silver', 'occur']
```

Μπορούμε να προσθέσουμε **ένα** στοιχείο σε μια λίστα με την μέθοδο *append* π.χ.

```
gods = ['Δίας', 'Ερμής', 'Ποσειδώνας', 'Ηφαίστος']
print("{} θεοί".format(len(gods)))
print(gods)
```

```
gods.append("Αππόλωνας")
gods.append("Άρης")
print("{} θεοί".format(len(gods)))
print(gods)
```

```
4 θεοί
['Δίας', 'Ερμής', 'Ποσειδώνας', 'Ηφαίστος']
6 θεοί
['Δίας', 'Ερμής', 'Ποσειδώνας', 'Ηφαίστος', 'Αππόλωνας', 'Άρης']
```

Αν επιθυμούμε να προσθέσουμε πολλά στοιχεία τότε χρησιμοποιείται η μέθοδος *extend*

```
months=["Ιανουάριος", "Φεβρουάριος", "Μάρτιος"]
months.extend(["Απρίλιος", "Μάϊος"]) # προσοχή το όρισμα στην μέθοδο extend είναι λίστα (ή κάποιο iterable object)
```

Μπορούμε να αφαιρέσουμε στοιχεία από μία λίστα με διάφορους τρόπους.

Με την πρόταση *del*:

Όταν γνωρίζουμε την θέση του στοιχείου που θέλουμε να αφαιρεθεί από μια λίστα χρησιμοποιούμαι την πρόταση *del*. Για παράδειγμα

```

god = ['Δίας', 'Ερμής', 'Ποσειδώνας', 'Ηφαίστος']
del god[2]
print(god)

# Το μήκος άλλαξε και μερικές από τις αντιστοιχίες στην λίστα. Πλέον ο θεός στην θέση 2 είναι o:
print(god[2])

```

```
['Δίας', 'Ερμής', 'Ηφαίστος']
```

Με την μέθοδο `pop`:

Μία άλλη χρήσιμη μέθοδο για την αφαίρεση στοιχείων από μια λίστα είναι η μέθοδος `pop`. Μέσω της μεθόδου αυτής όχι απλά αφαιρείται το στοιχείο από την λίστα αλλά επιστρέφεται και ως τιμή διαθέσιμη να την εκμεταλλευτεί ο προγραμματιστής π.χ. σε μία νέα μεταβλητή. Δείτε το εξής παράδειγμα:

```

cars=["Alfa Romeo", "Renault", "BMW", "Renault", "Porsche"]
speed_car=cars.pop()
print(cars)
print(speed_car)

speed_car2=cars.pop(0)
print(speed_car2)

```

```
['Alfa Romeo', 'Renault', 'BMW', 'Renault']
Porsche
Alfa Romeo
```

Όπως φαίνεται από το παράδειγμα αν χρησιμοποιηθεί η μέθοδος `pop()` χωρίς δείκτη τότε αφαιρείται το τελευταίο στοιχείο της λίστας.

Με την μέθοδο `remove`:

Επιπλέον μπορούμε να αφαιρέσουμε στοιχεία από μία λίστα με την χρήση μιας τιμής και όχι με βάση τον δείκτη. Ωστόσο η παραπάνω τεχνική θα αφαιρέσει το πρώτο στοιχείο που θα εντοπιστεί με την τιμή αυτή.

```

god = ['Δίας', 'Ερμής', 'Ποσειδώνας', 'Ερμής', 'Ηφαίστος']

god.remove("Ερμής")
print(god)

```

```
['Δίας', 'Ποσειδώνας', 'Ερμής', 'Ηφαίστος']
```

Χρήσιμες μέθοδοι (methods) και συναρτήσεις (functions)

Μία από τις πλέον χρήσιμες μεθόδους της κλάσσης `list` είναι η ταξινόμηση (`sort`).

```

cars=[ "Porsche", "Alfa Romeo", "Renault", "BMW", "Audi" ]
cars.sort()
print(cars)

```

```
['Alfa Romeo', 'Audi', 'BMW', 'Porsche', 'Renault']
```

όπως φαίνεται η ταξινόμηση των στοιχείων της λίστας είναι μόνιμη. Ωστόσο αν θέλουμε προσωρινή ταξινόμηση τότε χρησιμοποιείται η function `sorted`.

```

cars=[ "Porsche", "Alfa Romeo", "Renault", "BMW", "Audi" ]
print(sorted(cars))
print(cars)

```

```
['Alfa Romeo', 'Audi', 'BMW', 'Porsche', 'Renault']
['Porsche', 'Alfa Romeo', 'Renault', 'BMW', 'Audi']
```

Όπως βλέπετε κατά την τελευταία εκτύπωση η λίστα διατηρεί την αρχική ταξινόμηση.

Επιπλέον με την μέθοδο `reverse` μπορούμε να αντιστρέψουμε την διάταξη των στοιχείων της λίστας. Και εδώ το αποτέλεσμα είναι μόνιμο.

```
cars=[ "Porsche", "Alfa Romeo", "Renault", "BMW", "Audi" ]  
cars.reverse()  
print(cars)
```

```
['Audi', 'BMW', 'Renault', 'Alfa Romeo', 'Porsche']
```

Ακόμα μέσω της συνάρτησης `len` επιστρέφεται το πλήθος των στοιχείων της λίστας.

```
languages=['English', 'Gujarati', 'Hindi','Romanian','Spanish']  
print(len(languages))
```

```
5
```

Εδώ πρέπει να δοθεί προσοχή. Γιατί ενώ η αρίθμηση των δεικτών ξεκινά από το 0 το μήκος της λίστας ξεκινά από το 1 για λίστα με ένα στοιχείο. Οπότε στο παρακάτω παράδειγμα θα λάβουμε σφάλμα εκτέλεσης αν πάμε να πάρουμε το 50 και τελευταίο στοιχείο της λίστας. Γιατί αυτό ορίζεται με τον δείκτη 4 και όχι 5.

```
cars=[ "Porsche", "Alfa Romeo", "Renault", "BMW", "Audi" ]  
print("Το μήκος της λίστας (πλήθος στοιχείων) είναι: ", len(cars))  
print(cars[5])
```

```
Το μήκος της λίστας (πλήθος στοιχείων) είναι: 5
```

```
-----  
IndexError                                                 Traceback (most recent call last)  
Input In [18], in <cell line: 3>()  
    1 cars=[ "Porsche", "Alfa Romeo", "Renault", "BMW", "Audi" ]  
    2 print("Το μήκος της λίστας (πλήθος στοιχείων) είναι: ", len(cars))  
----> 3 print(cars[5])  
  
IndexError: list index out of range
```

Σημείωση

Συχνά χρησιμοποιούμε έναν βρόγχο (loop) όπως το `for` για να προσπελάσουμε ένα προς ένα τα στοιχεία μιας λίστας. Δείτε το επόμενο παράδειγμα.

```
list = ['physics', 'chemistry', 1997, 2000]  
for item in list:  
    print(item)
```

```
physics  
chemistry  
1997  
2000
```

Σημαντικό

Ιδιαίτερη προσοχή πρέπει να δίνεται στον τρόπο που αντιγράφουμε λίστες. Δείτε γιατί στο επόμενο παράδειγμα.

```
nisia = ["Μήλος", "Κρήτη", "Λέσβος"]  
greek_islands = nisia  
  
greek_islands.append("Κέρκυρα")  
  
print(nisia)
```

```
['Μήλος', 'Κρήτη', 'Λέσβος', 'Κέρκυρα']
```

Οπως φαίνεται οι μεταβλητές `nisia` και `greek_islands` αντιστοιχούν στο ίδιο υποκείμενο object και αν μεταβάλλοντας τα στοιχεία της μίας μεταβλητής η αλλαγή αντικατοπτρίζεται και στα στοιχεία της δεύτερης. Ο ενδεδειγμένος τρόπος για να αντιγράψουμε μια λίστα είναι να αντιγράψουμε όλα τα στοιχεία της ώστε να έχουμε δύο ανεξάρτητα objects (κλωνοποίηση).

```

nisia = ["Μήλος", "Κρήτη", "Λέσβος"]
greek_islands = nisia[:]

greek_islands.append("Κέρκυρα")

print(nisia)
print(greek_islands)

```

```

['Μήλος', 'Κρήτη', 'Λέσβος']
['Μήλος', 'Κρήτη', 'Λέσβος', 'Κέρκυρα']

```

Εκφράσεις και τελεστές

Μία **έκφραση (expression)** είναι ένας συνδυασμός από **τιμές**, **μεταβλητές**, **τελεστές** και **κλήσεις σε συναρτήσεις**. Οι **τελεστές (operators)** είναι λειτουργίες που κάνουν κάτι και μπορούν να αναπαρασταθούν με σύμβολα όπως το + ή με λέξεις κλειδιά όπως το and. Η αποτίμηση μιας έκφρασης παράγει μία τιμή και αυτός είναι και ο λόγος που μία έκφραση μπορεί να βρίσκεται στο δεξί μέρος μια εντολής εκχώρησης. Όταν μία μεταβλητή εμφανίζεται σε έκφραση, αντικαθίσταται από την τιμή της, πριν απότιμηθεί η έκφραση [[Αγγελιδάκης, 2015](#)]. Δεν απαιτείται μία έκφραση να περιέχει ταυτόχρονα και τιμές και μεταβλητές και τελεστές. Μία τιμή, όπως και μία μεταβλητή, από μόνες τους είναι επίσης εκφράσεις.

Οι τελεστές χρησιμοποιούνται με αριθμητικές τιμές για την εκτέλεση μαθηματικών πράξεων. Ορισμένοι τελεστές έχουν εφαρμογή και σε συμβολοσειρές. Πιο συγκεκρικά διατίθενται οι παρακάτω τελεστές:

Αριθμητικοί τελεστές

Τελεστής	Όνομα	Παράδειγμα
+	Πρόσθεση αριθμών ή συνένωση συμβολοσειρών	x + y
-	Αφαίρεση	x - y
*	Πολλαπλασίασμός ή επανάληψη συμβολοσειράς	x * y
/	Διαίρεση	x / y
%	Υπόλοιπο διαίρεσης δύο αριθμών	x % y
**	Έψωση αριθμού σε δύναμη	x ** y
//	Διαίρεση δύο αριθμών στρογγυλοποιημένη προς τα κάτω	x // y

Τελεστές εκχώρησης

Χρησιμοποιούνται για να αποδώσουν τιμές σε μεταβλητές.

Τελεστής Παράδειγμα Αντίστοιχο με

=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
=	x = 3	x = x 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Τελεστές σύγκρισης

Χρησιμοποιούνται για την σύγκριση 2 τιμών

Τελεστής	Σύγκριση	Παράδειγμα
<code>==</code>	Ίσον	<code>x == y</code>
<code>!=</code>	Διαφορετικό	<code>x != y</code>
<code>></code>	Μεγαλύτερο από	<code>x > y</code>
<code><</code>	Μικρότερο από	<code>x < y</code>
<code>>=</code>	Μεγαλύτερο ή ίσον από	<code>x >= y</code>
<code><=</code>	Μικρότερο ή ίσον από	<code>x <= y</code>

Λογικοί τελεστές

Τελεστής	Περιγραφή	Παράδειγμα
<code>and</code>	Επιστρέφει Αληθές (True) αν και οι δύο προτάσεις είναι αληθείς	<code>x < 5 and x < 10</code>
<code>or</code>	Επιστρέφει Αληθές (True) αν έστω μία από τις προτάσεις είναι αληθή	<code>x < 5 or x < 4</code>
<code>not</code>	Αντιστροφή απότελέσματος, επιστρέφει Μη Αληθές όταν το αποτέλεσμα είναι αληθές	<code>not(x < 5 and x < 10)</code>

Εκτός από τους παραπάνω τελεστές υπάρχουν πιο εξειδικεύμενοι τελεστές που δεν θα αναφερθούμε (Bitwise, Membership, Identity operators).

Οι τελεστές στη Python τηρούν την αντίστοιχη προτεραιότητα που χρησιμοποιείται και στα μαθηματικά.

Οι παρενθέσεις έχουν τη μεγαλύτερη προτεραιότητα. Πολλαπλασιασμός και διαίρεση έχουν υψηλότερα προτεταιότητα από την πρόσθεση και αφαίρεση.

Τελεστές με την ίδια προτεραιότητα αποτιμώνται από τα αριστερά προς τα δεξιά.

4. Πλειάδες και λεξικά

Στην συνέχεια παρουσιάζονται δύο άλλες καθιερωμένες δομές στην Python, οι πλειάδες (tuples) και τα λεξικά (dictionaries).

Πλειάδες (tuples)

Οι πλειάδες είναι μια δομή δεδομένων παρόμοια με τις λίστες. Όπως και οι λίστες αποτελούν μια συλλογή αντικειμένων (objects). Όμως διαφέρουν σε ένα σημαντικό χαρακτηριστικό από τις λίστες, είναι αμετάβλητες (immutable), δηλαδή δεν μπορούμε να μεταβάλλουμε το περιεχόμενό τους εφόσον τις δημιουργήσουμε. Κατά συνέπεια ενώ για την προσπέλαση των στοιχείων της χρησιμοποιούνται ίδιες τεχνικές και μέθοδοι (indexing, slicing κτλ) δεν υποστηρίζουν ωστόσο τις αντίστοιχες μεθόδους αφαίρεσης, τροποποίησης και προσθήκης στοιχείων. Υπό την έννοια αυτή οι πλειάδες είναι μια ασφαλή δομή δεδομένων μέσω της οποίας εξασφαλίζεται ότι τα δεδομένα δεν θα τροποποιηθούν από λάθος ή ακόμα και από επιλογή. Συνήθως χρησιμοποιούνται κατά την επιστροφή πολλαπλών τιμών από μια συνάρτηση αλλά και όταν θέλουμε να «πακετάρουμε» δεδομένα (tuple packing). Οι πλειάδες ορίζονται με παρόμοιο τρόπο σαν τις λίστες αλλά αντί για αγκύλες χρησιμοποιούνται παρανθέσεις (αν και δεν είναι απαραίτητες). Για παράδειγμα:

```
information = ('UTH', 'https://www.uth.gr/')
```

Το ίδιο μπορεί να γραφτεί και χωρίς παρενθέσεις

```
information = 'UTH', 'https://www.uth.gr/'
```

Όμως πρέπει να λαμβάνουμε υπόψιν ότι ο ορισμός μιας πλειάδες με ένα μόνο στοιχείο πρέπει να ορίζεται με το στοιχείο και να συνοδεύεται από ένα κόμμα. Σε διαφορετική περίπτωση θα επιστρέψει μια μεταβλητή με τύπο αυτόν που περάσαμε στο υποτιθέμενο πρώτο στοιχείο. Δηλαδή το παρακάτω θα δημιουργήσει μια μεταβλητή ακέραιου τύπου και όχι μια πλειάδα με ένα στοιχείο

```
year = (2022)
print(type(year))
```

```
<class 'int'>
```

Ο σωστός τρόπος συγγραφής αν θέλουμε να πάρουμε μια πλειάδα με ένα μόνο στοιχείο είναι:

```
year = (2022,)
print(type(year))
```

```
<class 'tuple'>
```

Ενώ αν θέλουμε να δημιουργήσουμε μία άδεια πλειάδα τότε χρησιμοποιούμε απλώς δύο παρενθέσεις:

```
empty_tuple = ()
```

Επιπλέον μέσω της συνάρτησης `tuple` η οποία δέχεται ως όρισμα μία συμβολοσειρά ή λίστα μπορούμε να δημιουργήσουμε νέες πλειάδες. Προσέξτε το αποτέλεσμα όταν ορίζουμε σαν παράμετρο συμβολοσειρά στην συνάρτηση `tuple`.

```
regions= tuple("Κρήτη")
print(regions)
```

```
('Κ', 'ρ', 'ή', 'τ', 'η')
```

```
regions= tuple(["Κρήτη", "Ηπειρος", "Θράκη"])
print(regions)
```

```
('Κρήτη', 'Ηπειρος', 'Θράκη')
```

Με κενό όρισμα επιστρέφεται μία άδεια λίστα:

```
regions= tuple()
```

Όπως προαναφέρθηκε οι πλειάδες είναι αμετάβλητες δομές δεδομένων και έτσι δεν μπορούμε να χρησιμοποιήσουμε μεθόδους όπως `append`, `del`, `sort` κ.α. Μπορούμε, όπως και στις λίστες, να αναφερθούμε στα στοιχεία της πλειάδας με τους τελεστές `[]` και με το ευρετήριο (`index`) ή να εξάγουμε τμήμα από τα στοιχεία (`slice`) με τον τελεστή `[:]`. Λαμβάνουμε το πλήθος των στοιχείων της πλειάδας (`length`) με την συνάρτηση `len`.

Οι πλειάδες χρησιμοποιούνται για να πακετάρουμε μια συλλογή δεδομένων σε ένα αντικείμενο (tuple packing) π.χ.

```
fruit1 = "Μήλος"
fruit2 = "Πορτοκάλι"
fruit1 = "Μανταρίνι"
fruits = fruit1, fruit1, fruit2 # tuple packing
print(fruits)
```

```
('Μανταρίνι', 'Μανταρίνι', 'Πορτοκάλι')
```

Βέβαια υπάρχει και η αντίστροφη διαδικασία, το «ξε-πακετάρισμα» δεδομένων (tuple unpacking). Όταν δηλαδή αποδίδουμε τα στοιχεία της πλειάδας σε ξεχωριστές μεταβλητές.

```
cities = "Αθήνα", "Βόλος", "Πάτρα"
capital, city1, city2 = cities # tuple packing
print(capital, city1, city2)
```

```
Αθήνα Βόλος Πάτρα
```

Ο αμετάβλητος χαρακτήρας των πλειάδων φαίνεται στα παρακάτω παραδείγμα στα οποία επιχειρείται η τροποποίηση των δεδομένων της.

```
cities = ("Αθήνα", "Βόλος", "Πάτρα")
cities.pop(1)
```

```
-----
AttributeError                                     Traceback (most recent call last)
Input In [11], in <cell line: 2>()
      1 cities = ("Αθήνα", "Βόλος", "Πάτρα")
----> 2 cities.pop(1)

AttributeError: 'tuple' object has no attribute 'pop'
```

```
cities.append("Ρόδος")
```

```
-----  
AttributeError                                     Traceback (most recent call last)  
Input In [12], in <cell line: 1>()  
----> 1 cities.append("Ρόδος")  
  
AttributeError: 'tuple' object has no attribute 'append'
```

Όπως μας ενημερώνει και το σχετικό σφάλμα εκτέλεσης δεν υπάρχουν οι σχετικές μέθοδοι για αντικείμενα της κλάσης *tuple*.

Αντίστοιχο σφάλμα λαμβάνουμε και με τον παρακάνω κώδικα όταν πάμε να τροποποιήσουμε ένα στοιχείο της πλειάδας:

```
mytuple = (1, 2, 3)  
mytuple[0] = 999
```

```
-----  
TypeError                                      Traceback (most recent call last)  
Input In [13], in <cell line: 2>()  
      1 mytuple = (1, 2, 3)  
----> 2 mytuple[0] = 999  
  
TypeError: 'tuple' object does not support item assignment
```

💡 Tip

Μια χρήσιμη λειτουργία των πλειάδων είναι η χρήση τους κατά την αντιμετάθεση δύο μεταβλητών. Δείτε το παρακάτω παράδειγμα.

```
island='Λέσβος'  
island2='Χίος'  
island, island2 = island2, island  
  
print(island)  
print(island2)
```

```
Χίος  
Λέσβος
```

⚠️ Προειδοποίηση

Προσοχή! Αν και οι πλειάδες είναι αμετάβλητος τύπος δεδομένων ωστόσο στα στοιχεία τους μπορούν να περιλαμβάνουν μεταβλητούς τύπους δεδομένων. Αυτό σημαίνει ότι μπορούμε να τροποποιήσουμε τα στοιχεία αυτά. Παρακάτω δίνεται ένα παράδειγμα.

```
cities = ("Αθήνα", ["Βόλος", "Πάτρα"])  
cities[1][0] = "Καβάλα"  
print(cities)
```

```
('Αθήνα', ['Καβάλα', 'Πάτρα'])
```

Μια εξήγηση σε αυτήν την αντιφατική συμπεριφορά δίνεται στο παρακάτω νήμα: <https://stackoverflow.com/questions/9755990/why-can-tuples-contain-mutable-items>

Λεξικά (Dictionaries)

Τα λεξικά στην Python αποτελούν συλλογές αντικειμένων, όπως οι λίστες και οι πλειάδες. Ένα σημαντικό χαρακτηριστικό των λεξικών είναι ότι αυτά αποθηκεύουν δεδομένα κατά ζεύγη, με την μορφή κλειδί-τιμή (key-value pairs). Κάθε κλειδί σε ένα λεξικό συνοδεύεται από μία τιμή. Κάθε κλειδί αποτελεί ουσιαστικά ένα μοναδικό αναγνωριστικό για την συνοδευτική τιμή και γι'αυτό τον λόγο δεν μπορεί να υπάρχει δεύτερο ίδιο κλειδί. Ακόμα, τα κλειδιά πρέπει να ορίζονται από αμετάβλητους τύπους δεδομένων δηλαδή είτε από μία συμβολοσειρά είτε από έναν ακέραιο ή δεκαδικό. Δεν μπορεί όμως μια λίστα να είναι κλειδί. Μια πλειάδα μπορεί να είναι κλειδί αλλά με την προϋπόθεση ότι και αυτή δεν θα αποτελείται από μεταβλητούς τύπους δεδομένων. Τα λεξικά περικλείονται σε {}, τα ζεύγη ορίζονται υπό την μορφή κλειδί:τιμή και χωρίζονται μεταξύ τους με κόμμα (,) δηλαδή:

```
d = {key1 : value1, key2 : value2 }
```

Για παράδειγμα:

```
phones ={"Χρήστος": "69936565", "Κώστας": "246541353", "Βαγγέλης": "546546536"}  
print(phones)
```

```
{'Χρήστος': '69936565', 'Κώστας': '246541353', 'Βαγγέλης': '546546536'}
```

Τα κλειδιά (keys) σε αυτήν την περίπτωση είναι τα ονόματα και οι αριθμοί τηλεφώνων (ως συμβολοσειρές ορισμένες) οι τιμές (values).

Εναλλακτικά μπορούμε να δημιουργήσουμε ένα άδειο λεξικό και να προσθέσουμε στην συνέχεια ζεύγη.

```
phones ={}  
phones["Χρήστος"] ="69936565"  
phones["Κώστας"] ="246541353"  
phones["Βαγγέλης"] ="546546536"
```

Ένας άλλος τρόπος δημιουργίας λεξικών είναι με την συνάρτηση *dict*.

```
phones =dict(Χρήστος="69936565", Κώστας="246541353", Βαγγέλης="546546536")
```

Με αυτόν τον τρόπο ορίζουμε τα κλειδιά μέσω μεταβλητών και γι" αυτό τον λόγο εφαρμόζονται οι περιορισμοί που αφορούν την ονοματολογία των μεταβλητών.

Στα λεξικά τα ζεύγη αυτά δεν ταξινομούνται με κάποια συγκεκριμένη σειρά αλλά με έναν μηχανισμό της Python που λέγεται *hashing* και αποσκοπεί στην γρήγορη ανάκτηση τους. Η ταξινόμηση αυτή αλλάζει κάθε φορά κατά τυχαίο τρόπο όταν τροποποιούμε ένα λεξικό. Για τον λόγο αυτό δεν υπάρχει η έννοια της θέσης ή του δείκτη (index) όπως στις λίστες και στις πλειάδες. Η ανάκτηση μιας τιμής από ένα ζεύγος γίνεται με βάση το κλειδί π.χ.* *d["a_key"]**. Γίνεται δηλαδή με παρόμοιο τρόπο όπως στις λίστες και τις πλειάδες αλλά αντί για το ευρετήριο θέσης ορίζουμε το κλειδί.

```
print(phones["Χρήστος"])  
print(phones["Βαγγέλης"])
```

```
69936565  
546546536
```

Όπως προαναφέρθηκε δεν μπορούμε να έχουμε διπλό κλειδί σε ένα λεξικό. Έτσι αν επιχειρήσουμε να προσθέσουμε ένα ζεύγος με υφιστάμενο κλειδί στην πράξη αυτό που θα γίνει είναι να αντικαταστήσουμε την παλιά τιμή με μια νέα:

```
print(phones["Βαγγέλης"])  
phones["Βαγγέλης"]="66666666"  
print(phones["Βαγγέλης"])
```

```
546546536  
66666666
```

Για την διαγραφή ενός ζεύγους από λεξικό χρησιμοποιείται η συνάρτηση *del* ή δηλ. *del(d["key"])*. Δείτε το επόμενο παράδειγμα:

```
del(phones["Βαγγέλης"])  
print(phones)
```

```
{'Χρήστος': '69936565', 'Κώστας': '246541353'}
```

Με την συνάρτηση *len* επιστρέφεται το μέγεθος του λεξικού δηλαδή το πλήθος των ζευγαριών κλειδιών/τιμών που περιέχει:

```
print(len(phones))
```

```
2
```

Μέσω του τελεστή *in* διαπιστώνεται αν υπάρχει ένα κλειδί σε ένα λεξικό:

```
print("Χρήστος" in phones)
```

```
True
```

Με την μέθοδο `keys` ενός λεξικού επιστρέφονται τα κλειδιά του. Αντίστοιχα με την μέθοδο `values` επιστρέφονται οι τιμές του ενώ με την μέθοδο `items` επιστρέφονται τα ζεύγη κλειδιών/τιμών. Τα επιστρεφόμενα objects είναι αντίστοιχα `dict_keys`, `dict_values`, `dict_items`. Ο παρακάτω κύρικας περιγράφει τις παραπάνω λειτουργίες:

```
d = {'a': 10, 'b': 20, 'c': 30}
print(d.keys())
print(d.values())
print(d.items())
```

```
dict_keys(['a', 'b', 'c'])
dict_values([10, 20, 30])
dict_items([('a', 10), ('b', 20), ('c', 30)])
```

Αν θέλουμε να ταξινομήσουμε το λεξικό με βάση τα κλειδιά τότε το κάνουμε με την συνάρτηση `sorted`:

```
thisdict = {
    "year": 1964,
    "brand": "Ford",
    "model": "Mustang"
}

print(sorted(thisdict))
```

```
['brand', 'model', 'year']
```

Πολύ συνηθισμένη περίπτωση είναι τα λεξικά να περιλαμβάνουν σαν τιμές άλλα λεξικά.

```
contacts = {"Χρήστος": {"Σπίτι": "457456456", "Εργασία": "48856"},
            "Γιάννης": {"Σπίτι": "8753778", "Εργασία": "45654656"},
            "Κώστας": {"Κινητό": "45475354"}}
```

Σε αυτή την περίπτωση ορίζοντας διαδοχικά τα κλειδιά παίρνουμε τις τιμές που επιθυμούμε:

```
print(contacts["Χρήστος"])
print(contacts["Χρήστος"]["Εργασία"])
```

```
{'Σπίτι': '457456456', 'Εργασία': '48856'}
48856
```

Μπορούμε να διατρέξουμε τα ζεύγη ενός λεξικού αλλά η σειρά που θα γίνεται η ανάκτηση μπορεί να μην είναι με την σειρά που ορίζονται και όχι πάντα η ίδια.

```
for phone in phones:
    print(phones[phone])
```

```
69936565
246541353
```

Όπως και με τις λίστες έτσι και με τα λεξικά πρέπει να είμαστε προσεκτικοί όταν αντιγράφουμε ένα λεξικό. Η αντιγραφή μπορεί να αναφέρεται στο ίδιο αντικείμενο και τροποποίηση των δεδομένων ενός λεξικού θα επιφέρει και την αντίστοιχη τροποποίηση στο άλλο.

```
names1 = {'name': "Κώστας"}
names2=names1
names2['name']="Γιάννης"
print(names1)
```

```
{'name': 'Γιάννης'}
```

Σε αυτή την περίπτωση χρησιμοποιείται η μέθοδος `copy` για την αντιγραφή των δεδομένων ενός λεξικού σε ένα άλλο.

```
names1 = {'name': "Κώστας"}
names2=names1.copy()
names2['name']="Γιάννης"
print(names1)
```

```
{'name': 'Κώστας'}
```

5. Έλεγχος ροής εκτέλεσης

Όλες οι γλώσσες προγραμματισμού απαρτίζονται από μια σειρά από προγραμματιστικές δομές. Οι τρεις βασικές δομές ελέγχου ροής προγράμματος είναι η δομή της ακολουθίας εντολών, η δομή της απόφασης και η δομή της επανάληψης. Κάποιες από αυτές αντιστοιχούν σε περισσότερες από μία εντολές [Manis, 2015].

Η λογική boolean

Για την υλοποίηση των δομών απόφασης χρησιμοποιείται η λογική **boolean** κατά την οποία ελέγχονται μία ή περισσότερες συνθήκες και ανάλογα το αποτέλεσμα επιλέγεται ποια ακολουθία εντολών θα εκτελεστεί. Το όνομα **boolean** προέρχεται από τον Βρετανό Μαθηματικό George Boole, ο οποίος εισήγαγε την ομώνυμη άλγεβρα που αφορά σε λογικούς κανόνες συνδυασμού των δύο τιμών True (Αληθής) και False (Ψευδής). Οι τιμές αυτές ονομάζονται λογικές ή Boolean και ο τύπος τους στην Python είναι ο `bool` [Αγγελιδάκης, 2015].

```
type(False)
```

```
bool
```

Μία μεταβλητή μπορεί να δείχνει και σε μία τιμή τύπου `bool`. Για παράδειγμα:

```
raining=True  
type(raining)
```

```
bool
```

Οι λογικές μεταβλητές είναι μεταβλητές οι οποίες διέπονται από την δυαδική λογική στην επιστήμη της πληροφορικής (1 και 0) και παίρνουν δύο τιμές `True` ή `False`. Ουσιαστικά πρόκεται για συνώνυμα του 1 (True) και 0 (False).

```
True+True  
42 * True + False
```

```
42
```

Ο συνδυασμός των Boolean τιμών γίνεται με τη χρήση τριών βασικών λογικών τελεστών: **not**, **and** και **or**. Οι τελεστές αυτοί συμβολίζουν λογικές πράξεις (όχι, και, ή) και βοηθούν στη λήψη αποφάσεων σε ένα πρόγραμμα. Η σημαντική τους είναι πολύ παρόμοια με την καθημερινή τους σημασία [Αγγελιδάκης, 2015]. Ας υποθέσουμε ότι A και B είναι δύο μεταβλητές που δείχνουν σε Boolean τιμές ή δύο εκφράσεις (ονομάζονται λογικές ή Boolean) που αποτιμώνται σε Boolean τιμές. Ο επόμενος πίνακας, ο οποίος ονομάζεται πίνακας αληθείας, περιέχει τις τιμές που επιστρέφουν οι τρεις λογικές πράξεις για όλους τους συνδυασμούς τιμών των A και B .

A	B	A AND B	A OR B	NOT A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Εικ. 4 Πίνακας αληθείας για τους βασικούς λογικούς τελεστές.

Η λογική έκφραση `not A` είναι αληθής, όταν η A ψευδής, και ψευδής, όταν η A είναι αληθής. Η λογική έκφραση A and B είναι αληθής μόνο όταν και η A και η B είναι αληθείς, σε κάθε άλλη περίπτωση είναι ψευδής. Η λογική έκφραση A or B είναι αληθής όταν η A είναι αληθής ή η B είναι αληθής, ή όταν και οι δύο είναι αληθείς.

Όταν χρησιμοποιούμε τον όρο **and** σημαίνει ότι και οι δύο προτάσεις πρέπει να είναι αληθείς για να αποδώσουν αληθές αποτέλεσμα. true.

Γενικά μπορούμε να συνοψίσουμε την χρήση του τελεστή **and** και την σύγκριση ανάμεσα σε τιμές boolean με βάση των παρακάτω πίνακα [[Heisler, 2021](#)]:

Συνδυασμός με τον τελεστή **and** Αποτέλεσμα

True and True	True
True and False	False
False and True	False
False and False	False

Με την χρήση του όρου **or** αρκεί μία από τις δύο προτάσεις να είναι True για να αποδώσει αποτέλεσμα αληθές. Αυτό φαίνεται πιο καλά στο παραπάνω πίνακα:

Συνδυασμός με τον τελεστή **or** Αποτέλεσμα

True or True	True
True or False	True
False or True	True
False or False	False

Τέλος με την χρήση του όρου **not** αντιστρέφεται η τιμή True ή False μιας πρότασης.

Επίδραση του τελεστή **not** Αποτέλεσμα

not True	False
not False	True

Φυσικά μπορούμε να συνδυάσουμε να συνδυάσουμε τους όρους **and** **or** και **not** και να διαχωρίσουμε τις συγκρίσεις με παρανθέσεις ώστε να δημιουργήσουμε πιο σύνθετες εκφράσεις σύγκρισης:

False or (False and True)
False
True and (False or True)
True
(not False) or True
True
False or (not False)
True

Σε μία λογική έκφραση μπορούμε να έχουμε και τελεστές σύγκρισης. Η Python έχει 6 τελεστές σύγκρισης:

Μικρότερο από (`<`) Μικρότερο/ίσο από (`<=`) Μεγαλύτερο από (`>`) Μεγαλύτερο/ίσο από (`>=`) Ίσο με (`==`) Διαφορετικό από (`!=`) Αυτοί οι τελεστές συγκρίνουν δύο τιμές και επιστρέφουν μια τιμή τύπου boolean δηλαδή είτε `True` είτε `False`.

Οι λογικές εκφράσεις χρησιμοποιούν παρενθέσεις και κανόνες προτεραιότητας, για να καθορίσουν τη σειρά αποτίμησης των τμημάτων από τα οποία αποτελούνται. Οι εκφράσεις μέσα σε παρενθέσεις αποτιμώνται πρώτες. Η προτεραιότητα των τελεστών, από τη μεγαλύτερη στη μικρότερη, είναι οι εξής:

`<, >, <=, >=, !=, ==, not, and, or.`

⚠ Προειδοποίηση

Προσοχή! Δεν πρέπει να συγχέεται ο τελεστής εκχώρησης τιμών σε μεταβλητή `=` με τον τελεστή ισότητας `==`.

Μερικά παραδείγματα τελεστών σύγκρισης

```
print(1 != 2)
print(1 != 1)
```

True
False

```
a = 1
b = 2
print(a == b)
a = b
a==b
print(a!=b)
```

False
False

```
1< 2 and 3 < 4
```

True

```
2< 1 and 4 < 3
```

False

```
1< 2 and 4 < 3
```

False

```
2< 1 and 3 < 4
```

False

Τους τελεστές σύγκρισης μπορούμε να τους χρησιμοποιήσουμε και με συμβολοσειρές:

```
"dog" == "cat"
```

False

```
"dog" == "dog"
```

True

```
"dog" != "cat"
```

True

Για να είναι δύο συμβολοσειρές ίσες πρέπει να έχουν ακριβώς την ίδια τιμή. Διαφοροποιήσεις ανάμεσα σε κεφαλαία ή μικρά ή η ύπαρξη κενών ανάμεσα σε δύο μεταβλητές συμβολοσειρών θα αποδόσει False σε μια έκφραση σύγκρισης ισότητας δηλαδή ότι οι δύο συμβολοσειρές δεν είναι ίσες.

Μπορούμε να συνδυάσουμε τους τελεστές σύγκρισης με λογικές τιμές boolean και τους λογικούς τελεστές or, and και not.

Για παράδειγμα:

```
True and not (1 != 1)
```

True

```
("A" != "A") or not (2 >= 3)
```

True

α. Η δομή της ακολουθίας εντολών

Οι εντολές σε ένα πρόγραμμα εκτελούνται σειριακά η μία μετά την άλλη ξεκινώντας από την πρώτη. Τελειώνει η εκτέλεση μίας εντολής και ακολου- θεί η εκτέλεση της επόμενης. Κάθε εντολή θα εκτελεστεί ακριβώς μία φορά, χωρίς δηλαδή να παραλειφθεί καμία από αυτές ή να επιστρέψουμε για να ξαναεκτελέσουμε κάποια. Παρακάτω δίνεται ένα χαρακτηριστικό παράδειγμα δομής ακολουθίας εντολών.

```
A=1  
B=5  
C=A+B  
print(C)
```

6

Σε αυτό το παράδειγμα οι εντολές εκτελούνται στο σύνολό τους, διαδοχικά η μία μετά την άλλη.

β. Η δομή της απόφασης

Στη δομή της απόφασης έχουμε μία περισσότερο πολύπλοκη δομή, στην οποία ο έλεγχος του προγράμματος καλείται να επιλέξει ανάμεσα σε δύο ή και περισσότερες διαφορετικές διαδρομές ανάλογα με το αν ισχύει ή όχι κάποια ή κάποιες συνθήκες. Για την υλοποίηση της απόφασης, κάθε γλώσσα μπορεί να έχει μία ή περισσότερες δομές. Η πιο συνηθισμένη είναι η δομή **if**, ενώ υπάρχει συνήθως και μία δομή για πολλαπλή απόφαση. Η Python έχει την **if-elif-else** για να υλοποιήσει την απόφαση, αλλά δεν έχει δομή για την πολλαπλή απόφαση και χρησιμοποιεί την **if-elif-else** για τον σκοπό αυτόν [[Manis, 2015](#)].

Η εντολή **if** χρησιμοποείται για έλεγχο της ροής εκτέλεσης ενός προγράμματος. Ελέγχεται μία συνθήκη και ανάλογα με το αποτέλεσμα (Αληθής ή Ψευδής) εκτελείται ή δεν εκτελείται μία ή κάποια άλλη ομάδα (μπλοκ) εντολών [[Αγγελιδάκης, 2015](#)]. Η εντολή if συντάσσεται ως εξής:

```
if συνθήκη:  
    μπλοκ εντολών 1 (true_block)  
  
else:  
    μπλοκ εντολών (false_block)
```

Κατά την εκτέλεση του παρακάτω κώδικα η Python θα δοκιμάσει την έκφραση (expression). Αν επιστρέψει True τότε θα εκτελέσει τις προτάσεις κώδικα που περιλαμβάνει η ενότητα true_block. Αν επιστρέψει όμως False θα εκτελέσει τις εντολές στην ενότητα false_block. Στην πιο απλή της μορφή η δομή της απόφασης μπορεί να παραλείπει το else τμήμα και το αντίστοιχο μπλοκ εντολών δηλ:

```
if συνθήκη:  
    μπλοκ εντολών (true_block)
```

Παρατηρήστε ότι η εντολή if μετά τον έλεγχο της συνθήκης κλείνει με : και στην συνέχεια ακολουθούν με εσοχή (indentation) οι εντολές/προτάσεις που θα εκτελεστούν εφόσον αληθεύει το αποτέλεσμα του ελέγχου. Από : ακολουθείται και η εντολή else. Το μπλοκ της εντολής ονομάζεται σώμα (body). Πρέπει να περιέχει υποχρεωτικά μια εντολή. Αν προσωρινά δεν θέλουμε να περιέχει κάποια εντολή μπορούμε να εισάγουμε την εντολή **pass**. Ιδιαίτερη σημασία πρέπει να δίνεται κατά την σύνταξη κατά την χρήση δηλαδή του σημείου : και την εσοχή των εντολών. Παρακάτω δίνεται ένα απλό παράδειγμα δομής απόφασης:

```
weight = 100  
if weight > 90:  
    print("Είστε υπέρβαρος. Παρακαλώ αποφύγετε τον ανελκυστήρα.")  
print("Ευχαριστούμε για την συνεργασία.")
```

Είστε υπέρβαρος. Παρακαλώ αποφύγετε τον ανελκυστήρα.
Ευχαριστούμε για την συνεργασία.

Παράδειγμα κώδικα if-else:

```

temperature = 20
if temperature > 30:
    print('Φορέστε κοντομάνικα.')
else:
    print('Φορέστε μακρυμάνικα.')
print('Ευχαριστώ.')

```

Φορέστε μακρυμάνικα.
Ευχαριστώ.

Για να προσθέσουμε περισσότερες επιλογές κατά τον έλεγχο συνθηκών χρησιμοποιούμε την σύνταξη if-elif-else:

```

score=90
if score >= 90:
    letter = 'A'
elif score >= 80:
    letter = 'B'
elif score >= 70:
    letter = 'C'
elif score >= 60:
    letter = 'D'
else:
    letter = 'F'

```

Οι συνθήκες if είναι δυνατόν να είναι αλυσιδωτές (εμφωλευμένες) δηλαδή if πρόταση μέσα σε άλλη if πρόταση:

```

num = 15
if num >= 0:
    if num == 0:
        print("Ο αριθμός δεν είναι ούτε αρνητικός ούτε θετικός")
    else:
        print("Ο αριθμός είναι θετικός")
else:
    print("Ο αριθμός είναι θετικός")

```

Ο αριθμός είναι θετικός

γ. η δομή της επανάληψης

Η δομή της επανάληψης είναι μια μορφή κώδικα κατά την οποία περιλαμβάνει εντολές οι οποίες εκτελούνται επαναληπτικά για όσο τηρείται μια συνθήκη. Με αυτόν τον τρόπο αποφεύγεται η επανάληψη της συγγραφής του ίδιου κώδικα πολλές φορές. Στο παρακάτω τμήμα κώδικα είναι χαρακτηριστικό πως επαναλαμβάνεται ο ίδιος κώδικα πολλές φορές.

```

print("Hello World")
print("Hello World")
print("Hello World")
print("Hello World")
print("Hello World")

```

Hello World
Hello World
Hello World
Hello World
Hello World

Αυτό όμως θα μπορούσε να αποφευχθεί με την χρήση μια δομής επανάληψης. Μια δομή επανάληψη ονομάζεται και βρόγχος (loop). Οι δομές επανάληψης διαχωρίζονται σε 3 κατηγορίες:

- ο βρόγχος **while**, ο οποίος επαναλαμβάνει μια πρόταση ή μια σειρά προτάσεων όσο ισχύει μια συνθήκη. Κάθε φορά που επαναλαμβάνεται ένας κύκλος εκτέλεσης δοκιμάζεται αν ισχύει η συνθήκη αυτή.
- ο βρόγχος **for**, κατά τον οποίο επαναλαμβάνεται μια σειρά προτάσεων για κάθε ένα στοιχείο μιας ακολουθίας δεδομένων (πχ λίστας, πλειάδας, λεξικού κτλ.).
- εμφωλευμένοι βρόγχοι, που ουσιαστικά πρόκειται για συνδυασμό βρόγχων while ή for. Έτσι μπορούμε να δομήσουμε ένα βρόγχο while μέσα σε έναν for, έναν for μέσα σε έναν while, έναν for μέσα σε έναν for και έναν while μέσα σε έναν while.

Παρακάτω δίνονται παραδείγματα για την κάθε κατηγορία.

Βρόγχος while

```
count = 0
while (count < 10):
    count = count + 1
    print("Hello World")
```

```
Hello World
```

Βρόγχος for

```
fruits = ["Μήλο", "Κεράσι", "Αχλάδι"]
for x in fruits:
    print(x)
```

```
Μήλο
Κεράσι
Αχλάδι
```

εμφωλευμένοι βρόγχοι (for μέσα σε for)

```
for i in range(1, 11):
    # nested loop
    # to iterate from 1 to 10
    for j in range(1, 11):
        # print multiplication
        print(i * j, end=' ')
```

```
1 2 3 4 5 6 7 8 9 10 2 4 6 8 10 12 14 16 18 20 3 6 9 12 15 18 21 24 27 30 4 8 12 16
20 24 28 32 36 40 5 10 15 20 25 30 35 40 45 50 6 12 18 24 30 36 42 48 54 60 7 14 21
28 35 42 49 56 63 70 8 16 24 32 40 48 56 64 72 80 9 18 27 36 45 54 63 72 81 90 10
20 30 40 50 60 70 80 90 100
```

Οι βρόγχοι for πρέπει να εκτελούν κάποια πρόταση. Σε διαφορετική περίπτωση αν θέλουμε να τους χρησιμοποιήσουμε χωρίς να κάνουν τίποτα (πχ για να συγγράψουμε αργότερα το περιεχόμενό τους) μπορούμε να χρησιμοποιήσουμε την πρόταση **pass**.

```
for x in [0, 1, 2]:
    pass
```

Μια χρήσιμη συνάρτηση που χρησιμοποιείται σε συνδυασμό με τους βρόγχους for είναι η συνάρτηση **range()**. Η συνάρτηση **range** δημιουργεί μια ακολουθία αριθμών και προσφέρει σχετικές παραμέτρους για την έναρξη και λήξη και το βήμα της ακολουθίας δηλαδή **range(start, stop, step_size)**.

```
range(0, 10)
```

```
range(0, 10)
```

Σε συνδυασμό με τον βρόγχο for μπορούμε να επαναλάβουμε μια σειρά προτάσεων κώδικα για μια ακολουθία αριθμών.

```
genre = ['Φυσικά', 'Μαθηματικά', 'Χημεία']

# iterate over the list using index
for i in range(len(genre)):
    print("Διαβάζω ", genre[i])
```

```
Διαβάζω  Φυσικά
Διαβάζω  Μαθηματικά
Διαβάζω  Χημεία
```

εμφωλευμένοι βρόγχοι (while μέσα σε for)

```

names = ['Νίκος', 'Κώστας', 'Ελένη']
# outer loop
for name in names:
    # inner while loop
    count = 0
    while count < 5:
        print(name, end=' ')
        # increment counter
        count = count + 1
    print()

```

Νίκος Νίκος Νίκος Νίκος Νίκος
 Κώστας Κώστας Κώστας Κώστας Κώστας
 Ελένη Ελένη Ελένη Ελένη Ελένη

εμφωλευμένοι βρόγχοι (while μέσα σε while)

```

i = 1
while i <= 4 :
    j = 0
    while j <= 3 :
        print(i*j, end=" ")
        j += 1
    print()
    i += 1

```

0 1 2 3
 0 2 4 6
 0 3 6 9
 0 4 8 12

Η πρόταση else μέσα σε βρόγχους

Όταν η συνθήκη βάσει της οποία ελέγχεται η εκτέλεση ενός βρόγχου while αποτυγχάνει (δεν είναι αληθής) τότε μπορούμε να εκτελέσουμε μια άλλη σειρά προτάσεων που ορίζεται από την ενότητα else. Για παράδειγμα:

```

count=0
while(count<5):
    print(count)
    count +=1
else:
    print("count value reached %d" %(count))

```

0
 1
 2
 3
 4
 count value reached 5

Αντίστοιχα σε έναν βρόγχο for μπορούμε να εκτελέσουμε μια ομάδα προτάσεων που ορίζεται από το else όταν έχει εξαντληθεί η προσπέλαση όλων των στοιχείων μιας ακολουθίας. Δείτε το παρακάτω παράδειγμα:

```

for x in range(6):
    print(x)
else:
    print("Finally finished!")

```

0
 1
 2
 3
 4
 5
 Finally finished!

Οι προτάσεις break και continue

Οι προτάσεις break και continue χρησιμοποιούνται για το έλεγχο της ροής των επαναλήψεων και λειτουργούν με τον ίδιο τρόπο τόσο στους βρόγχους while όσο και στους βρόγχους for.

Η πρόταση break τερματίζει άμεσα την εκτέλεση του βρόγχου με βάση τον έλεγχο μιας συνθήκης και στην συνέχεια ακολουθεί η εκτέλεση της πρότασης που ακολουθεί τον βρόγχο.

Παράδειγμα με βρόγχο for:

```
for num in range(1, 11):
    if num == 5:
        break
    else:
        print(num)

print("Τέλος")
```

```
1
2
3
4
Τέλος
```

Παράδειγμα με βρόγχο while:

```
n = 5
while n > 0:
    n -= 1
    if n == 2:
        break
    print(n)
print('Loop ended.')
```

```
4
3
Loop ended.
```

Αντίστοιχη είναι η λειτουργία της πρότασης continue. Ωστόσο σε αυτήν την περίπτωση ο βρόγχος δεν τερματίζεται εντελώς αλλά σταματάει την τρέχουσα σειρά εντολών στο τρέχον σημείο του βρόγχου και συνεχίζει στο επόμενο στοιχείο της ακολουθίας και στον έλεγχο της αντίστοιχης συνθήκης.

6. Συναρτήσεις (functions)

Τι είναι οι συναρτήσεις

Οι συναρτήσεις (functions) είναι μια επαναχρησιμοποιούμενη ομάδα εντολών η οποία εκτελείται μόνο όταν κληθεί. Στο σύνολο αυτό των εντολών δίνεται ένα όνομα με το οποίο είναι αναγνωρίσιμο. Μπορούμε να καλέσουμε μια συνάρτηση όσες φορές θέλουμε στο πρόγραμμά μας. Μια συνάρτηση μπορεί να δέχεται παραμέτρους δηλαδή δεδομένα εισόδου και να παράγει αποτελέσματα δηλαδή δεδομένα εξόδου. Με την χρήση συναρτήσεων επιτυγχάνεται, αρθρωτή δομή (modularity), λιγότερη επανάληψη κώδικα (code reusing), ευκολότερη αποσφαλμάτωση, αναγνωσιμότητα και ευκολία διόρθωσης. Πέρα από τις συναρτήσεις που μπορεί να συντάξει ο προγραμματιστής (user-defined functions), η Python προσφέρει ήδη έτοιμες συναρτήσεις (built-in functions).

Η βασική ιδέα με την χρήση των συναρτήσεων είναι να απομονώσουμε ένα κομμάτι κώδικα που επιτελεί πολύ συγκεκριμένη λειτουργία και να το καλούμε όταν μας είναι αναγκαίο. Για παράδειγμα αν σε ένα πρόγραμμα απαιτείται ο τακτικός υπολογισμός του Πυθαγόρειου θεωρήματος, αντί να επιλύουμε κάθε φορά την αντίστοιχη εξίσωση αρκεί να δημιουργήσουμε μια συνάρτηση που θα δέχεται σαν παραμέτρους τα μήκη δύο καθέτων πλευρών του ορθογωνίου τριγώνου και θα επιστρέψει το τετράγωνο της υποτείνουσας. Οπότε κάθε φορά που θα χρειάζεται η επίλυση του Πυθαγόρειου θεωρήματος αρκεί η κλήση της συνάρτησης και ο ορισμός των αντίστοιχων παραμέτρων σε αυτήν. Αν διαπιστώσουμε ότι ο μαθηματικός τύπος που έχουμε χρησιμοποιήσει είναι λάθος τότε αρκεί η διόρθωση μόνο σε ένα σημείο του κώδικα, σε αυτό της συνάρτησης. Αν αντιθέτως στο πρόγραμμά μας αντι για συνάρτηση είχαμε χρησιμοποιήσει ξεχωριστά την επίλυση της ίδιας εξίσωσης τότε θα έπρεπε να εντοπίσουμε και να διορθώσουμε όλα αυτά τα επιμέρους σημεία του κώδικα.

Τρόπος σύνταξης και κλήσης

Ο ορισμός μιας συνάρτησης στην Python ξεκινάει με την δεσμευμένη λέξη def, στην συνέχεια δίνεται ένα όνομα για την συνάρτηση, ακολουθούν οι παρενθέσεις (εντός τους μπορεί να ορίζονται και οι παράμετροι) και ακολουθεί η άνω/κάτω τελεία (colon). Στην συνέχει ακολουθεί (με εσοχές 4 space) μια σύντομη περιγραφή (docstrings) για το τι κάνει η συνάρτηση (προαιρετικά), και το σχετικό κομμάτι εντολών. Μπορεί κατά την ολοκλήρωση, επίσης προαιρετικά, η συνάρτηση να επιστρέψει κάποια τιμή μέσω της λέξης-κλειδί return. Σε κάθε περίπτωση όταν το return καλείται μέσα στην συνάρτηση

τότε αυτή σταματάει την εκτέλεση του κώδικα που περιέχει και επιστρέφει στο σημείο απ" όπου καλέστηκε. Η ονοματολογία των συναρτήσεων υπακούει στους κανόνες ονοματολογίας των μεταβλητών και συνήθως περιγράφουν τι κάνουν. Προηγείται ο ορισμός μια συνάρτησης και μετά η κλήση της. Γι" αυτό τον λόγο οι συναρτήσεις δηλώνονται πρώτα σε ένα πρόγραμμα και μετά ακολουθεί το κυρίως σώμα του κώδικα μέσα στον οποίο μπορούμε να τις καλέσουμε. Παρακάτω δίνεται μια απλουστευτική μορφή της σύνταξης μιας συνάρτησης.

```
def function_name(parameters):
    """docstring"""
    statement(s)
```

Ας δούμε ένα πιο πρακτικό παράδειγμα. Παρακάτω ακολουθεί η σύνταξη μιας απλής συνάρτησης όπου δεν ορίζονται κάποιοι παράμετροι ούτε επιστρέφεται κάποια τιμή.

```
def sayHello():
    """Μια συνάρτηση που χαιρετά τον κόσμο."""
    print("Hello World!")
```

Στην συνέχεια ακολουθεί η κλήση της όπου ουσιαστικά εκτελείται το σώμα της συνάρτησης. Η κλήση της γίνεται ορίζοντας το όνομά της και στην συνέχεια ακολουθούν παρενθέσεις:

```
sayHello()
```

```
Hello World!
```

Συνάρτηση με παραμέτρους

Σε αρκετές περιπτώσεις μια συνάρτηση μπορεί να περιλαμβάνει δεδομένα εισόδου ([παράμετροι](#)) τα οποία είναι χρήσιμα κατά την εκτέλεση του κώδικά της. Οι παράμετροι αυτοί ορίζονται κατά την σύνταξη της συνάρτησης, μέσα στις παρενθέσεις που ακολουθούν το όνομα της. Κατά την κλήση οι παράμετροι αυτοί λαμβάνουν τιμές ή μεταβλητές και ονομάζονται [ορίσματα](#).

```
def sayHelloUser(name):
    """
    Μια συνάρτηση που χαιρετά τον κάποιο.
    Παράμετροι:
        name(str): ένα όνομα
    Επιστρέψει:
        Τίποτα

    """
    print("Hello", name, "!")
```

Στην παραπάνω συνάρτηση με το ονόμα `sayHelloUser` ορίσαμε την παράμετρο `name` όπου την χρησιμοποιούμε στην συνέχεια στο κυρίως σώμα της συνάρτησης.

Στην συνέχεια καλούμε την συνάρτηση δίνοντας σαν όρισμα (τιμή στην παράμετρο) την συμβολοσειρά «Κώστας». Έτσι η παράμετρος `name` δέχεται όρισμα την τιμή «Κώστας».

```
sayHelloUser("Κώστας")
```

```
Hello Κώστας !
```

Αντίστοιχα μπορούμε να δώσουμε διαφορετική τιμή στο όρισμα ακόμα και μέσω μίας μεταβλητής:

```
onoma="Χρήστος"
sayHelloUser(onoma)
```

```
Hello Χρήστος !
```

Το παραπάνω μπορεί να γραφτεί και πιο ρητά κάτα πέρασμα του ορίσματος στην παράμετρο.

```
onoma="Ελένη"
sayHelloUser(name=onoma)
```

```
Hello Ελένη !
```

Η χρησιμότητα του docstring που αναγράφουμε σε μια συνάρτηση φαίνεται αν καλέσουμε την βοήθεια της Python αναφορικά με την συγκεκριμένη συνάρτηση. Με αυτόν τον τρόπο λαμβάνουμε πληροφορίες για την λειτουργία της χωρίς να χρειαστεί να ανατρέξουμε στην σύνταξή της.

```
help(sayHelloUser)
```

```
Help on function sayHelloUser in module __main__:  
  
sayHelloUser(name)  
    Μια συνάρτηση που χαιρετά τον κάποιο.  
    Παράμετροι:  
        name(str): ένα όνομα  
    Επιστρέφει:  
        Τίποτα
```

Εναλλακτικά μπορούμε να χρησιμοποιήσουμε το `__doc__` attribute της συνάρτησης.

```
print(sayHelloUser.__doc__)
```

```
Μια συνάρτηση που χαιρετά τον κάποιο.  
Παράμετροι:  
    name(str): ένα όνομα  
Επιστρέφει:  
    Τίποτα
```

Παρακάτω ορίζεται μια συνάρτηση με δύο παραμέτρους:

```
def add(num1, num2) :  
    """Add two numbers"""  
    num3 = num1 + num2  
    print("Το άθροισμα των αριθμών " + str(num1) + " και " +str(num2) + " είναι "  
+ str(num3))
```

Την καλούμε με τα σχετικά ορίσματα:

```
add(10, 7)  
  
# αντί τιμών πέρασμα μεταβλητών ως ορίσματα στην συνάρτηση  
a=2  
b=5  
  
add(a, b)  
  
# ή μεικτός τρόπος  
add(9, a)
```

```
Το άθροισμα των αριθμών 10 και 7 είναι 17  
Το άθροισμα των αριθμών 2 και 5 είναι 7  
Το άθροισμα των αριθμών 9 και 2 είναι 11
```

Ορίσματα Θέσης (Positional arguments)

Όταν συντάσσεται μια συνάρτηση με πολλές παραμέτρους κατά την κλήση της πρέπει να ορίσουμε και αντίστοιχα ορίσματα. Το πέρασμα αυτών των ορισμάτων γίνεται είτε α) κατά θέση (**positional arguments**) είτε β) με βάση την λέξη κλειδί (**keyword arguments**). Τα ορίσματα θέσης ορίζονται κατά σειρά με βάση τις αντίστοιχες παραμέτρους που έχουν οριστεί κατά την σύνταξη της συνάρτησης. Το παρακάτω παράδειγμα είναι πιο κατανοητό:

```
def car(style, color):  
    '''Information about a car'''  
    print("Ο τύπους του αυτοκινήτου είναι:", style)  
    print("Και έχει χρώματα:", color)  
  
car("Sedan", "μαύρο")
```

```
Ο τύπους του αυτοκινήτου είναι: Sedan  
και έχει χρώματα: μαύρο
```

Στην παραπάνω περίπτωση ορίσαμε μια function που εκτυπώνει το *style* και το *color* ενός αυτοκινήτου. Κατά την κλήση της περνάμε τα ορίσματα κατά θέση. Δηλαδή στην πρώτη θέση κατά την σύνταξη έχουμε την παράμετρο *style* που παίρνει το όρισμα «*Sedan*» και στην δεύτερη θέση έχουμε την παράμετρο *color* που παίρνει το όρισμα «μαύρο». Με τα ορίσματα θέσης πρέπει να είμαστε ακριβής στις τιμές που περνάμε σε κάθε γιατί αλλιώς μπορεί να έχουμε απροσδιόριστα αποτελέσματα. Δείτε το παρακάτω παράδειγμα:

```
car("κόκκινο", "SUV")
```

```
Ο τύπους του αυτοκινήτου είναι: κόκκινο  
και έχει χρώματα: SUV
```

Ορίσματα με βάση την λέξη-κλειδί (keyword arguments)

Τα ορίσματα κλειδιά είναι πιο ευέλικτα καθότι περνάμε ένα ζεύγος κλειδιού-τιμής. Έτσι ορίζουμε ρητά και ονομαστικά σε κάθε παράμετρο τι τιμή θα λάβει. Σε αυτή την περίπτωση δεν παίζει ρόλο η σειρά που περνάμε τιμές αλλά η αντιστοιχία ονομασία παραμέτρου με τιμή ορίσματος. Για αυτόν τον λόγο χρειάζεται προσοχή κατά την αντιστοιχία να χρησιμοποιούμε τα σωστά ονόματα παραμέτρων όπως διατυπώνονται στην σύνταξη της συνάρτησης. Ας δούμε το παραπάνω παράδειγμα με ορίσματα κλειδιά:

```
car(color="κόκκινο", style="SUV")  
car(style="SUV", color="κόκκινο", )
```

```
Ο τύπους του αυτοκινήτου είναι: SUV  
και έχει χρώματα: κόκκινο  
Ο τύπους του αυτοκινήτου είναι: SUV  
και έχει χρώματα: κόκκινο
```

Προκαθορισμένη τιμή παραμέτρου

Μια παράμετρο μπορεί να έχει μια προκαθορισμένη τιμή κατά την σύνταξη της συνάρτησης. Αν κατά την κλήση της συνάρτησης δώσουμε τιμή σε αυτήν την παράμετρο τότε αγνοείται η προκαθορισμένη τιμή. Όμως αν δεν δώσουμε τιμή τότε η παράμετρος κρατάει την προκαθορισμένη τιμή. Έτσι σε αρκετές περιπτώσεις όταν καλούμε μια συνάρτηση μπορεί οι προκαθορισμένες τιμές να αρκούν και να μην χρειάζεται να ορίσουμε τιμές στην αντίστοιχη παράμετρο. Ας δούμε το παρακάτω παράδειγμα:

```
def car(style, color, wheels="τέσσερις"):  
    '''Information about a car'''  
    print("Ο τύπους του αυτοκινήτου είναι:", style)  
    print("Έχει χρώμα", color, "και έχει", wheels, "τροχούς")
```

```
car(color="μπλέ", style="SUV")
```

```
Ο τύπους του αυτοκινήτου είναι: SUV  
Έχει χρώμα μπλέ και έχει τέσσερις τροχούς
```

Στο παραπάνω παράδειγμα δεν χρειάζεται να ορίσουμε τιμή για την παράμετρο *wheels* γιατί μας αρκεί η προκαθορισμένη, μιας και όλα σχεδόν τα επιβατικά έχουν τέσσερις τροχούς. Αντίστοιχα μπορεί να οριστούν και προκαθορισμένες τιμές και για τις άλλες παραμέτρους.

Βέβαια μπορούμε να αγνοήσουμε την προκαθορισμένη τιμή και να περάσουμε την αναγκαία κατά περίπτωση. π.χ. εδώ αγνοούμε την προκαθορισμένη τιμή για την παράμετρο *wheels* (τέσσερις) και ορίζουμε τιμή έξι.

```
car(color="μπλέ", wheels="έξι", style="SUV")
```

```
Ο τύπους του αυτοκινήτου είναι: SUV  
Έχει χρώμα μπλέ και έχει έξι τροχούς
```

Εναλλακτικά το προηγούμενο μπορεί να συνταχθεί με ορίσματα θέσης:

```
car("μπλέ", "SUV", "έξι")
```

Ο τύπους του αυτοκινήτου είναι: μπλέ
Έχει χρώμα SUV και έχει έξι τροχούς

Επιστροφή τιμής από μία συνάρτηση

Στα προηγούμενα παραδείγματα περιγράφεται η σύνταξη συναρτήσεων χωρίς να επιστρέφονται τιμές από αυτές. Στην συνέχεια θα δούμε πως μια συνάρτηση μπορεί να επιστρέψει τιμές μέσω της λέξης-κλειδιού `return`. Ας επεκτείνουμε το προηγούμενο παράδειγμα με την συνάρτηση που προσθέτει δύο αριθμούς. Η συνάρτηση αυτή μέχρι τώρα απλά δημιουργεί μια νέα μεταβλητή με το άθροισμα και στην συνέχεια εκτυπώνει ένα μήνυμα. Δεν επιστρέφει τίποτα. Αν δοκιμάσουμε να δούμε τι τύπο δεδομένων επιστρέφει θα δούμε ότι είναι `NoneType`.

```
type(add(1, 2))
```

Το άθροισμα των αριθμών 1 και 2 είναι 3

NoneType

Ας τροποποιήσουμε την συνάρτηση ώστε να επιστρέψει μία τιμή. Θα δώσουμε νέο όνομα στην συνάρτηση, `addV2`

```
def addV2(num1, num2) :  
    """Add two numbers"""  
    num3 = num1 + num2  
    return(num3)
```

Πλέον η συνάρτηση επιστρέφει το άθροισμα των τιμών που είναι ακέραιος τύπος δεδομένων. Το αποτέλεσμα της συνάρτησης που επιστρέφει το `return` μπορούμε να το προσαρτήσουμε σε μια μεταβλητή.

```
result=addV2(5,3)
```

Η συνάρτηση συνεχίζει να εκτυπώνει το μήνυμα που έχει οριστεί κατά την σύνταξη αλλά πλέον επιστρέφει και τιμή:

```
result
```

8

Την οποία μπορούμε να χρησιμοποιήσουμε σε άλλες προτάσεις του κώδικα. πχ

```
print("Το αποτέλεσμα της πράξης είναι:" + str(result))
```

Το αποτέλεσμα της πράξης είναι:8

ή μπορούμε να την καλέσουμε άμεσα:

```
print("Το αποτέλεσμα της πράξης είναι:" + str(addV2(9,9)))
```

Το αποτέλεσμα της πράξης είναι:18

Ο τύπος δεδομένων που επιστρέφει η συνάρτηση είναι ακέραιος (`int`):

```
type(addV2(1, 2))
```

int

Πολλαπλές επιστρεφόμενες τιμές

Σε ορισμένες περιπτώσεις συναρτήσεων μπορεί να απαιτούμε να επιστρέφονται παραπάνω από μία τιμής. Τότε χρησιμοποιούμε μια μέθοδος που λέγεται `tuple packing` κατά την οποία δημιουργούμε μια πλειάδα με τις αναγκαίες τιμές. Κατά την κλήση της συνάρτησης μπορούμε να προσαρτήσουμε το αποτέλεσμα που επιστρέφει σε αντίστοιχο πλήθος μεταβλητών (`tuple unpacking`).

```
def addV3(num1, num2) :  
    """Add two numbers"""  
    num3 = num1 + num2  
    return(num3, num3**2) #tuple packing
```

```
sum, squareofsum = addV3(4, 1) # tuple unpacking  
print("Sum:", sum, ", Square of sum:", squareofsum)
```

```
Sum: 5 , Square of sum: 25
```

Μια συνάρτηση μπορεί να επιστρέψει και ένα λεξικό ή μία λίστα.

Εμβέλεια μεταβλητών

Οι μεταβλήτες στην Python διαχωρίζονται με βάση την εμβέλεια τους (δηλαδή από ποιό σημείο του κώδικα είναι «օρατές»), σε **τοπικές (local)** και **καθολικές (global)**.

Καθολικές είναι οι μεταβλητές που ορίζονται στο κυρίως σώμα του κώδικα και δεν εντάσσονται μέσα σε κάποια συνάρτηση. Αυτές είναι προσπελάσιμες από κάθε σημείο του κώδικα ακόμα και μέσα από συναρτήσεις.

Τοπικές είναι οι μεταβλητές οι οποίες ορίζονται μέσα σε συναρτήσεις, είναι προσπελάσιμες μόνο μέσα σε αυτές και διαρκούν όσο διαρκεί η εκτέλεση μιας συνάρτησης. Κατά την πολλάπλη κλήση μιας συνάρτησης δημιουργούνται αντίστοιχες τοπικές μεταβλητές που περιγράφονται στην σύνταξή της. Οι παράμετροι μιας συνάρτησης αποτελούν και αυτές τοπικές μεταβλητές. Ας ορίσουμε μια τοπική μεταβλητή. Στην παρακάτω συνάρτηση ορίζεται η τοπική μεταβλητή `text`.

```
def PrintMyText():  
    text = "Athens"  
    print(text)  
  
PrintMyText()
```

```
Athens
```

Αν δοκιμάσουμε να προσπελάσουμε την μεταβλητή `text` εκτός συνάρτησης θα λάβουμε σχετικό σφάλμα:

```
print(text)
```

```
NameError  
Input In [29], in <cell line: 1>()  
----> 1 print(text)  
  
NameError: name 'text' is not defined
```

Όπως προαναφέρθηκε και μια παράμετρος αποτελεί τοπική μεταβλητή:

```
```{code-cell} ipython3  
def PrintMyText(p):
 print(p)

PrintMyText("Βόλος")
```

Αντιθέτως μια καθολική μεταβλητή είναι προσβάσιμη σε μία συνάρτηση.

```
city="Αριστο"
def PrintMyText():
 print("Η τιμή της μεταβλητής city ΕΝΤΟΣ της συνάρτησης", city) # η καθολική
 μεταβλητή είναι προσβάσιμη μέσα στην συνάρτηση

PrintMyText() # γι αύτό και εκτυπώνεται κατά την κλήση

print("Η τιμή της μεταβλητής city ΕΚΤΟΣ συνάρτησης", city) # και φυσικά είναι
διαθέσιμη και εκτός συνάρτησης από οποιοδήποτε σημείο του κώδικα
```

```
Η τιμή της μεταβλητής city ΕΝΤΟΣ της συνάρτησης Λάρισα
Η τιμή της μεταβλητής city ΕΚΤΟΣ συνάρτησης Λάρισα
```

Τι συμβαίνει όμως όταν μια μεταβλητή ορίζεται με την ίδια ονομασία σαν καθολική και τοπική; Δείτε το παρακάτω παράδειγμα:

```
city="Λάρισα"
def PrintMyText():
 city="Βόλος" # τοπική μεταβλητή, προτεραιότητα έναντι της καθολικής
 print("Η τιμή της μεταβλητής city ΕΝΤΟΣ της συνάρτησης (τοπική)", city) # εδώ
 εκτυπώνεται η τοπική μεταβλητή

PrintMyText() # θα εκτυπώσει την τοπική

print("Η τιμή της μεταβλητής city ΕΚΤΟΣ συνάρτησης (καθολική)", city) # θα
εκτυπώσει την καθολική
```

```
Η τιμή της μεταβλητής city ΕΝΤΟΣ της συνάρτησης (τοπική) Βόλος
Η τιμή της μεταβλητής city ΕΚΤΟΣ συνάρτησης (καθολική) Λάρισα
```

Αν θέλουμε να αλλάξουμε την τιμή μιας καθολικής μέσα σε μια συνάρτηση χρησιμοποιούμε την λέξη κλειδί **global** για να αναφερθούμε σε αυτήν.

```
Αυτή η συνάρτηση θα τροποποιήσει την καθολική μεταβλητή city
def printCity():
 global city
 city += ', πρωτεύουσα της Ελλάδας'
 print(city)

 city = "Βόλος"
 print(city)

Καθολική εμβέλεια
city= "Αθήνα"
printCity()
print(city)
```

```
Αθήνα, πρωτεύουσα της Ελλάδας
Βόλος
Βόλος
```

## Συναρτήσεις lambda

Οι συναρτήσεις *lambda* είναι μικρές ανώνυμες συναρτήσεις που μπορούν να έχουν πολλά ορίσματα αλλά μία έκφραση. Για παράδειγμα:

```
x = lambda a : a ** 2
x(5)
```

25

Σε αυτό το παράδειγμα ορίζεται μια συνάρτηση *lambda* που επιστρέφει το τετράγωνο ενός αριθμού. Στο παρακάτω παράδειγμα μια συνάρτηση *lambda* υπολογίζει το γινόμενο δύο αριθμών.

```
x = lambda a, b : a * b
x(2,3)
```

6

Άλλο παράδειγμα

```
onomatoponymo = lambda onoma, epitheto, birth: f'Όνοματεπώνυμο: {ονόμα.title()}
{επίθετο.title()}, Έτος γέννησης: {birth}'
onomatoponymo('αριστοτέλης', 'ωνάσης', 1906)
```

```
'Όνοματεπώνυμο: Αριστοτέλης Ωνάσης, Έτος γέννησης: 1906'
```

Πρακτική εφαρμογή της συνάρτησης *lambda* όπου χρησιμοποιείται για να φιλτραριστούν οι τιμές μιας λίστας με βάση ένα κριτήριο.

```
Program to filter out only the even items from a list
nums = [1, 2, 9, 10, 18, 31, 53, 120]

nums_filtered = list(filter(lambda x: x >= 10 , nums))

print(nums_filtered)
```

```
[10, 18, 31, 53, 120]
```

Στο παρακάτω παράδειγμα χρησιμοποιείται μια *lambda* συνάρτηση σε συνδυασμό με την συνάρτηση *map* που στόχο έχει πολλάπλασιάσει x2 τα στοιχεία μας λίστας.

```
Program to double each item in a list using map()

nums = [1, 2, 9, 10, 18, 31, 53, 120]

nums_squared = list(map(lambda x: x * 2 , nums))

print(nums_squared)
```

```
[2, 4, 18, 20, 36, 62, 106, 240]
```

## 7. Ανάγνωση & εγγραφή αρχείων, μετονομασία, αναζήτηση, αντιγραφή μετακίνηση αρχείων και καταλογών

Πριν ξεκινήσουμε την επίδειξη των εντολών για το σημερινό μάθημα, θα πρέπει να εισάγουμε μερικές απαραίτητες βιβλιοθήκες και να ορίσουμε τον τρέχων κατάλογο στη Python:

```
import os
import pickle
from pathlib import Path
from datetime import datetime, timezone
import fnmatch
import tempfile
from tempfile import TemporaryFile, TemporaryDirectory
import shutil

print(os.getcwd()) # εκτύπωση τρέχοντος καταλόγου στην Python

path="../some_directory/"
os.chdir(path) # ορισμός τρέχοντος καταλόγου

print(os.getcwd()) # επιβεβαίωση του τρέχοντος καταλόγου στην Python

/home/leonidas/Documents/uth/Programming/JupyterNotebooks/notes/notebooks
/home/leonidas/Documents/uth/Programming/JupyterNotebooks/notes/notebooks/some_directory
```

### Ανάγνωση & εγγραφή αρχείων

Μπορούμε να ανοίξουμε ένα αρχείο με την μέθοδο *open*, να δούμε μερικές ιδιότητες του και να το κλείσουμε

```
myfile=open("foo.txt", "r")
print ("Name of the file: ", myfile.name)
print ("Closed or not : ", myfile.closed)
print ("Opening mode : ", myfile.mode)
myfile.close()
print(myfile.closed)
```

```
Name of the file: foo.txt
Closed or not : False
Opening mode : r
True
```

Με την μέθοδο `write()` σε ένα object ανοικτού αρχείου (TextIOWrapper object πιο συγκεκριμένα) μπορούμε να γράψουμε σε αυτό περιεχόμενο. Κατά το `open` δίνουμε σαν παράμετρους το όνομα του αρχείου και την επιλογή προσπέλασης «w» (όπου w= εγγραφή, r= ανάγνωση, a=προσθήκη). Το w μας επιτρέπει την δυνατότητα εγγραφής. Πάντα πρέπει να κλείνουμε το ανοικτό αρχείο με την μέθοδο `close()`.

```
Open a file
fo = open("foo.txt", "w") # Σημαντική παράμετρος το "w"
fo.write("Ενα ταξίδι χιλιών χιλιομέτρων αρχίζει με ένα βήμα.\nΠλάστε, διαβάστε, συναντήστε")
Close opened file
fo.close()
```

Επιβεβαιώνουμε ότι ίντως έγινε η εγγραφή ξαναδιαβάζοντας το αρχείο με την μέθοδο `read()` η οποία διαβάζει το συνολικό περιεχόμενο από το αρχείο.

```
Open a file
fo = open("foo.txt", "r") # Σημαντική παράμετρος το "w"

text = fo.read()
print (text)

Close opened file
fo.close()
```

Ένα ταξίδι χιλίων χιλιομέτρων αρχίζει με ένα βήμα. Λόγο Τσε, δος αιώνας π.χ., Κινέζος φιλόσοφος

Επειδή υπάρχει ο κίνδυνος να καλέσουμε την μέθοδο `close()` σε ένα ανοικτό αρχείο, μπορούμε εναλλακτικά να ανοίξουμε ένα αρχείο με το `with`. Σε αυτήν την περίπτωση το αρχείο κλείνει αυτόματα όταν ολοκληρωθεί το μπλοκ εντολών `with`.

```
with open('foo.txt', 'r') as reader:
 print(reader.read())
```

Ένα ταξίδι χιλίων χιλιομέτρων αρχίζει με ένα βήμα.  
Λάση Τσε, 60ς αιώνας π.Χ., Κινέζος φιλόσοφος

Η μέθοδος `read()` δέχεται σαν όρισμα τον αριθμό των bytes που θα επιστρέψει.

```
with open('foo.txt', 'r') as fo:
 text = fo.read(5)
 print (text)

 text = fo.read(10)
 print (text)
```

Ένα τ  
αξίδι χιλι

Με την χρήση της μεθόδου `readlines()` προσαρτούμε κάθε γραμμή που υπάρχει στο αρχείο σε μια λίστα. Όμως ο χαρακτήρας που ορίζει την νέα σειρά (`\n`) δεν αγνοείται από την ανάγνωση.

```
with open('dog_breeds.txt', 'r') as reader:
 # Note: readlines doesn't trim the line endings
 dog_breeds = reader.readlines()
 dog_breeds = [line.rstrip() for line in dog_breeds] # μπορούμε να αφαιρέσουμ
τα new line characters με αυτόν τον τρόπο
 print(dog_breeds)
```

Με την χρήση του `with` μπορούμε να εγγράψουμε δεδομένα κιόλας. Σημαντική παράμετρος το `"w"` κατά το `open()` και η μεθόδος `write()`.

```
with open('dog_breeds_reversed.txt', 'w') as writer:
 # Alternatively you could use
 # writer.writelines(reversed(dog_breeds))

 # Write the dog breeds to the file in reversed order
 for breed in reversed(dog_breeds):
 writer.write(breed)
```

Με το όρισμα "a" (append) κατά το άνοιγμα ενός αρχείου μπορούμε να εγγράψουμε σε ένα αρχείο χωρίς να διαγραφεί το προηγούμενο περιεχόμενο.

```
with open('dog_breeds.txt', 'a') as a_writer:
 a_writer.write('Beagle\n')
```

```
with open('dog_breeds.txt', 'r') as reader:
 print(reader.read())
```



```
d_path = 'dog_breeds.txt'
d_r_path = 'dog_breeds_reversed.txt'
with open(d_path, 'r') as reader, open(d_r_path, 'w') as writer:
 dog_breeds = reader.readlines()
 writer.writelines(reversed(dog_breeds))
```

Ανάγνωση αρχείου ανά γραμμή μέσω βρόγχου `while` και της μεθόδου `readline()`:

```
with open('dog_breeds.txt', 'r') as reader:
 # Read and print the entire file line by line
 line = reader.readline()
 while line != '':
 # The EOF char is an empty string
 print(line, end='')
 line = reader.readline()
```

Μπορούμε να διαβάσουμε γραμμή - γραμμή το περιεχόμενο ενός αρχείου μέσω ενός loop στην λίστα γραμμών που μας προσφέρει η μέθοδος `readlines()`

```
with open('dog_breeds.txt', 'r') as reader:
 for line in reader.readlines():
 print(line, end='')
```

Και για περισσότερο ευανάγνωστο κώδικα η Python μας δίνει την δυνατότητα να κάνουμε loop μέσω του reader object

```
with open('dog_breeds.txt', 'r') as reader:
 # Read and print the entire file line by line
 for line in reader:
 print(line, end='') # use end='' to avoid new line after each print
 statement
```



Στα προηγούμενα παραδείγματα διαβάσαμε ή γράψαμε συμβολοσειρές σε ένα αρχείο. Όμως, μέσω της σειριοποίησης μπορούμε να αποθηκεύσουμε σε ένα binary αρχείο τα αντικείμενα της python με τις ιδιότητές τους. Όχι μόνον σαν απλές συμβολοσειρές. Στο παρακάτω κομμάτι κώδικα θα αποθηκεύσουμε μία λίστα και μια μεταβλήτη σε ένα αρχείο με την βοήθεια της βιβλιοθήκης *pickle*.

```
mylist=['one', 2 , 'tree']
pi=3.14
with open('pickle.txt', 'ab') as pickle_writer:
 pickle.dump(mylist, pickle_writer)
 pickle.dump(pi, pickle_writer)
```

Αφού έχουμε αποθηκεύσει τα σχετικά object σε ένα αρχείο, μπορούμε σε μεταγενέστερα στάδια του κώδικα να τα ανακαλέσουμε αυτούσια μέσω της ανάγνωσης αυτού του αρχείου.

```
with open('pickle.txt', 'rb') as pickle_read:
 pickle.load(pickle_read)
 print(pi)
 print(mylist)
```

```
3.14
['one', 2, 'tree']
```

## Ανάκτηση περιεχομένων φακέλου

Μέσω της μεθόδου *os.scandir()* μπορούμε να λάβουμε μια λίστα με τα αρχεία και τους φακέλους σε ένα κατάλογο.

```
entries = os.scandir('.')
for entry in entries:
 print(entry.name)
```

```
sub_dir2
sub_dir
dog_breeds_reversed.txt
sub_dir3
data_02.txt
foo.txt
data_02_backup.txt
data_03_backup.txt
data_01_backup.txt
dog_breeds.txt
2018
example_directory
pickle.txt
file1.py
data_03.txt
data_01.txt
tests.py
admin.py
admin2.py
```

```
εναλλακτικά
os.listdir('./')
```

```
['sub_dir2',
 'sub_dir',
 'dog_breeds_reversed.txt',
 'sub_dir3',
 'data_02.txt',
 'foo.txt',
 'data_02_backup.txt',
 'data_03_backup.txt',
 'data_01_backup.txt',
 'dog_breeds.txt',
 '2018',
 'example_directory',
 'pickle.txt',
 'file1.py',
 'data_03.txt',
 'data_01.txt',
 'tests.py',
 'admin.py',
 'admin2.py']
```

```
Εναλλακτικά μέσω πρόταση with
with os.scandir('.') as entries:
 for entry in entries:
 print(entry.name)
```

```
sub_dir2
sub_dir
dog_breeds_reversed.txt
sub_dir3
data_02.txt
foo.txt
data_02_backup.txt
data_03_backup.txt
data_01_backup.txt
dog_breeds.txt
2018
example_directory
pickle.txt
file1.py
data_03.txt
data_01.txt
tests.py
```

```
admin.py
admin2.py
```

Το ίδιο μπορούμε να κάνουμε με την μέθοδο `iterdir()` σε ένα Path object από την βιβλιοθήκη `pathlib`

```
entries = Path('.')
for entry in entries.iterdir():
 print(entry.name)
```

```
sub_dir2
sub_dir
dog_breeds_reversed.txt
sub_dir3
data_02.txt
foo.txt
data_02_backup.txt
data_03_backup.txt
data_01_backup.txt
dog_breeds.txt
2018
example_directory
pickle.txt
file1.py
data_03.txt
data_01.txt
tests.py
admin.py
admin2.py
```

Η μέθοδος `is_file()` μας επιτρέπει να τεστάρουμε αν ένα αντικείμενο τύπου Path είναι αρχείο.

```
basepath = Path('.')
files_in_basepath = basepath.iterdir()
for item in files_in_basepath:
 if item.is_file():
 print(item.name)
```

```
dog_breeds_reversed.txt
data_02.txt
foo.txt
data_02_backup.txt
data_03_backup.txt
data_01_backup.txt
dog_breeds.txt
pickle.txt
file1.py
data_03.txt
data_01.txt
tests.py
admin.py
admin2.py
```

Επιπλέον για ένα path μπορούμε να λάβουμε την απόλυτη μορφή του (absolute path) μέσω της συνάρτησης `os.path.abspath()` τον κατάλογο στον οποίο βρίσκεται (μέσω της συνάρτησης `dirname()`) και το όνομα του αρχείου (με `os.path.basename()`)

```
path="./foo.txt" # relative path
abspath = os.path.abspath(path) # absolute path
print(abspath)
print(os.path.dirname(abspath)) # όνομα καταλόγου που βρίσκεται το αρχείο
print(os.path.basename(abspath)) # όνομα αρχείου (filename)
```

```
/home/leonidas/Documents/uth/Programming/JupyterNotebooks/notes/notebooks/some_directory/foo.txt
/home/leonidas/Documents/uth/Programming/JupyterNotebooks/notes/notebooks/some_directory
foo.txt
```

Η μέθοδος `is_dir()` μας επιτρέπει να τεστάρουμε αν ένα αντικείμενο τύπου Path είναι φάκελος (directory).

```
List all subdirectory using pathlib
basepath = Path('.')
for entry in basepath.iterdir():
 if entry.is_dir():
 print(entry.name)
```

```
sub_dir2
sub_dir
sub_dir3
2018
example_directory
```

Με την μέθοδο `stat()` μπορούμε να δούμε χρήσιμες λεπτομέρειες για ένα αρχείο και να ανακτήσουμε δεδομένα όπως το μέγεθος του, το όνομά του και η τελευταία ημερομηνία/ώρα τροποποίησης (δίνεται σε seconds από την 1/1/1970)

```
current_dir = Path('./') #ορίστε έναν φάκελο. Στην συγκεκριμένη περίπτωση ορίζεται ο τρέχων κατάλογος.
for path in current_dir.iterdir():
 info = path.stat()

 size=info.st_size
 modification_time=datetime.fromtimestamp(info.st_mtime, tz=tzzone.utc) #
 st_time= the number of seconds passed since 1st January 1970 (epoch)
 name=path.name

 print("\nΌνομα αρχείου:", name)
 print("\tΜέγεθος: ",size) # το μέγεθος του αρχείου σε bytes
 print("\tΗμερομηνία τελευταίας τροποποίησης", modification_time)
```

Όνομα αρχείου: sub\_dir2  
Μέγεθος: 4096  
Ημερομηνία τελευταίας τροποποίησης 2023-02-19 17:12:58.796885+00:00

Όνομα αρχείου: sub\_dir  
Μέγεθος: 4096  
Ημερομηνία τελευταίας τροποποίησης 2023-02-19 17:12:58.796885+00:00

Όνομα αρχείου: dog\_breeds\_reversed.txt  
Μέγεθος: 693  
Ημερομηνία τελευταίας τροποποίησης 2023-05-23 14:00:57.796534+00:00

Όνομα αρχείου: sub\_dir3  
Μέγεθος: 4096  
Ημερομηνία τελευταίας τροποποίησης 2023-02-19 17:12:58.796885+00:00

Όνομα αρχείου: data\_02.txt  
Μέγεθος: 0  
Ημερομηνία τελευταίας τροποποίησης 2023-02-19 17:12:58.796885+00:00

Όνομα αρχείου: foo.txt  
Μέγεθος: 168  
Ημερομηνία τελευταίας τροποποίησης 2023-05-23 14:00:57.712534+00:00

Όνομα αρχείου: data\_02\_backup.txt  
Μέγεθος: 0  
Ημερομηνία τελευταίας τροποποίησης 2023-02-19 17:12:58.796885+00:00

Όνομα αρχείου: data\_03\_backup.txt  
Μέγεθος: 0  
Ημερομηνία τελευταίας τροποποίησης 2023-02-19 17:12:58.796885+00:00

Όνομα αρχείου: data\_01\_backup.txt  
Μέγεθος: 0  
Ημερομηνία τελευταίας τροποποίησης 2023-02-19 17:12:58.796885+00:00

Όνομα αρχείου: dog\_breeds.txt  
Μέγεθος: 693  
Ημερομηνία τελευταίας τροποποίησης 2023-05-23 14:00:57.776534+00:00

Όνομα αρχείου: 2018  
Μέγεθος: 4096  
Ημερομηνία τελευταίας τροποποίησης 2022-05-13 18:28:15.820793+00:00

Όνομα αρχείου: example\_directory  
Μέγεθος: 4096  
Ημερομηνία τελευταίας τροποποίησης 2022-05-13 18:28:15.800793+00:00

Όνομα αρχείου: pickle.txt  
Μέγεθος: 3828  
Ημερομηνία τελευταίας τροποποίησης 2023-05-23 14:00:57.884534+00:00

Όνομα αρχείου: file1.py  
Μέγεθος: 0  
Ημερομηνία τελευταίας τροποποίησης 2023-02-19 17:12:58.796885+00:00

Όνομα αρχείου: data\_03.txt  
Μέγεθος: 0  
Ημερομηνία τελευταίας τροποποίησης 2023-02-19 17:12:58.796885+00:00

Όνομα αρχείου: data\_01.txt  
Μέγεθος: 0  
Ημερομηνία τελευταίας τροποποίησης 2023-02-19 17:12:58.796885+00:00

Όνομα αρχείου: tests.py  
Μέγεθος: 0  
Ημερομηνία τελευταίας τροποποίησης 2023-02-19 17:12:58.796885+00:00

Όνομα αρχείου: admin.py  
Μέγεθος: 0  
Ημερομηνία τελευταίας τροποποίησης 2023-02-19 17:12:58.796885+00:00

Όνομα αρχείου: admin2.py  
Μέγεθος: 0  
Ημερομηνία τελευταίας τροποποίησης 2023-05-23 13:54:00.581575+00:00

Μέσω του Path μπορούμε να φτιάξουμε και διαδρομές προς ένα κατάλογο του δίσκου μας όπως περιγράφεται παρακάτω:

```
δημιουργία paths

in_file_1 = Path.cwd() / "in" / "input.xlsx"
out_file_1 = Path.cwd() / "out" / "output.xlsx"

print(in_file_1)
print(out_file_1)

ή

in_file_2 = Path.cwd().joinpath("in").joinpath("input.xlsx")
out_file_2 = Path.cwd().joinpath("out").joinpath("output.xlsx")
```

```
/home/leonidas/Documents/uth/Programming/JupyterNotebooks/notes/notebooks/some_directory/in/input.xlsx
/home/leonidas/Documents/uth/Programming/JupyterNotebooks/notes/notebooks/some_directory/out/output.xlsx
```

## Δημιουργία καταλόγων

Στόχος είναι η δημιουργία νέων καταλόγων (directories) στο σύστημα αρχείων του υπολογιστή μας.

Πριν ξεκινήσουμε, επιβεβαιώνουμε για μια ακόμη φορά τον τρέχοντα κατάλογο

```
path = Path.cwd() # pathlib object, εναλλακτικό του os.getcwd()

print(str(path)) # print σαν συμβολοσειρά
```

```
/home/leonidas/Documents/uth/Programming/JupyterNotebooks/notes/notebooks/some_directory
```

Μπορούμε να δημιουργήσουμε καταλόγους με την μέθοδο `mkdir()`

```
create directory
try:
 p = Path('example_directory/') # ορισμός absolute ή relative Path
 p.mkdir() # δημιουργία καταλόγου
except FileExistsError:
 print(f"Ο κατάλογος {str(p)} υπάρχει ήδη")
```

Ο κατάλογος `example_directory` υπάρχει ήδη

```
p.mkdir(exist_ok=True) # δημιουργία καταλόγου, αγνοεί την δημιουργία φακέλου αν αυτός υπάρχει ήδη
```

```
p = Path('2018/10/05') # δημιουργία καταλόγου 05 και όλων των γονικών (parent) καταλόγων
p.mkdir(parents=True, exist_ok=True)
```

Είτε να αναζητήσουμε αρχεία και καταλόγους που περιλαμβάνουν συγκεκριμένους χαρακτήρες στο ονομά τους

```
content = os.listdir('.')
print(content)

print("Εύρεση αρχείων με καταληξη txt\n")

for file_name in content:
 if fnmatch.fnmatch(file_name, '*.txt'):
 print(file_name)
```

```
['sub_dir2', 'sub_dir', 'dog_breeds_reversed.txt', 'sub_dir3', 'data_02.txt',
'foo.txt', 'data_02_backup.txt', 'data_03_backup.txt', 'data_01_backup.txt',
'dog_breeds.txt', '2018', 'example_directory', 'pickle.txt', 'file1.py',
'data_03.txt', 'data_01.txt', 'tests.py', 'admin.py', 'admin2.py']
```

Εύρεση αρχείων με καταληξη txt

```
dog_breeds_reversed.txt
data_02.txt
foo.txt
data_02_backup.txt
data_03_backup.txt
data_01_backup.txt
dog_breeds.txt
pickle.txt
data_03.txt
data_01.txt
```

Εύρεση όλων των αρχείων στον τρέχοντα κατάλογο που το όνομά τους έχει την παρακάτω μορφή:

```
data_*_backup.txt
```

το αστεράκι (\*) αντιπροσωπεύει οποιοδήποτε αριθμό χαρακτήρων μέσα στο όνομα.

```
for filename in os.listdir('.'):
 if fnmatch.fnmatch(filename, 'data_*_backup.txt'):
 print(filename)
```

```
data_02_backup.txt
data_03_backup.txt
data_01_backup.txt
```

Εναλλακτικά με την χρήση της μεθόδου glob()

```
p = Path('.')
for name in p.glob('data_*_backup.txt'):
 print(name)
```

```
data_02_backup.txt
data_03_backup.txt
data_01_backup.txt
```

```
p = Path('.')
for name in p.glob('*[0-9]*backup.txt'):
 print(name)
```

```
data_02_backup.txt
data_03_backup.txt
data_01_backup.txt
```

Οι παραπάνω αναζητήσεις αφορούσαν το περιεχόμενο μόνο στον τρέχοντα κατάλογο και όχι ταυτόχρονα και στους υποκαταλόγους (child) που υπάρχουν μέσα σε αυτόν. Για να αναζητήσουμε διαδοχικά και σε αυτούς τους καταλόγους χρησιμοποιούμε το παρακάτω πρόθεμα πριν από το κριτήριο αναζήτησης \*\*/

```
Append "**/" before the search term in pattern to recursively search this
directory
p = Path('.')
for name in p.glob('**/*.py'):
 print(name)
```

```
file1.py
tests.py
admin.py
admin2.py
sub_dir2/file2.py
sub_dir2/file1.py
sub_dir/file2.py
sub_dir/file1.py
sub_dir3/file2.py
sub_dir3/file1.py
```

```
αναζήτηση για ότι περιέχει f και 1 στο filename
p = Path('.')
for name in p.glob('**/f*1*'):
 print(name)
```

```
file1.py
sub_dir2/file1.py
sub_dir/file1.py
sub_dir3/file1.py
```

Για να ανατρέξουμε διαδοχικά σε όλους του φακέλους ενός καταλόγου χρησιμοποιούμε την μέθοδο os.walk

```
walk
Walking a directory tree and printing the names of the directories and files
for dirpath, dirnames, files in os.walk('.'):
 print(f'Found directory: {dirpath}')
 for file_name in files:
 print(file_name)
```

```
Found directory: ./sub_dir2
file2.py
file1.py
Found directory: ./sub_dir
file2.py
file1.py
Found directory: ./sub_dir3
file2.py
file1.py
Found directory: ./2018/10/05
Found directory: ./2018/10
Found directory: ./2018
Found directory: ./example_directory
Found directory: .
dog_breeds_reversed.txt
data_02.txt
foo.txt
data_02_backup.txt
data_03_backup.txt
data_01_backup.txt
dog_breeds.txt
pickle.txt
file1.py
data_03.txt
data_01.txt
tests.py
admin.py
admin2.py
```

## Προσωρινά αρχεία και κατάλογοι

Με την python μπορούμε να δημιουργήσουμε προσωρινά αρχεία και καταλόγους οι οποιού παύουν να υπάρχουν μετά την εκτέλεση του κώδικα. Αυτό γίνεται με την βοήθεια του αρθρώματος (module) tempfile και της συνάρτησης TemporaryFile() και TemporaryDirectory()

```
from tempfile import TemporaryFile

temporary files
with TemporaryFile('w+t') as fp: # άνοιγμα προσωρινού αρχείου για εγγραφή
 fp.write('Hello universe!')
 fp.seek(0)
 print(fp.read())
File is now closed and removed
```

```
Hello universe!
```

```
with TemporaryDirectory() as tmpdir:
 print('Created temporary directory ', tmpdir)
 print(tmpdir)
 print(os.path.exists(tmpdir))

Directory contents have been removed
```

```
Created temporary directory /tmp/tmp1xb7en5
/tmp/tmp1xb7en5
True
```

Αφού έχουμε εξέλθει από τον block κώδικα *with* το προσωρινό directory παύει να υπάρχει

```
os.path.exists(tmpdir)
```

```
False
```

## Διαγραφή αρχείων και φακέλων

Με την μέθοδο unlink() μπορούμε να διαγράψουμε έναν **άδειο** κατάλογο ή ένα αρχείο.

```
data_file = Path('./data_04.txt')

if data_file.is_file():
 print ("Το αρχείο υπάρχει και θα διαγραφεί")
 data_file.unlink()
else:
 print ("Το αρχείο δεν υπάρχει")
```

```
Το αρχείο δεν υπάρχει
```

Για την διαγραφή άδειου φακέλου καλούμε την μέθοδο rmdir() στο Path object.

```
my_dir = Path('./tmp')

if my_dir.is_dir():
 print ("Το directory υπάρχει και θα διαγραφεί")
 my_dir.rmdir()
else:
 print ("Το directory δεν υπάρχει")
```

```
Το directory δεν υπάρχει
```

Αν θέλουμε να διαγράψουμε έναν κατάλογο ο οποίος περιλαμβάνει και περιεχόμενα τότε χρησιμοποιούμε η συνάρτηση rmtree() από την βιβλιοθήκη shutil.

```
διαγραφή φακέλου με περιεχόμενα

dest = Path('./tmp2')
shutil.rmtree(dest, ignore_errors=True)
```

## Αντιγραφή αρχείων και φακέλων

Ταυτόχρονα μπορούμε να αντιγράψουμε αρχεία με την συνάρτηση copy() πάλι από το άρθρωμα shutil.

```
Αντιγραφή αρχείου

src = 'admin.py'
dst = 'admin2.py'
shutil.copy(src, dst)
```

```
'admin2.py'
```

ή ολόκληρους καταλόγους μέσω της συνάρτησης copytree() πάλι από το ίδιο άρθρωμα.

```
try:
 shutil.copytree('sub_dir', 'sub_dir3')
except:
 print("Σφάλμα κατά την αντιγραφή")
```

```
Σφάλμα κατά την αντιγραφή
```

## Μετακίνηση

ή ακόμα και να μετακινήσουμε αρχεία και καταλόγους με την συναρτηση move()

```
try:
 shutil.move('data_04.txt', 'sub_dir/data_04.txt')
except FileNotFoundError:
 print("File does not exist")
```

File does not exist

```
μετακίνηση αρχείου και μάλιστα με μετονομασία κατά την μετακίνηση data_04.txt ->
data_05.txt

try:
 shutil.move('sub_dir/data_04.txt', 'data_05.txt')
except FileNotFoundError:
 print("File does not exist")
```

File does not exist

```
μετακίνηση φακέλου

try:
 shutil.move('tmp2', 'tmp/tmp2')
except FileNotFoundError:
 print("Directory does not exist")
```

Directory does not exist

```
επιστροφή στην θέση του
try:
 shutil.move('tmp/tmp2','tmp2')
except FileNotFoundError:
 print("Directory does not exist")
```

Directory does not exist

## Μετονομασία

Με την χρήση της μεθόδου rename() σε έναν Path αντικείμενο μπορούμε να το μετονομάσουμε.

```
αρχείου
data_file = Path('data_01.txt')
data_file.rename('data.txt')
```

PosixPath('data.txt')

```
ξανά όπως ήταν
data_file = Path('data.txt')
data_file.rename('data_01.txt')
```

PosixPath('data\_01.txt')

## Βιβλιογραφία

- Αγγελιδάκης, N., 2015. Εισαγωγή στον προγραμματισμό με την Python. Αγγελιδάκης, Ηράκλειο.
- Working With Files in Python, <https://realpython.com/read-write-files-python/>, Πρόσβαση: 13/05/2022
- Reading and Writing Files in Python, <https://realpython.com/read-write-files-python/>, Πρόσβαση: 13/05/2022

## 8. Ανάγνωση αρχείων csv, η βιβλιοθήκη pandas

Στόχος του μαθήματος είναι να εξοικειωθεί ο φοιτητής με την ανάγνωση και εγγραφή αρχείων CSV. Επίσης γίνεται μια σύντομη αναφορά στις δυνατότητες της βιβλιοθήκης pandas όσον αφορά την διαχείριση δεδομένων σε μορφή πινάκων.

## Ανάγνωση αρχείων CSV

Εισάγουμε τα απαραίτητα άρθρωματα (modules).

```
import os # μας επιτρέπει να έχουμε αλληλεπίδραση με το λειτουργικό σύστημα
import csv # απαραίτητο για την ανάγνωση και εγγραφή αρχείων CSV
from urllib import request # απαραίτητο για την λήψη αρχείων από το διαδίκτυο
import pandas as pd # η βιβλιοθήκη pandas για την διαχείρηση πινάκων
from pathlib import Path # βιβλιοθήκη για την διαχείριση των διαδρομών (paths)
στον δίσκο
import numpy as np
```

Προαιρετικά ελέγχουμε ποιος είναι ο τρέχων κατάλογος:

```
print(os.getcwd())
```

```
/home/leonidas/Documents/uth/Programming/JupyterNotebooks/notes/notebooks
```

Αρχικά κατεβάζουμε από το διαδίκτυο τα αναγκαία αρχεία CSV.

(Πηγή δεδομένων <https://www.kaggle.com/datasets/rinichristy/covid19-coronavirus-pandemic>)

```
Πρώτο αρχείο
remote_url =
'https://raw.githubusercontent.com/kokkytos/programming/main/docs/COVID-
19%20Coronavirus.csv'
Define the local filename to save data
local_file1 = 'COVID-19_Coronavirus.csv'
Download remote and save locally
request.urlretrieve(remote_url, local_file1)

Δεύτερο αρχείο
remote_url =
'https://raw.githubusercontent.com/kokkytos/programming/main/docs/COVID-
19%20Coronavirus_V2.csv'
Define the local filename to save data
local_file2 = 'COVID-19_Coronavirus_V2.csv'
Download remote and save locally
request.urlretrieve(remote_url, local_file2)
```

```
('COVID-19_Coronavirus_V2.csv', <http.client.HTTPMessage at 0x7f4b667d5630>)
```

```
with open(local_file1, 'r') as file:
 reader = csv.reader(file)
 for row in reader:
 print(row)
```

```
['Country', 'Other names', 'ISO 3166-1 alpha-3 CODE', 'Population', 'Continent',
'Total Cases', 'Total Deaths', 'Tot\xa0Cases//1M pop', 'Tot\xa0Deaths/1M pop',
'Death percentage']
['Afghanistan', 'Afghanistan', 'AFG', '40462186', 'Asia', '177827', '7671', '4395',
'190', '4.313743132']
['Albania', 'Albania', 'ALB', '2872296', 'Europe', '273870', '3492', '95349',
'1216', '1.275057509']
['Algeria', 'Algeria', 'DZA', '45236699', 'Africa', '265691', '6874', '5873',
'152', '2.587215976']
['Andorra', 'Andorra', 'AND', '77481', 'Europe', '40024', '153', '516565', '1975',
'0.382270638']
['Angola', 'Angola', 'AGO', '34654212', 'Africa', '99194', '1900', '2862', '55',
'1.915438434']
['Anguilla', 'Anguilla', 'AIA', '15237', 'Latin America and the Caribbean', '2700',
'9', '177200', '591', '0.333333333']
['Antigua and Barbuda', 'Antigua and Barbuda', 'ATG', '99348', 'Latin America and
the Caribbean', '7493', '135', '75422', '1359', '1.801681569']
['Argentina', 'Argentina', 'ARG', '45921761', 'Latin America and the Caribbean',
'9041124', '128065', '196881', '2789', '1.416472111']
['Armenia', 'Armenia', 'ARM', '2972939', 'Asia', '422574', '8617', '142140',
'2898', '2.039169471']
['Aruba', 'Aruba', 'ABW', '107560', 'Latin America and the Caribbean', '34051',
'212', '316577', '1971', '0.622595518']
['Australia', 'Australia', 'AUS', '26017767', 'Oceania', '4680816', '6384',
```

'179908', '245', '0.136386476']  
[ 'Austria', 'Austria', 'AUT', '9096360', 'Europe', '3887355', '15985', '427353',  
'1757', '0.411205048']  
[ 'Azerbaijan', 'Azerbaijan', 'AZE', '10299156', 'Asia', '792061', '9697', '76905',  
'942', '1.224274393']  
[ 'Bahamas', 'Bahamas', 'BHM', '399822', 'Latin America and the Caribbean', '33295',  
'788', '83275', '1971', '2.36672173']  
[ 'Bahrain', 'Bahrain', 'BHR', '1804995', 'Asia', '556241', '1471', '308168', '815',  
'0.264453717']  
[ 'Bangladesh', 'Bangladesh', 'BGD', '167561502', 'Asia', '1951770', '29122',  
'11648', '174', '1.492081546']  
[ 'Barbados', 'Barbados', 'BRB', '287991', 'Latin America and the Caribbean',  
'59938', '375', '208125', '1302', '0.625646501']  
[ 'Belarus', 'Belarus', 'BLR', '9443882', 'Europe', '965322', '6844', '102217',  
'725', '0.708986224']  
[ 'Belgium', 'Belgium', 'BEL', '11677924', 'Europe', '3851048', '30826', '329772',  
'2640', '0.800457434']  
[ 'Belize', 'Belize', 'BLZ', '410260', 'Latin America and the Caribbean', '57289',  
'656', '139641', '1599', '1.14507148']  
[ 'Benin', 'Benin', 'BEN', '12678649', 'Africa', '26952', '163', '2126', '13',  
'0.604778866']  
[ 'Bermuda', 'Bermuda', 'BMU', '61875', 'Northern America', '12564', '128',  
'203055', '2069', '1.018783827']  
[ 'Bhutan', 'Bhutan', 'BTN', '786480', 'Asia', '31437', '12', '39972', '15',  
'0.038171581']  
[ 'Bolivia', 'Bolivia (Plurinational State of)', 'BOL', '11951714', 'Latin America  
and the Caribbean', '902448', '21896', '75508', '1832', '2.426289382']  
[ 'Bosnia and Herzegovina', 'Bosnia and Herzegovina', 'BIH', '3245097', 'Europe',  
'375693', '15719', '115773', '4844', '4.184001299']  
[ 'Botswana', 'Botswana', 'BWA', '2434708', 'Africa', '305526', '2686', '125488',  
'1103', '0.879139582']  
[ 'Brazil', 'Brazil', 'BRA', '215204501', 'Latin America and the Caribbean',  
'29999816', '660269', '139401', '3068', '2.200910166']  
[ 'British Virgin Islands', 'British Virgin Islands', 'VGB', '30583', 'Latin America  
and the Caribbean', '6155', '62', '201256', '2027', '1.007311129']  
[ 'Brunei ', 'Brunei Darussalam', 'BRN', '444812', 'Asia', '135974', '213',  
'305689', '479', '0.156647594']  
[ 'Bulgaria', 'Bulgaria', 'BGR', '6856886', 'Europe', '1140679', '36568', '166355',  
'5333', '3.205809873']  
[ 'Burkina Faso', 'Burkina Faso', 'BFA', '21905848', 'Africa', '20853', '382',  
'952', '17', '1.831870714']  
[ 'Burundi', 'Burundi', 'BDI', '12510155', 'Africa', '38519', '38', '3079', '3',  
'0.098652613']  
[ 'Cabo Verde', 'Cabo Verde', 'CPV', '566557', 'Africa', '55960', '401', '98772',  
'708', '0.716583274']  
[ 'Cambodia', 'Cambodia', 'KHM', '17123941', 'Asia', '135747', '3054', '7927',  
'178', '2.249773476']  
[ 'Cameroon', 'Cameroon', 'CMR', '27701805', 'Africa', '119544', '1927', '4315',  
'70', '1.611958777']  
[ 'Canada', 'Canada', 'CAN', '38321435', 'Northern America', '3499226', '37690',  
'91312', '984', '1.077095335']  
[ 'CAR', 'Central African Republic', 'CAF', '4976719', 'Africa', '14649', '113',  
'2944', '23', '0.771383712']  
[ 'Caribbean Netherlands', 'Bonaire, Sint Eustatius and Saba', 'BES', '26650',  
'Latin America and the Caribbean', '8574', '33', '321726', '1238', '0.384884535']  
[ 'Cayman Islands', 'Cayman Islands', 'CYM', '67073', 'Latin America and the  
Caribbean', '20606', '24', '307218', '358', '0.116470931']  
[ 'Chad', 'Chad', 'TCD', '17250246', 'Africa', '7308', '191', '424', '11',  
'2.613574165']  
[ 'Channel Islands', 'Guernsey', 'GGY', '176668', 'Europe', '69036', '156',  
'390767', '883', '0.22596906']  
[ 'Chile', 'Chile', 'CHL', '19403451', 'Latin America and the Caribbean', '3486653',  
'56750', '179692', '2925', '1.627635443']  
[ 'China', 'China', 'CHN', '1439323776', 'Asia', '154738', '4638', '108', '3',  
'2.99732451']  
[ 'Colombia', 'Colombia', 'COL', '51832231', 'Latin America and the Caribbean',  
'6085926', '139660', '117416', '2694', '2.294802796']  
[ 'Comoros', 'Comoros', 'COM', '902011', 'Africa', '8093', '160', '8972', '177',  
'1.977017175']  
[ 'Congo', 'Congo', 'COG', '5755689', 'Africa', '24071', '385', '4182', '67',  
'1.599435005']  
[ 'Cook Islands', 'Cook Islands', 'COK', '17592', 'Oceania', '2118', '0', '120396',  
'0', '0']  
[ 'Costa Rica', 'Costa Rica', 'CRI', '5175547', 'Latin America and the Caribbean',  
'839368', '8308', '162180', '1605', '0.98979232']  
[ 'Croatia', 'Croatia', 'HRV', '4060951', 'Europe', '1102730', '15601', '271545',  
'3842', '1.414761546']  
[ 'Cuba', 'Cuba', 'CUB', '11314513', 'Latin America and the Caribbean', '1092547',  
'8514', '96562', '752', '0.779279976']  
[ 'Curaçao', 'Curaçao', 'CUW', '165268', 'Latin America and the Caribbean', '40671',  
'267', '246091', '1616', '0.656487423']  
[ 'Cyprus', 'Cyprus', 'CYP', '1222745', 'Asia', '439964', '947', '359817', '774',  
'0.215244884']  
[ 'Czechia', 'Czech Republic', 'CZE', '10743762', 'Europe', '3830631', '39720',  
'356545', '3697', '1.036904886']  
[ 'Denmark', 'Denmark', 'DNK', '5827911', 'Europe', '2919428', '5762', '500939',  
'989', '0.19736743']

[**'Djibouti'**, 'Djibouti', 'DJI', '1013146', 'Africa', '15590', '189', '15388', '187', '1.212315587']  
[**'Dominica'**, 'Dominica', 'DMA', '72299', 'Latin America and the Caribbean', '11891', '63', '164470', '871', '0.529812463']  
[**'Dominican Republic'**, 'Dominican Republic', 'DOM', '11038333', 'Latin America and the Caribbean', '578130', '4375', '52375', '396', '0.756750212']  
[**'Democratic Republic of the Congo'**, 'Democratic Republic of the Congo', 'COD', '94323344', 'Africa', '86748', '1337', '920', '14', '1.541245908']  
[**'Ecuador'**, 'Ecuador', 'ECU', '18111933', 'Latin America and the Caribbean', '859890', '35421', '47476', '1956', '4.119247811']  
[**'Egypt'**, 'Egypt', 'EGY', '105711844', 'Africa', '505264', '24417', '4780', '231', '4.832523196']  
[**'El Salvador'**, 'El Salvador', 'SLV', '6543499', 'Latin America and the Caribbean', '161570', '4120', '24692', '630', '2.549978338']  
[**'Equatorial Guinea'**, 'Equatorial Guinea', 'GNQ', '1483588', 'Africa', '15903', '183', '10719', '123', '1.150726278']  
[**'Eritrea'**, 'Eritrea', 'ERI', '3632329', 'Africa', '9728', '103', '2678', '28', '1.058799342']  
[**'Estonia'**, 'Estonia', 'EST', '1328097', 'Europe', '558706', '2468', '420682', '1858', '0.441735009']  
[**'Eswatini'**, 'Eswatini', 'SWZ', '1181191', 'Africa', '69851', '1394', '59136', '1180', '1.995676511']  
[**'Ethiopia'**, 'Ethiopia', 'ETH', '119945147', 'Africa', '469819', '7504', '3917', '63', '1.597210841']  
[**'Faeroe Islands'**, 'Faeroe Islands', 'FRO', '49188', 'Europe', '34237', '28', '696044', '569', '0.081782866']  
[**'Falkland Islands'**, 'Falkland Islands (Malvinas)', 'FLK', '3657', 'Latin America and the Caribbean', '123', '0', '33634', '0', '0']  
[**'Fiji'**, 'Fiji', 'FJI', '907817', 'Oceania', '64422', '834', '70964', '919', '1.294588805']  
[**'Finland'**, 'Finland', 'FIN', '5555788', 'Europe', '889626', '3178', '160126', '572', '0.357228768']  
[**'France'**, 'France', 'FRA', '65526369', 'Europe', '25997852', '142506', '396754', '2175', '0.548145285']  
[**'French Guiana'**, 'French Guiana', 'GUF', '312224', 'Latin America and the Caribbean', '79075', '394', '253264', '1262', '0.498261144']  
[**'French Polynesia'**, 'French Polynesia', 'PYF', '283751', 'Oceania', '72318', '646', '254864', '2277', '0.893276916']  
[**'Gabon'**, 'Gabon', 'GAB', '2317612', 'Africa', '47586', '303', '20532', '131', '0.636741899']  
[**'Gambia'**, 'Gambia', 'GMB', '2535418', 'Africa', '11988', '365', '4728', '144', '3.044711378']  
[**'Georgia'**, 'Georgia', 'GEO', '3975762', 'Asia', '1649222', '16756', '414819', '4215', '1.015994208']  
[**'Germany'**, 'Germany', 'DEU', '84252947', 'Europe', '21646375', '130563', '256921', '1550', '0.603163347']  
[**'Ghana'**, 'Ghana', 'GHA', '32207812', 'Africa', '160971', '1445', '4998', '45', '0.897677221']  
[**'Gibraltar'**, 'Gibraltar', 'GIB', '33673', 'Europe', '16979', '101', '504232', '2999', '0.594852465']  
[**'Greece'**, 'Greece', 'GRC', '10333930', 'Europe', '3077711', '27684', '297826', '2679', '0.899499661']  
[**'Greenland'**, 'Greenland', 'GRL', '56942', 'Northern America', '11971', '21', '210231', '369', '0.175423941']  
[**'Grenada'**, 'Grenada', 'GRD', '113436', 'Latin America and the Caribbean', '14024', '218', '123629', '1922', '1.554478038']  
[**'Guadeloupe'**, 'Guadeloupe', 'GLP', '400244', 'Latin America and the Caribbean', '130705', '843', '326563', '2106', '0.64496385']  
[**'Guatemala'**, 'Guatemala', 'GTM', '18495493', 'Latin America and the Caribbean', '830745', '17325', '44916', '937', '2.085477493']  
[**'Guinea'**, 'Guinea', 'GIN', '13755881', 'Africa', '36459', '440', '2650', '32', '1.206835075']  
[**'Guinea-Bissau'**, 'Guinea-Bissau', 'GNB', '2049374', 'Africa', '8151', '170', '3977', '83', '2.085633665']  
[**'Guyana'**, 'Guyana', 'GUY', '793196', 'Latin America and the Caribbean', '63272', '1226', '79768', '1546', '1.93766595']  
[**'Haiti'**, 'Haiti', 'HTI', '11645833', 'Latin America and the Caribbean', '30549', '833', '2623', '72', '2.726766834']  
[**'Honduras'**, 'Honduras', 'HND', '10180299', 'Latin America and the Caribbean', '421062', '10880', '41360', '1069', '2.583942507']  
[**'Hong Kong'**, 'China', 'Hong Kong Special Administrative Region', 'HKG', '7603455', 'Asia', '1171422', '8172', '154064', '1075', '0.69761367']  
[**'Hungary'**, 'Hungary', 'HUN', '9617409', 'Europe', '1854198', '45510', '192796', '4732', '2.454430433']  
[**'Iceland'**, 'Iceland', 'ISL', '345120', 'Europe', '181830', '101', '526860', '293', '0.055546389']  
[**'India'**, 'India', 'IND', '1403754381', 'Asia', '43029044', '521388', '30653', '371', '1.211711792']  
[**'Indonesia'**, 'Indonesia', 'IDN', '278586508', 'Asia', '6019981', '155288', '21609', '557', '2.579543025']  
[**'Iran'**, 'United Kingdom', 'IRN', '85874667', 'Asia', '7167646', '140315', '83466', '1634', '1.95761621']  
[**'Iraq'**, 'Iraq', 'IRQ', '41801625', 'Asia', '2320260', '25173', '55506', '602', '1.084921517']  
[**'Ireland'**, 'Ireland', 'IRL', '5034333', 'Europe', '1471210', '6786', '292235', '1348', '0.461252982']  
[**'Isle of Man'**, 'Isle of Man', 'IMN', '85821', 'Europe', '28416', '84', '331108',

'979', '0.295608108']  
[['Israel', 'Israeli', 'ISR', '9326000', 'Asia', '3943153', '10530', '422813',  
'1129', '0.267045179']]  
[['Italy', 'Italy', 'ITA', '60306185', 'Europe', '14846514', '159784', '246186',  
'2650', '1.076239176']]  
[['Ivory Coast', "CÃ©te d'Ivoire", 'CIV', '27520953', 'Africa', '81761', '796',  
'2971', '29', '0.973569306']]  
[['Jamaica', 'Jamaica', 'JAM', '2983794', 'Latin America and the Caribbean',  
'128811', '2893', '43170', '970', '2.245926202']]  
[['Japan', 'Japan', 'JPN', '125798669', 'Asia', '6653841', '28248', '52893', '225',  
'0.424536745']]  
[['Jordan', 'Jordan', 'JOR', '10380442', 'Asia', '1689314', '14003', '162740',  
'1349', '0.828916353']]  
[['Kazakhstan', 'Kazakhstan', 'KAZ', '19169833', 'Asia', '1305188', '13660',  
'68086', '713', '1.046592522']]  
[['Kenya', 'Kenya', 'KEN', '55843563', 'Africa', '323454', '5648', '5792', '101',  
'1.746152467']]  
[['Kiribati', 'Kiribati', 'KIR', '122656', 'Oceania', '3067', '13', '25005', '106',  
'0.423866971']]  
[['Kuwait', 'Kuwait', 'KWT', '4381108', 'Asia', '629525', '2554', '143691', '583',  
'0.405702712']]  
[['Kyrgyzstan', 'Kyrgyzstan', 'KGZ', '6712569', 'Asia', '200968', '2991', '29939',  
'446', '1.488296644']]  
[['Laos', "Lao People's Democratic Republic", 'LAO', '7460338', 'Asia', '183560',  
'679', '24605', '91', '0.369906298']]  
[['Latvia', 'Latvia', 'LVA', '1849698', 'Europe', '802534', '5643', '433873',  
'3051', '0.703147779']]  
[['Lebanon', 'Lebanon', 'LBN', '6771939', 'Asia', '1092995', '10315', '161401',  
'1523', '0.943737163']]  
[['Lesotho', 'Lesotho', 'LSO', '2171978', 'Africa', '32910', '697', '15152', '321',  
'2.117897296']]  
[['Liberia', 'Liberia', 'LBR', '5265647', 'Africa', '7400', '294', '1405', '56',  
'3.972972973']]  
[['Libya', 'Libya', 'LBY', '7034832', 'Africa', '501738', '6419', '71322', '912',  
'1.279352969']]  
[['Liechtenstein', 'Liechtenstein', 'LIE', '38320', 'Europe', '16429', '84',  
'428732', '2192', '0.51129101']]  
[['Lithuania', 'Lithuania', 'LTU', '2655811', 'Europe', '1030966', '8907', '388193',  
'3354', '0.863947017']]  
[['Luxembourg', 'Luxembourg', 'LUX', '643801', 'Europe', '216979', '1037', '337028',  
'1611', '0.477926435']]  
[['Macao', 'China, Macao Special Administrative Region', 'MAC', '664828', 'Asia',  
'82', '0', '123', '0', '0']]  
[['Madagascar', 'Madagascar', 'MDG', '28936285', 'Africa', '64050', '1388', '2213',  
'48', '2.167056987']]  
[['Malawi', 'Malawi', 'MWI', '19994654', 'Africa', '85664', '2626', '4284', '131',  
'3.065465073']]  
[['Malaysia', 'Malaysia', 'MYS', '33091831', 'Asia', '4246467', '35099', '128324',  
'1061', '0.826545926']]  
[['Maldives', 'Maldives', 'MDV', '557204', 'Asia', '176993', '298', '317645', '535',  
'0.168368241']]  
[['Mali', 'Mali', 'MLI', '21271006', 'Africa', '30495', '728', '1434', '34',  
'2.387276603']]  
[['Malta', 'Malta', 'MLT', '443602', 'Europe', '81596', '641', '183940', '1445',  
'0.785577724']]  
[['Marshall Islands', 'Marshall Islands', 'MHL', '59889', 'Oceania', '7', '0',  
'117', '0', '0']]  
[['Martinique', 'Martinique', 'MTQ', '374756', 'Latin America and the Caribbean',  
'141415', '909', '377352', '2426', '0.642788954']]  
[['Mauritania', 'Mauritania', 'MRT', '4863443', 'Africa', '58670', '982', '12063',  
'202', '1.673768536']]  
[['Mauritius', 'Mauritius', 'MUS', '1275463', 'Africa', '36628', '968', '28717',  
'759', '2.642786939']]  
[['Mayotte', 'Mayotte', 'MYTÂxa0', '284330', 'Africa', '36891', '187', '129747',  
'658', '0.506898702']]  
[['Mexico', 'Mexico', 'MEX', '1313039551', 'Latin America and the Caribbean',  
'5665376', '323212', '43147', '2462', '5.705040583']]  
[['Micronesia', 'Micronesia (Federated States of)', 'FSM', '117134', 'Oceania', '1',  
'0', '9', '0', '0']]  
[['Moldova', 'Republic of Moldova', 'MDA', '4017550', 'Europe', '514199', '11446',  
'127988', '2849', '2.225986437']]  
[['Monaco', 'Monaco', 'MCO', '39729', 'Europe', '10842', '54', '272899', '1359',  
'0.498063088']]  
[['Mongolia', 'Mongolia', 'MNG', '3370682', 'Asia', '468610', '2177', '139025',  
'646', '0.464565417']]  
[['Montenegro', '', 'MNE', '628205', 'Europe', '233326', '2705', '371417', '4306',  
'1.15932215']]  
[['Montserrat', 'Montserrat', 'MSR', '4997', 'Latin America and the Caribbean',  
'175', '2', '35021', '400', '1.142857143']]  
[['Morocco', 'Morocco', 'MAR', '37676342', 'Africa', '1163526', '16060', '30882',  
'426', '1.380287162']]  
[['Mozambique', 'Mozambique', 'MOZ', '32787052', 'Africa', '225266', '2200', '6871',  
'67', '0.976623192']]  
[['Myanmar', 'Myanmar', 'MMR', '55048340', 'Asia', '611875', '19433', '11115',  
'353', '3.175975485']]  
[['Namibia', 'Namibia', 'NAM', '2621429', 'Africa', '157646', '4019', '60137',  
'1533', '2.549382794']]

[ 'Nepal', 'Nepal', 'NPL', '30053867', 'Asia', '978475', '11951', '32557', '398', '1.221390429' ]  
[ 'Netherlands', 'Netherlands', 'NLD', '17201245', 'Europe', '7908701', '22016', '459775', '1280', '0.278376942' ]  
[ 'New Caledonia', 'New Caledonia', 'NCL', '290302', 'Oceania', '60294', '311', '207694', '1071', '0.515805884' ]  
[ 'New Zealand', 'New Zealand', 'NZL', '5002100', 'Oceania', '693219', '350', '138586', '70', '0.050489095' ]  
[ 'Nicaragua', 'Nicaragua', 'NIC', '6762511', 'Latin America and the Caribbean', '18434', '224', '2726', '33', '1.215145926' ]  
[ 'Niger', 'Niger', 'NER', '25738714', 'Africa', '8811', '308', '342', '12', '3.495630462' ]  
[ 'Nigeria', 'Nigeria', 'NGA', '215077352', 'Africa', '255468', '3142', '1188', '15', '1.229899635' ]  
[ 'Niue', 'Niue', 'NIU', '1645', 'Oceania', '7', '0', '4255', '0', '0' ]  
[ 'North Macedonia', 'The former Yugoslav Republic of Macedonia', 'MKD', '2083224', 'Europe', '306670', '9228', '147209', '4430', '3.009097727' ]  
[ 'Norway', 'Norway', 'NOR', '5495449', 'Europe', '1408708', '2518', '256341', '458', '0.178745347' ]  
[ 'Oman', 'Oman', 'OMN', '5333815', 'Asia', '388468', '4251', '72831', '797', '1.094298629' ]  
[ 'Pakistan', 'Pakistan', 'PAK', '228397520', 'Asia', '1525466', '30361', '6679', '133', '1.990277069' ]  
[ 'Palau', 'Palau', 'PLW', '18245', 'Oceania', '4042', '6', '221540', '329', '0.148441366' ]  
[ 'Palestine', 'State of Palestine', 'WBG', '5308883', 'Asia', '581236', '5351', '109484', '1008', '0.920624325' ]  
[ 'Panama', 'Panama', 'PAN', '4433639', 'Latin America and the Caribbean', '765213', '8170', '172593', '1843', '1.067676582' ]  
[ 'Papua New Guinea', 'Papua New Guinea', 'PNG', '9243590', 'Oceania', '42203', '640', '4566', '69', '1.516479871' ]  
[ 'Paraguay', 'Paraguay', 'PRY', '7285892', 'Latin America and the Caribbean', '648353', '18731', '88987', '2571', '2.889012621' ]  
[ 'Peru', 'Peru', 'PER', '33775745', 'Latin America and the Caribbean', '3548559', '212328', '105062', '6286', '5.983499218' ]  
[ 'Philippines', 'Philippines', 'PHL', '112133868', 'Asia', '3679485', '59343', '32813', '529', '1.612807227' ]  
[ 'Poland', 'Poland', 'POL', '37774045', 'Europe', '5969621', '115345', '158035', '3054', '1.932199716' ]  
[ 'Portugal', 'Portugal', 'PRT', '10144662', 'Europe', '3604114', '21693', '355272', '2138', '0.601895501' ]  
[ 'Qatar', 'Qatar', 'QAT', '2807805', 'Asia', '361819', '677', '128862', '241', '0.18711013' ]  
[ 'Réunion', 'RÃ©union', 'REU', '906497', 'Africa', '336945', '709', '371700', '782', '0.210420098' ]  
[ 'Romania', 'Romania', 'ROU', '19013049', 'Europe', '2860094', '65090', '150428', '3423', '2.275799327' ]  
[ 'Russia', 'Russian Federation', 'RUS', '146044010', 'Europe', '17896866', '369708', '122544', '2531', '2.065769504' ]  
[ 'Rwanda', 'Rwanda', 'RWA', '13513881', 'Africa', '129728', '1458', '9600', '108', '1.123889985' ]  
[ 'S. Korea', 'Republic of Korea', 'KOR', '51346429', 'Asia', '13874216', '17235', '270208', '336', '0.124223235' ]  
[ 'Saint Helena', 'Saint Helena', 'SHN', '6109', 'Africa', '2', '0', '327', '0', '0' ]  
[ 'Saint Kitts and Nevis', 'Saint Kitts and Nevis', 'KNA', '53858', 'Latin America and the Caribbean', '5549', '43', '103030', '798', '0.774914399' ]  
[ 'Saint Lucia', 'Saint Lucia', 'LCA', '185096', 'Latin America and the Caribbean', '22964', '365', '124065', '1972', '1.589444348' ]  
[ 'Saint Martin', 'Saint Martin', 'MAF', '39820', 'Latin America and the Caribbean', '10107', '63', '253817', '1582', '0.623330365' ]  
[ 'Saint Pierre Miquelon', '\xa0Saint Pierre and Miquelon', 'SPM', '5744', 'Northern America', '1957', '1', '340703', '174', '0.05109862' ]  
[ 'Samoa', 'Samoa', 'WSM', '200722', 'Oceania', '2285', '1', '11384', '5', '0.043763676' ]  
[ 'San Marino', 'San Marino', 'SMR', '34056', 'Europe', '15181', '113', '445766', '3318', '0.744351492' ]  
[ 'Sao Tome and Principe', 'Sao Tome and Principe', 'STP', '226281', 'Africa', '5945', '73', '26273', '323', '1.227922624' ]  
[ 'Saudi Arabia', 'Saudi Arabia', 'SAU', '35762746', 'Asia', '751076', '9048', '21002', '253', '1.204671698' ]  
[ 'Senegal', 'Senegal', 'SEN', '17515750', 'Africa', '85919', '1965', '4905', '112', '2.287037791' ]  
[ 'Serbia', 'Serbia', 'SRB', '8675762', 'Europe', '1980722', '15825', '228305', '1824', '0.79895109' ]  
[ 'Seychelles', 'Seychelles', 'SYC', '99413', 'Africa', '40421', '164', '406597', '1650', '0.405729695' ]  
[ 'Sierra Leone', 'Sierra Leone', 'SLE', '8260822', 'Africa', '7674', '125', '929', '15', '1.628876727' ]  
[ 'Singapore', 'Singapore', 'SGP', '5930887', 'Asia', '1109744', '1276', '187113', '215', '0.114981473' ]  
[ 'Sint Maarten', 'Sint Maarten', 'SXM', '43728', 'Latin America and the Caribbean', '9766', '86', '223335', '1967', '0.880606185' ]  
[ 'Slovakia', 'Slovakia', 'SVK', '5464272', 'Europe', '1725487', '19417', '315776', '3553', '1.125305493' ]  
[ 'Slovenia', 'Slovenia', 'SVN', '2079438', 'Europe', '973892', '6501', '468344', '3126', '0.667527816' ]

[ 'Solomon Islands', 'Solomon Islands', 'SLB', '716351', 'Oceania', '11470', '133',  
'16012', '186', '1.159546643' ]  
[ 'Somalia', 'Somalia', 'SOM', '16668781', 'Africa', '26400', '1348', '1584', '81',  
'5.106060606' ]  
[ 'South Africa', 'South Africa', 'ZAF', '60617532', 'Africa', '3722954', '100050',  
'61417', '1651', '2.687382116' ]  
[ 'South Sudan', 'South Sudan', 'SSD', '11423439', 'Africa', '17278', '138', '1513',  
'12', '0.798703554' ]  
[ 'Spain', 'Spain', 'ESP', '46786482', 'Europe', '11551574', '102541', '246900',  
'2192', '0.887679895' ]  
[ 'Sri Lanka', 'Sri Lanka', 'LKA', '21570428', 'Asia', '661991', '16481', '30690',  
'764', '2.489610880' ]  
[ 'St. Barth', 'Saint Barthélemy', 'BLM', '9930', 'Latin America and the  
Caribbean', '4150', '6', '417925', '604', '0.144578313' ]  
[ 'St. Vincent Grenadines', 'Saint Vincent and the Grenadines', 'VCT', '111557',  
'Latin America and the Caribbean', '6746', '106', '60471', '950', '1.571301512' ]  
[ 'Sudan', 'Sudan', 'SDN', '45640385', 'Africa', '61955', '4907', '1357', '108',  
'7.920264708' ]  
[ 'Suriname', 'Suriname', 'SUR', '595833', 'Latin America and the Caribbean',  
'79232', '1325', '132977', '2224', '1.67230412' ]  
[ 'Sweden', 'Sweden', 'SWE', '10209507', 'Europe', '2487852', '18331', '243680',  
'1795', '0.736820357' ]  
[ 'Switzerland', 'Switzerland', 'CHE', '8765420', 'Europe', '3490876', '13715',  
'398255', '1565', '0.392881328' ]  
[ 'Syria', 'Syrian Arab Republic', 'SYR', '18244381', 'Asia', '55711', '3144',  
'3054', '172', '5.64340974' ]  
[ 'Taiwan', 'China, Taiwan Province of China', 'TWN', '23892241', 'Asia', '24310',  
'853', '1017', '36', '3.508844097' ]  
[ 'Tajikistan', 'Tajikistan', 'TJK', '9912437', 'Asia', '17388', '124', '1754',  
'13', '0.713135496' ]  
[ 'Tanzania', 'United Republic of Tanzania', 'TZA', '62710097', 'Africa', '33815',  
'800', '539', '13', '2.365813988' ]  
[ 'Thailand', 'Thailand', 'THA', '70106601', 'Asia', '3711595', '25418', '52942',  
'363', '0.684826874' ]  
[ 'Timor-Leste', 'Timor-Leste', 'TLS', '1362386', 'Asia', '22832', '130', '16759',  
'95', '0.569376314' ]  
[ 'Togo', 'Togo', 'TGO', '8618172', 'Africa', '36944', '272', '4287', '32',  
'0.736249459' ]  
[ 'Tonga', 'Tonga', 'TON', '107792', 'Oceania', '7127', '9', '66118', '83',  
'0.126280342' ]  
[ 'Trinidad and Tobago', 'Trinidad and Tobago', 'TTO', '1407422', 'Latin America and  
the Caribbean', '138425', '3756', '98354', '2669', '2.713382698' ]  
[ 'Tunisia', 'Tunisia', 'TUN', '12035092', 'Africa', '1035884', '28323', '86072',  
'2353', '2.734186453' ]  
[ 'Turkey', 'Turkey', 'TUR', '85927644', 'Asia', '14894731', '98157', '173340',  
'1142', '0.659004852' ]  
[ 'Turks and Caicos', 'Turks and Caicos Islands', 'TCA\x0', '39634', 'Latin  
America and the Caribbean', '5910', '36', '149114', '908', '0.609137056' ]  
[ 'UAE', 'United Arab Emirates', 'ARE', '10099567', 'Asia', '892170', '2302',  
'88337', '228', '0.258022574' ]  
[ 'Uganda', 'Uganda', 'UGA', '48267221', 'Africa', '163936', '3595', '3396', '74',  
'2.192928948' ]  
[ 'UK', 'United Kingdom of Great Britain and Northern Ireland', 'GBR', '68510300',  
'Europe', '21216874', '165570', '309689', '2417', '0.780369436' ]  
[ 'Ukraine', 'Ukraine', 'UKR', '43273831', 'Europe', '4968881', '107980', '114824',  
'2495', '2.173125096' ]  
[ 'Uruguay', 'Uruguay', 'URY', '3494806', 'Latin America and the Caribbean',  
'889513', '7166', '254524', '2050', '0.805609362' ]  
[ 'USA', 'United States of America', 'USA', '334400597', 'Northern America',  
'81839052', '1008222', '244734', '3015', '1.231957086' ]  
[ 'Uzbekistan', 'Uzbekistan', 'UZB', '34318156', 'Asia', '237853', '1637', '6931',  
'48', '0.688240216' ]  
[ 'Vanuatu', 'Vanuatu', 'VUT', '319701', 'Oceania', '4107', '2', '12846', '6',  
'0.048697346' ]  
[ 'Vatican City', 'Holy See', 'VAT', '805', 'Europe', '29', '0', '36025', '0', '0' ]  
[ 'Venezuela', 'Venezuela (Bolivarian Republic of)', 'VEN', '28294895', 'Latin  
America and the Caribbean', '520843', '5686', '18408', '201', '1.091691738' ]  
[ 'Vietnam', 'Viet Nam', 'VNM', '98871712', 'Asia', '9818328', '42600', '99304',  
'431', '0.433882429' ]  
[ 'Wallis and Futuna', 'Wallis and Futuna Islands', 'WLF', '10894', 'Oceania',  
'454', '7', '41674', '643', '1.54185022' ]  
[ 'Western Sahara', 'Western Sahara', 'ESH\x0', '623031', 'Africa', '10', '1',  
'16', '2', '10' ]  
[ 'Yemen', 'Yemen', 'YEM', '30975258', 'Asia', '11806', '2143', '381', '69',  
'18.15178723' ]  
[ 'Zambia', 'Zambia', 'ZMB', '19284482', 'Africa', '317076', '3967', '16442', '206',  
'1.251119605' ]  
[ 'Zimbabwe', 'Zimbabwe', 'ZWE', '15241601', 'Africa', '246525', '5446', '16174',  
'357', '2.209106581' ]

```
with open(local_file1, 'r') as file:
 reader = csv.reader(file)
 line_count = 0
 for row in reader:
 if line_count == 0:
 print(f'Όνόματα στηλών {" ".join(row)}')
 print("\n\n")
 line_count += 1
 else:
 print(row)
```

Ονόματα στηλών Country, Other names, ISO 3166-1 alpha-3 CODE, Population, Continent, Total Cases, Total Deaths, Tot Cases//1M pop, Tot Deaths/1M pop, Death percentage

```
['Afghanistan', 'Afghanistan', 'AFG', '40462186', 'Asia', '177827', '7671', '4395', '190', '4.313743132']
['Albania', 'Albania', 'ALB', '2872296', 'Europe', '273870', '3492', '95349', '1216', '1.275057509']
['Algeria', 'Algeria', 'DZA', '45236699', 'Africa', '265691', '6874', '5873', '152', '2.587215976']
['Andorra', 'Andorra', 'AND', '77481', 'Europe', '40024', '153', '516565', '1975', '0.382270638']
['Angola', 'Angola', 'AGO', '34654212', 'Africa', '99194', '1900', '2862', '55', '1.915438434']
['Anguilla', 'Anguilla', 'AIA', '15237', 'Latin America and the Caribbean', '2700', '9', '177200', '591', '0.333333333']
['Antigua and Barbuda', 'Antigua and Barbuda', 'ATG', '99348', 'Latin America and the Caribbean', '7493', '135', '75422', '1359', '1.801681569']
['Argentina', 'Argentina', 'ARG', '45921761', 'Latin America and the Caribbean', '90411241', '128065', '196881', '2789', '1.416472111']
['Armenia', 'Armenia', 'ARM', '2972939', 'Asia', '422574', '8617', '142140', '2898', '2.039169471']
['Aruba', 'Aruba', 'ABW', '107560', 'Latin America and the Caribbean', '34051', '212', '316577', '1971', '0.622595518']
['Australia', 'Australia', 'AUS', '26017767', 'Oceania', '4680816', '6384', '179908', '245', '0.136386476']
['Austria', 'Austria', 'AUT', '9096360', 'Europe', '3887355', '15985', '427353', '1757', '0.411205048']
['Azerbaijan', 'Azerbaijan', 'AZE', '10299156', 'Asia', '792061', '9697', '76905', '942', '1.224274393']
['Bahamas', 'Bahamas', 'BHM', '399822', 'Latin America and the Caribbean', '33295', '788', '83275', '1971', '2.36672173']
['Bahrain', 'Bahrain', 'BHR', '1804995', 'Asia', '556241', '1471', '308168', '815', '0.264453717']
['Bangladesh', 'Bangladesh', 'BGD', '167561502', 'Asia', '1951770', '29122', '11648', '174', '1.492081546']
['Barbados', 'Barbados', 'BRB', '287991', 'Latin America and the Caribbean', '59938', '375', '208125', '1302', '0.625646501']
['Belarus', 'Belarus', 'BLR', '9443882', 'Europe', '965322', '6844', '102217', '725', '0.708986224']
['Belgium', 'Belgium', 'BEL', '11677924', 'Europe', '3851048', '30826', '329772', '2640', '0.800457434']
['Belize', 'Belize', 'BLZ', '410260', 'Latin America and the Caribbean', '57289', '656', '139641', '1599', '1.14507148']
['Benin', 'Benin', 'BEN', '12678649', 'Africa', '26952', '163', '2126', '13', '0.604778866']
['Bermuda', 'Bermuda', 'BMU', '61875', 'Northern America', '12564', '128', '203055', '2069', '1.018783827']
['Bhutan', 'Bhutan', 'BTN', '786480', 'Asia', '31437', '12', '39972', '15', '0.038171581']
['Bolivia', 'Bolivia (Plurinational State of)', 'BOL', '11951714', 'Latin America and the Caribbean', '902448', '21896', '75508', '1832', '2.426289382']
['Bosnia and Herzegovina', 'Bosnia and Herzegovina', 'BIH', '3245097', 'Europe', '375693', '15719', '115773', '4844', '4.184001299']
['Botswana', 'Botswana', 'BWA', '2434708', 'Africa', '305526', '2686', '125488', '1103', '0.879139582']
['Brazil', 'Brazil', 'BRA', '215204501', 'Latin America and the Caribbean', '29999816', '660269', '139401', '3068', '2.200910166']
['British Virgin Islands', 'British Virgin Islands', 'VGB', '30583', 'Latin America and the Caribbean', '6155', '62', '201256', '2027', '1.007311129']
['Brunei', 'Brunei Darussalam', 'BRN', '444812', 'Asia', '135974', '213', '305689', '479', '0.156647594']
['Bulgaria', 'Bulgaria', 'BGR', '6856886', 'Europe', '1140679', '36568', '166355', '5333', '3.205809873']
['Burkina Faso', 'Burkina Faso', 'BFA', '21905848', 'Africa', '20853', '382', '952', '17', '1.831870714']
['Burundi', 'Burundi', 'BDI', '12510155', 'Africa', '38519', '38', '3079', '3', '0.098652613']
['Cabo Verde', 'Cabo Verde', 'CPV', '566557', 'Africa', '55960', '401', '98772', '708', '0.716583274']
['Cambodia', 'Cambodia', 'KHM', '17123941', 'Asia', '135747', '3054', '7927', '178', '2.249773476']
```

[ 'Cameroon', 'Cameroon', 'CMR', '27701805', 'Africa', '119544', '1927', '4315', '70', '1.611958777']  
[ 'Canada', 'Canada', 'CAN', '38321435', 'Northern America', '3499226', '37690', '91312', '984', '1.077095335']  
[ 'CAR', 'Central African Republic', 'CAF', '4976719', 'Africa', '14649', '113', '2944', '23', '0.771383712']  
[ 'Caribbean Netherlands', 'Bonaire, Sint Eustatius and Saba', 'BES', '26650', 'Latin America and the Caribbean', '8574', '33', '321726', '1238', '0.384884535']  
[ 'Cayman Islands', 'Cayman Islands', 'CYM', '67073', 'Latin America and the Caribbean', '20606', '24', '307218', '358', '0.116470931']  
[ 'Chad', 'Chad', 'TCD', '17250246', 'Africa', '7308', '191', '424', '11', '2.613574165']  
[ 'Channel Islands', 'Guernsey', 'GGY', '176668', 'Europe', '69036', '156', '390767', '883', '0.22596906']  
[ 'Chile', 'Chile', 'CHL', '19403451', 'Latin America and the Caribbean', '3486653', '56750', '179692', '2925', '1.627635443']  
[ 'China', 'China', 'CHN', '1439323776', 'Asia', '154738', '4638', '108', '3', '2.99732451']  
[ 'Colombia', 'Colombia', 'COL', '51832231', 'Latin America and the Caribbean', '6085926', '139660', '117416', '2694', '2.294802796']  
[ 'Comoros', 'Comoros', 'COM', '902011', 'Africa', '8093', '160', '8972', '177', '1.977017175']  
[ 'Congo', 'Congo', 'COG', '5755689', 'Africa', '24071', '385', '4182', '67', '1.599435005']  
[ 'Cook Islands', 'Cook Islands', 'COK', '17592', 'Oceania', '2118', '0', '120396', '0', '0']  
[ 'Costa Rica', 'Costa Rica', 'CRI', '5175547', 'Latin America and the Caribbean', '839368', '8308', '162180', '1605', '0.98979232']  
[ 'Croatia', 'Croatia', 'HRV', '4060951', 'Europe', '1102730', '15601', '271545', '3842', '1.414761546']  
[ 'Cuba', 'Cuba', 'CUB', '11314513', 'Latin America and the Caribbean', '1092547', '8514', '96562', '752', '0.779279976']  
[ 'Curaçao', 'Curaçao', 'CUW', '165268', 'Latin America and the Caribbean', '40671', '2671', '246091', '1616', '0.656487423']  
[ 'Cyprus', 'Cyprus', 'CYP', '1222745', 'Asia', '439964', '947', '359817', '774', '0.215244884']  
[ 'Czechia', 'Czech Republic', 'CZE', '10743762', 'Europe', '3830631', '39720', '356545', '3697', '1.036904886']  
[ 'Denmark', 'Denmark', 'DNK', '5827911', 'Europe', '2919428', '5762', '500939', '989', '0.19736743']  
[ 'Djibouti', 'Djibouti', 'DJI', '1013146', 'Africa', '15590', '189', '15388', '187', '1.212315587']  
[ 'Dominica', 'Dominica', 'DMA', '72299', 'Latin America and the Caribbean', '11891', '63', '164470', '871', '0.529812463']  
[ 'Dominican Republic', 'Dominican Republic', 'DOM', '11038333', 'Latin America and the Caribbean', '578130', '4375', '52375', '396', '0.756750212']  
[ 'Democratic Republic of the Congo', 'Democratic Republic of the Congo', 'COD', '94323344', 'Africa', '86748', '1337', '920', '14', '1.541245908']  
[ 'Ecuador', 'Ecuador', 'ECU', '18111933', 'Latin America and the Caribbean', '859890', '35421', '47476', '1956', '4.119247811']  
[ 'Egypt', 'Egypt', 'EGY', '105711844', 'Africa', '505264', '24417', '4780', '231', '4.832523196']  
[ 'El Salvador', 'El Salvador', 'SLV', '6543499', 'Latin America and the Caribbean', '161570', '4120', '24692', '630', '2.549978338']  
[ 'Equatorial Guinea', 'Equatorial Guinea', 'GNQ', '1483588', 'Africa', '15903', '183', '10719', '123', '1.150726278']  
[ 'Eritrea', 'Eritrea', 'ERI', '3632329', 'Africa', '9728', '103', '2678', '28', '1.058799342']  
[ 'Estonia', 'Estonia', 'EST', '1328097', 'Europe', '558706', '2468', '420682', '1858', '0.441735009']  
[ 'Eswatini', 'Eswatini', 'SWZ', '1181191', 'Africa', '69851', '1394', '59136', '1180', '1.995676511']  
[ 'Ethiopia', 'Ethiopia', 'ETH', '119945147', 'Africa', '469819', '7504', '3917', '63', '1.597210841']  
[ 'Faeroe Islands', 'Faeroe Islands', 'FRO', '49188', 'Europe', '34237', '28', '696044', '569', '0.081782866']  
[ 'Falkland Islands', 'Falkland Islands (Malvinas)', 'FLK', '3657', 'Latin America and the Caribbean', '123', '0', '33634', '0', '0']  
[ 'Fiji', 'Fiji', 'FJI', '907817', 'Oceania', '64422', '834', '70964', '919', '1.294588805']  
[ 'Finland', 'Finland', 'FIN', '5555788', 'Europe', '889626', '3178', '160126', '572', '0.357228768']  
[ 'France', 'France', 'FRA', '65526369', 'Europe', '25997852', '142506', '396754', '2175', '0.548145285']  
[ 'French Guiana', 'French Guiana', 'GUF', '312224', 'Latin America and the Caribbean', '79075', '394', '253264', '1262', '0.498261144']  
[ 'French Polynesia', 'French Polynesia', 'PYF', '283751', 'Oceania', '72318', '646', '254864', '2277', '0.893276916']  
[ 'Gabon', 'Gabon', 'GAB', '2317612', 'Africa', '47586', '303', '20532', '131', '0.636741899']  
[ 'Gambia', 'Gambia', 'GMB', '2535418', 'Africa', '11988', '365', '4728', '144', '3.044711378']  
[ 'Georgia', 'Georgia', 'GEO', '3975762', 'Asia', '1649222', '16756', '414819', '4215', '1.015994208']  
[ 'Germany', 'Germany', 'DEU', '84252947', 'Europe', '21646375', '130563', '256921', '1550', '0.603163347']  
[ 'Ghana', 'Ghana', 'GHA', '32207812', 'Africa', '160971', '1445', '4998', '45',

'0.897677221']  
[['Gibraltar', 'Gibraltar', 'GIB', '33673', 'Europe', '16979', '101', '504232', '2999', '0.594852465']]  
[['Greece', 'Greece', 'GRC', '10333930', 'Europe', '3077711', '27684', '297826', '2679', '0.899499661']]  
[['Greenland', 'Greenland', 'GRL', '56942', 'Northern America', '11971', '21', '210231', '369', '0.175423941']]  
[['Grenada', 'Grenada', 'GRD', '113436', 'Latin America and the Caribbean', '14024', '218', '123629', '1922', '1.554478038']]  
[['Guadeloupe', 'Guadeloupe', 'GLP', '400244', 'Latin America and the Caribbean', '130705', '843', '326563', '2106', '0.64496385']]  
[['Guatemala', 'Guatemala', 'GTM', '18495493', 'Latin America and the Caribbean', '830745', '17325', '44916', '937', '2.085477493']]  
[['Guinea', 'Guinea', 'GIN', '13755881', 'Africa', '36459', '440', '2650', '32', '1.206835075']]  
[['Guinea-Bissau', 'Guinea-Bissau', 'GNB', '2049374', 'Africa', '8151', '170', '3977', '83', '2.085633665']]  
[['Guyana', 'Guyana', 'GUY', '793196', 'Latin America and the Caribbean', '63272', '1226', '79768', '1546', '1.93766595']]  
[['Haiti', 'Haiti', 'HTI', '11645833', 'Latin America and the Caribbean', '30549', '833', '2623', '72', '2.726766834']]  
[['Honduras', 'Honduras', 'HND', '10180299', 'Latin America and the Caribbean', '421062', '10880', '41360', '1069', '2.583942507']]  
[['Hong Kong', 'China, Hong Kong Special Administrative Region', 'HKG', '7603455', 'Asia', '1171422', '8172', '154064', '1075', '0.69761367']]  
[['Hungary', 'Hungary', 'HUN', '9617409', 'Europe', '1854198', '45510', '192796', '4732', '2.454430433']]  
[['Iceland', 'Iceland', 'ISL', '345120', 'Europe', '181830', '101', '526860', '293', '0.055546389']]  
[['India', 'India', 'IND', '1403754381', 'Asia', '43029044', '521388', '30653', '371', '1.211711792']]  
[['Indonesia', 'Indonesia', 'IDN', '278586508', 'Asia', '6019981', '155288', '21609', '557', '2.579543025']]  
[['Iran', 'United Kingdom', 'IRN', '85874667', 'Asia', '7167646', '140315', '83466', '1634', '1.95761621']]  
[['Iraq', 'Iraq', 'IRQ', '41801625', 'Asia', '2320260', '25173', '55506', '602', '1.084921517']]  
[['Ireland', 'Ireland', 'IRL', '5034333', 'Europe', '1471210', '6786', '292235', '1348', '0.461252982']]  
[['Isle of Man', 'Isle of Man', 'IMN', '85821', 'Europe', '28416', '84', '331108', '979', '0.295608108']]  
[['Israel', 'Israel', 'ISR', '9326000', 'Asia', '3943153', '10530', '422813', '1129', '0.267045179']]  
[['Italy', 'Italy', 'ITA', '60306185', 'Europe', '14846514', '159784', '246186', '2650', '1.076239176']]  
[['Ivory Coast', 'Côte d'Ivoire', 'CIV', '27520953', 'Africa', '81761', '796', '2971', '29', '0.973569306']]  
[['Jamaica', 'Jamaica', 'JAM', '2983794', 'Latin America and the Caribbean', '128811', '2893', '43170', '970', '2.245926202']]  
[['Japan', 'Japan', 'JPN', '125798669', 'Asia', '6653841', '28248', '52893', '225', '0.424536745']]  
[['Jordan', 'Jordan', 'JOR', '10380442', 'Asia', '1689314', '14003', '162740', '1349', '0.828916353']]  
[['Kazakhstan', 'Kazakhstan', 'KAZ', '19169833', 'Asia', '1305188', '13660', '68086', '713', '1.046592522']]  
[['Kenya', 'Kenya', 'KEN', '55843563', 'Africa', '323454', '5648', '5792', '101', '1.746152467']]  
[['Kiribati', 'Kiribati', 'KIR', '122656', 'Oceania', '3067', '13', '25005', '106', '0.423866971']]  
[['Kuwait', 'Kuwait', 'KWT', '4381108', 'Asia', '629525', '2554', '143691', '583', '0.405702712']]  
[['Kyrgyzstan', 'Kyrgyzstan', 'KGZ', '6712569', 'Asia', '200968', '2991', '29939', '446', '1.488296644']]  
[['Laos', 'Lao People's Democratic Republic', 'LAO', '7460338', 'Asia', '183560', '679', '24605', '91', '0.369906298']]  
[['Latvia', 'Latvia', 'LVA', '1849698', 'Europe', '802534', '5643', '433873', '3051', '0.703147779']]  
[['Lebanon', 'Lebanon', 'LBN', '6771939', 'Asia', '1092995', '10315', '161401', '1523', '0.943737163']]  
[['Lesotho', 'Lesotho', 'LSO', '2171978', 'Africa', '32910', '697', '15152', '321', '2.117897296']]  
[['Liberia', 'Liberia', 'LBR', '5265647', 'Africa', '7400', '294', '1405', '56', '3.972972973']]  
[['Libya', 'Libya', 'LBY', '7034832', 'Africa', '501738', '6419', '71322', '912', '1.279352969']]  
[['Liechtenstein', 'Liechtenstein', 'LIE', '38320', 'Europe', '16429', '84', '428732', '2192', '0.51129101']]  
[['Lithuania', 'Lithuania', 'LTU', '2655811', 'Europe', '1030966', '8907', '388193', '3354', '0.863947017']]  
[['Luxembourg', 'Luxembourg', 'LUX', '643801', 'Europe', '216979', '1037', '337028', '1611', '0.477926435']]  
[['Macao', 'China, Macao Special Administrative Region', 'MAC', '664828', 'Asia', '82', '0', '123', '0', '0']]  
[['Madagascar', 'Madagascar', 'MDG', '28936285', 'Africa', '64050', '1388', '2213', '48', '2.167056987']]  
[['Malawi', 'Malawi', 'MWI', '19994654', 'Africa', '85664', '2626', '4284', '131', '3.065465073']]

[ 'Malaysia', 'Malaysia', 'MYS', '33091831', 'Asia', '4246467', '35099', '128324', '1061', '0.826545926' ]  
[ 'Maldives', 'Maldives', 'MDV', '557204', 'Asia', '176993', '298', '317645', '535', '0.168368241' ]  
[ 'Mali', 'Mali', 'MLI', '21271006', 'Africa', '30495', '728', '1434', '34', '2.387276603' ]  
[ 'Malta', 'Malta', 'MLT', '443602', 'Europe', '81596', '641', '183940', '1445', '0.785577724' ]  
[ 'Marshall Islands', 'Marshall Islands', 'MHL', '59889', 'Oceania', '7', '0', '117', '0', '0' ]  
[ 'Martinique', 'Martinique', 'MTQ', '374756', 'Latin America and the Caribbean', '141415', '909', '377352', '2426', '0.642788954' ]  
[ 'Mauritania', 'Mauritania', 'MRT', '4863443', 'Africa', '58670', '982', '12063', '202', '1.673768536' ]  
[ 'Mauritius', 'Mauritius', 'MUS', '1275463', 'Africa', '36628', '968', '28717', '759', '2.642786939' ]  
[ 'Mayotte', 'Mayotte', 'MYTÃ\x00', '284330', 'Africa', '36891', '187', '129747', '658', '0.506898702' ]  
[ 'Mexico', 'Mexico', 'MEX', '131303955', 'Latin America and the Caribbean', '5665376', '323212', '43147', '2462', '5.705040583' ]  
[ 'Micronesia', 'Micronesia (Federated States of)', 'FSM', '117134', 'Oceania', '1', '0', '9', '0', '0' ]  
[ 'Moldova', 'Republic of Moldova', 'MDA', '4017550', 'Europe', '514199', '11446', '127988', '2849', '2.225986437' ]  
[ 'Monaco', 'Monaco', 'MCO', '39729', 'Europe', '10842', '54', '272899', '1359', '0.498063088' ]  
[ 'Mongolia', 'Mongolia', 'MNG', '3370682', 'Asia', '468610', '2177', '139025', '646', '0.464565417' ]  
[ 'Montenegro', '', 'MNE', '628205', 'Europe', '233326', '2705', '371417', '4306', '1.15932215' ]  
[ 'Montserrat', 'Montserrat', 'MSR', '4997', 'Latin America and the Caribbean', '175', '2', '35021', '400', '1.142857143' ]  
[ 'Morocco', 'Morocco', 'MAR', '37676342', 'Africa', '1163526', '16060', '30882', '426', '1.380287162' ]  
[ 'Mozambique', 'Mozambique', 'MOZ', '32787052', 'Africa', '225266', '2200', '6871', '67', '0.976623192' ]  
[ 'Myanmar', 'Myanmar', 'MMR', '55048340', 'Asia', '611875', '19433', '11115', '353', '3.175975485' ]  
[ 'Namibia', 'Namibia', 'NAM', '2621429', 'Africa', '157646', '4019', '60137', '1533', '2.549382794' ]  
[ 'Nepal', 'Nepal', 'NPL', '30053867', 'Asia', '978475', '11951', '32557', '398', '1.221390429' ]  
[ 'Netherlands', 'Netherlands', 'NLD', '17201245', 'Europe', '7908701', '22016', '459775', '1280', '0.278376942' ]  
[ 'New Caledonia', 'New Caledonia', 'NCL', '290302', 'Oceania', '60294', '311', '207694', '1071', '0.515805884' ]  
[ 'New Zealand', 'New Zealand', 'NZL', '5002100', 'Oceania', '693219', '350', '138586', '70', '0.0504890951' ]  
[ 'Nicaragua', 'Nicaragua', 'NIC', '6762511', 'Latin America and the Caribbean', '18434', '224', '2726', '33', '1.215145926' ]  
[ 'Niger', 'Niger', 'NER', '25738714', 'Africa', '8811', '308', '342', '12', '3.495630462' ]  
[ 'Nigeria', 'Nigeria', 'NGA', '215077352', 'Africa', '255468', '3142', '1188', '15', '1.229899635' ]  
[ 'Niue', 'Niue', 'NIU', '1645', 'Oceania', '7', '0', '4255', '0', '0' ]  
[ 'North Macedonia', 'The former Yugoslav Republic of Macedonia', 'MKD', '2083224', 'Europe', '306670', '9228', '147209', '4430', '3.009097727' ]  
[ 'Norway', 'Norway', 'NOR', '5495449', 'Europe', '1408708', '2518', '256341', '458', '0.178745347' ]  
[ 'Oman', 'Oman', 'OMN', '5333815', 'Asia', '388468', '4251', '72831', '797', '1.094298629' ]  
[ 'Pakistan', 'Pakistan', 'PAK', '228397520', 'Asia', '1525466', '30361', '6679', '133', '1.990277069' ]  
[ 'Palau', 'Palau', 'PLW', '18245', 'Oceania', '4042', '6', '221540', '329', '0.148441366' ]  
[ 'Palestine', 'State of Palestine', 'WBG', '5308883', 'Asia', '581236', '5351', '109484', '1008', '0.920624325' ]  
[ 'Panama', 'Panama', 'PAN', '4433639', 'Latin America and the Caribbean', '765213', '8170', '172593', '1843', '1.0676765821' ]  
[ 'Papua New Guinea', 'Papua New Guinea', 'PNG', '9243590', 'Oceania', '42203', '640', '4566', '69', '1.516479871' ]  
[ 'Paraguay', 'Paraguay', 'PRY', '7285892', 'Latin America and the Caribbean', '648353', '18731', '88987', '2571', '2.889012621' ]  
[ 'Peru', 'Peru', 'PER', '33775745', 'Latin America and the Caribbean', '3548559', '212328', '105062', '6286', '5.983499218' ]  
[ 'Philippines', 'Philippines', 'PHL', '112133868', 'Asia', '3679485', '59343', '32813', '529', '1.612807227' ]  
[ 'Poland', 'Poland', 'POL', '37774045', 'Europe', '5969621', '115345', '158035', '3054', '1.932199716' ]  
[ 'Portugal', 'Portugal', 'PRT', '10144662', 'Europe', '3604114', '21693', '355272', '2138', '0.601895501' ]  
[ 'Qatar', 'Qatar', 'QAT', '2807805', 'Asia', '361819', '677', '128862', '241', '0.18711013' ]  
[ 'RÃ©union', 'RÃ©union', 'REU', '906497', 'Africa', '336945', '709', '371700', '782', '0.210420098' ]  
[ 'Romania', 'Romania', 'ROU', '19013049', 'Europe', '2860094', '65090', '150428', '3423', '2.275799327' ]

[ 'Russia', 'Russian Federation', 'RUS', '146044010', 'Europe', '17896866',  
'369708', '122544', '2531', '2.065769504']  
[ 'Rwanda', 'Rwanda', 'RWA', '13513881', 'Africa', '129728', '1458', '9600', '108',  
'1.123889985']  
[ 'S. Korea', 'Republic of Korea', 'KOR', '51346429', 'Asia', '13874216', '17235',  
'270208', '336', '0.124223235']  
[ 'Saint Helena', 'Saint Helena', 'SHN', '6109', 'Africa', '2', '0', '327', '0',  
'0']  
[ 'Saint Kitts and Nevis', 'Saint Kitts and Nevis', 'KNA', '53858', 'Latin America  
and the Caribbean', '5549', '43', '103030', '798', '0.774914399']  
[ 'Saint Lucia', 'Saint Lucia', 'LCA', '185096', 'Latin America and the Caribbean',  
'22964', '365', '124065', '1972', '1.589444348']  
[ 'Saint Martin', 'Saint Martin', 'MAF', '39820', 'Latin America and the Caribbean',  
'10107', '63', '253817', '1582', '0.623330365']  
[ 'Saint Pierre Miquelon', '\xa0Saint Pierre and Miquelon', 'SPM', '5744', 'Northern  
America', '1957', '1', '340703', '174', '0.05109862']  
[ 'Samoa', 'Samoa', 'WSM', '200722', 'Oceania', '2285', '1', '11384', '5',  
'0.043763676']  
[ 'San Marino', 'San Marino', 'SMR', '34056', 'Europe', '15181', '113', '445766',  
'3318', '0.744351492']  
[ 'Sao Tome and Principe', 'Sao Tome and Principe', 'STP', '226281', 'Africa',  
'5945', '73', '26273', '323', '1.227922624']  
[ 'Saudi Arabia', 'Saudi Arabia', 'SAU', '35762746', 'Asia', '751076', '9048',  
'21002', '253', '1.204671698']  
[ 'Senegal', 'Senegal', 'SEN', '17515750', 'Africa', '85919', '1965', '4905', '112',  
'2.287037791']  
[ 'Serbia', 'Serbia', 'SRB', '8675762', 'Europe', '1980722', '15825', '228305',  
'1824', '0.79895109']  
[ 'Seychelles', 'Seychelles', 'SYC', '99413', 'Africa', '40421', '164', '406597',  
'1650', '0.405729695']  
[ 'Sierra Leone', 'Sierra Leone', 'SLE', '8260822', 'Africa', '7674', '125', '929',  
'15', '1.628876727']  
[ 'Singapore', 'Singapore', 'SGP', '5930887', 'Asia', '1109744', '1276', '187113',  
'215', '0.114981473']  
[ 'Sint Maarten', 'Sint Maarten', 'SXM', '43728', 'Latin America and the Caribbean',  
'9766', '86', '223335', '1967', '0.880606185']  
[ 'Slovakia', 'Slovakia', 'SVK', '5464272', 'Europe', '1725487', '19417', '315776',  
'3553', '1.125305493']  
[ 'Slovenia', 'Slovenia', 'SVN', '2079438', 'Europe', '973892', '6501', '468344',  
'3126', '0.667527816']  
[ 'Solomon Islands', 'Solomon Islands', 'SLB', '716351', 'Oceania', '11470', '133',  
'16012', '186', '1.159546643']  
[ 'Somalia', 'Somalia', 'SOM', '16668781', 'Africa', '26400', '1348', '1584', '81',  
'5.106060606']  
[ 'South Africa', 'South Africa', 'ZAF', '60617532', 'Africa', '3722954', '100050',  
'61417', '1651', '2.687382116']  
[ 'South Sudan', 'South Sudan', 'SSD', '11423439', 'Africa', '17278', '138', '1513',  
'12', '0.798703554']  
[ 'Spain', 'Spain', 'ESP', '46786482', 'Europe', '11551574', '102541', '246900',  
'2192', '0.887679895']  
[ 'Sri Lanka', 'Sri Lanka', 'LKA', '21570428', 'Asia', '661991', '16481', '30690',  
'764', '2.489610886']  
[ 'St. Barth ', 'Saint Barthélemy', 'BLM', '9930', 'Latin America and the  
Caribbean', '4150', '6', '417925', '604', '0.144578313']  
[ 'St. Vincent Grenadines', 'Saint Vincent and the Grenadines', 'VCT', '111557',  
'Latin America and the Caribbean', '6746', '106', '60471', '950', '1.571301512']  
[ 'Sudan', 'Sudan', 'SDN', '45640385', 'Africa', '61955', '4907', '1357', '108',  
'7.920264708']  
[ 'Suriname', 'Suriname', 'SUR', '595833', 'Latin America and the Caribbean',  
'79232', '1325', '132977', '2224', '1.67230412']  
[ 'Sweden', 'Sweden', 'SWE', '10209507', 'Europe', '2487852', '18331', '243680',  
'1795', '0.736820357']  
[ 'Switzerland', 'Switzerland', 'CHE', '8765420', 'Europe', '3490876', '13715',  
'398255', '1565', '0.392881328']  
[ 'Syria', 'Syrian Arab Republic', 'SYR', '18244381', 'Asia', '55711', '3144',  
'3054', '172', '5.64340974']  
[ 'Taiwan', 'China, Taiwan Province of China', 'TWN', '23892241', 'Asia', '24310',  
'853', '1017', '36', '3.508844097']  
[ 'Tajikistan', 'Tajikistan', 'TJK', '9912437', 'Asia', '17388', '124', '1754',  
'13', '0.713135496']  
[ 'Tanzania', 'United Republic of Tanzania', 'TZA', '62710097', 'Africa', '33815',  
'800', '539', '13', '2.365813988']  
[ 'Thailand', 'Thailand', 'THA', '70106601', 'Asia', '3711595', '25418', '52942',  
'363', '0.684826874']  
[ 'Timor-Leste', 'Timor-Leste', 'TLS', '1362386', 'Asia', '22832', '130', '16759',  
'95', '0.569376314']  
[ 'Togo', 'Togo', 'TGO', '8618172', 'Africa', '36944', '272', '4287', '32',  
'0.736249459']  
[ 'Tonga', 'Tonga', 'TON', '107792', 'Oceania', '7127', '9', '66118', '83',  
'0.126280342']  
[ 'Trinidad and Tobago', 'Trinidad and Tobago', 'TTO', '1407422', 'Latin America and  
the Caribbean', '138425', '3756', '98354', '2669', '2.713382698']  
[ 'Tunisia', 'Tunisia', 'TUN', '12035092', 'Africa', '1035884', '28323', '86072',  
'2353', '2.734186453']  
[ 'Turkey', 'Turkey', 'TUR', '85927644', 'Asia', '14894731', '98157', '173340',  
'1142', '0.659004852']  
[ 'Turks and Caicos', 'Turks and Caicos Islands', 'TCA\x01', '39634', 'Latin

```

['America and the Caribbean', '5910', '36', '149114', '908', '0.609137056']
['UAE', 'United Arab Emirates', 'ARE', '10099567', 'Asia', '892170', '2302',
'88337', '228', '0.258022574']
['Uganda', 'Uganda', 'UGA', '48267221', 'Africa', '163936', '3595', '3396', '74',
'2.192928948']
['UK', 'United Kingdom of Great Britain and Northern Ireland', 'GBR', '68510300',
'Europe', '21216874', '165570', '309689', '2417', '0.780369436']
['Ukraine', 'Ukraine', 'UKR', '43273831', 'Europe', '4968881', '107980', '114824',
'2495', '2.173125096']
['Uruguay', 'Uruguay', 'URY', '3494806', 'Latin America and the Caribbean',
'889513', '7166', '254524', '2050', '0.805609362']
['USA', 'United States of America', 'USA', '334400597', 'Northern America',
'81839052', '1008222', '244734', '3015', '1.231957086']
['Uzbekistan', 'Uzbekistan', 'UZB', '34318156', 'Asia', '237853', '1637', '6931',
'48', '0.688240216']
['Vanuatu', 'Vanuatu', 'VUT', '319701', 'Oceania', '4107', '2', '12846', '6',
'0.048697346']
['Vatican City', 'Holy See', 'VAT', '805', 'Europe', '29', '0', '36025', '0', '0']
['Venezuela', 'Venezuela (Bolivarian Republic of)', 'VEN', '28294895', 'Latin
America and the Caribbean', '520843', '5686', '18408', '201', '1.091691738']
['Vietnam', 'Viet Nam', 'VNM', '98871712', 'Asia', '9818328', '42600', '99304',
'431', '0.433882429']
['Wallis and Futuna', 'Wallis and Futuna Islands', 'WLF', '10894', 'Oceania',
'454', '7', '41674', '643', '1.54185022']
['Western Sahara', 'Western Sahara', 'ESH\xa0', '623031', 'Africa', '10', '1',
'16', '2', '10']
['Yemen', 'Yemen', 'YEM', '30975258', 'Asia', '11806', '2143', '381', '69',
'18.15178723']
['Zambia', 'Zambia', 'ZMB', '19284482', 'Africa', '317076', '3967', '16442', '206',
'1.251119605']
['Zimbabwe', 'Zimbabwe', 'ZWE', '15241601', 'Africa', '246525', '5446', '16174',
'357', '2.209106581']

```

Το κόμμα (,) χρησιμοποιείται ως προκαθορισμένη επιλογή σαν διαχωριστικό στηλών. Μπορούμε ρητά να δηλώσουμε ποιο θα είναι το διαχωριστικό με την επιλογή *delimiter*.

```

with open(local_file2, 'r') as file:
 reader = csv.reader(file, delimiter = ';', quoting=csv.QUOTE_ALL)
 for row in reader:
 print(row)

```

```
['Country', 'Other names', 'ISO 3166-1 alpha-3 CODE', 'Population', 'Continent',
'Total Cases', 'Total Deaths', 'Tot\xA0Cases//1M pop', 'Tot\xA0Deaths/1M pop',
'Death percentage']
['Afghanistan', 'Afghanistan', 'AFG', '40462186', 'Asia', '177827', '7671', '4395',
'190', '4.313743132']
['Albania', 'Albania', 'ALB', '2872296', 'Europe', '273870', '3492', '95349',
'1216', '1.275057509']
['Algeria', 'Algeria', 'DZA', '45236699', 'Africa', '265691', '6874', '5873',
'152', '2.587215976']
['Andorra', 'Andorra', 'AND', '77481', 'Europe', '40024', '153', '516565', '1975',
'0.382270638']
['Angola', 'Angola', 'AGO', '34654212', 'Africa', '99194', '1900', '2862', '55',
'1.915438434']
['Anguilla', 'Anguilla', 'AIA', '15237', 'Latin America and the Caribbean', '2700',
'9', '177200', '591', '0.333333333']
['Antigua and Barbuda', 'Antigua and Barbuda', 'ATG', '99348', 'Latin America and
the Caribbean', '7493', '135', '75422', '1359', '1.801681569']
['Argentina', 'Argentina', 'ARG', '45921761', 'Latin America and the Caribbean',
'9041124', '128065', '196881', '2789', '1.416472111']
['Armenia', 'Armenia', 'ARM', '2972939', 'Asia', '422574', '8617', '142140',
'2898', '2.039169471']
['Aruba', 'Aruba', 'ABW', '107560', 'Latin America and the Caribbean', '34051',
'212', '316577', '1971', '0.622595518']
['Australia', 'Australia', 'AUS', '26017767', 'Oceania', '4680816', '6384',
'179908', '245', '0.136386476']
['Austria', 'Austria', 'AUT', '9096360', 'Europe', '3887355', '15985', '427353',
'1757', '0.411205048']
['Azerbaijan', 'Azerbaijan', 'AZE', '10299156', 'Asia', '792061', '9697', '76905',
'942', '1.224274393']
['Bahamas', 'Bahamas', 'BHM', '399822', 'Latin America and the Caribbean', '33295',
'788', '83275', '1971', '2.36672173']
['Bahrain', 'Bahrain', 'BHR', '1804995', 'Asia', '556241', '1471', '308168', '815',
'0.264453717']
['Bangladesh', 'Bangladesh', 'BGD', '167561502', 'Asia', '1951770', '29122',
'11648', '174', '1.492081546']
['Barbados', 'Barbados', 'BRB', '287991', 'Latin America and the Caribbean',
'59938', '375', '208125', '1302', '0.625646501']
['Belarus', 'Belarus', 'BLR', '9443882', 'Europe', '965322', '6844', '102217',
'725', '0.708986224']
['Belgium', 'Belgium', 'BEL', '11677924', 'Europe', '3851048', '30826', '329772',
'2640', '0.800457434']
['Belize', 'Belize', 'BLZ', '410260', 'Latin America and the Caribbean', '57289',
'656', '139641', '1599', '1.14507148']
```

[ 'Benin', 'Benin', 'BEN', '12678649', 'Africa', '26952', '163', '2126', '13', '0.604778866' ]  
[ 'Bermuda', 'Bermuda', 'BMU', '61875', 'Northern America', '12564', '128', '203055', '2069', '1.018783827' ]  
[ 'Bhutan', 'Bhutan', 'BTN', '786480', 'Asia', '31437', '12', '39972', '15', '0.038171581' ]  
[ 'Bolivia', 'Bolivia (Plurinational State of)', 'BOL', '11951714', 'Latin America and the Caribbean', '902448', '21896', '75508', '1832', '2.426289382' ]  
[ 'Bosnia and Herzegovina', 'Bosnia and Herzegovina', 'BIH', '3245097', 'Europe', '375693', '15719', '115773', '4844', '4.184001299' ]  
[ 'Botswana', 'Botswana', 'BWA', '2434708', 'Africa', '305526', '2686', '125488', '1103', '0.879139582' ]  
[ 'Brazil', 'Brazil', 'BRA', '215204501', 'Latin America and the Caribbean', '29999816', '660269', '139401', '3068', '2.200910166' ]  
[ 'British Virgin Islands', 'British Virgin Islands', 'VGB', '30583', 'Latin America and the Caribbean', '6155', '62', '201256', '2027', '1.007311129' ]  
[ 'Brunei', 'Brunei Darussalam', 'BRN', '444812', 'Asia', '135974', '213', '305689', '479', '0.156647594' ]  
[ 'Bulgaria', 'Bulgaria', 'BGR', '6856886', 'Europe', '1140679', '36568', '166355', '5333', '3.205809873' ]  
[ 'Burkina Faso', 'Burkina Faso', 'BFA', '21905848', 'Africa', '20853', '382', '952', '17', '1.831870714' ]  
[ 'Burundi', 'Burundi', 'BDI', '12510155', 'Africa', '38519', '38', '3079', '3', '0.098652613' ]  
[ 'Cabo Verde', 'Cabo Verde', 'CPV', '566557', 'Africa', '55960', '401', '98772', '708', '0.716583274' ]  
[ 'Cambodia', 'Cambodia', 'KHM', '17123941', 'Asia', '135747', '3054', '7927', '178', '2.249773476' ]  
[ 'Cameroon', 'Cameroon', 'CMR', '27701805', 'Africa', '119544', '1927', '4315', '70', '1.611958777' ]  
[ 'Canada', 'Canada', 'CAN', '38321435', 'Northern America', '3499226', '37690', '91312', '984', '1.077095335' ]  
[ 'CAR', 'Central African Republic', 'CAF', '4976719', 'Africa', '14649', '113', '2944', '23', '0.771383712' ]  
[ 'Caribbean Netherlands', 'Bonaire, Sint Eustatius and Saba', 'BES', '26650', 'Latin America and the Caribbean', '8574', '33', '321726', '1238', '0.384884535' ]  
[ 'Cayman Islands', 'Cayman Islands', 'CYM', '67073', 'Latin America and the Caribbean', '20606', '24', '307218', '358', '0.116470931' ]  
[ 'Chad', 'Chad', 'TCD', '17250246', 'Africa', '7308', '191', '424', '11', '2.613574165' ]  
[ 'Channel Islands', 'Guernsey', 'GGY', '176668', 'Europe', '69036', '156', '390767', '883', '0.22596906' ]  
[ 'Chile', 'Chile', 'CHL', '19403451', 'Latin America and the Caribbean', '3486653', '56750', '179692', '2925', '1.627635443' ]  
[ 'China', 'China', 'CHN', '1439323776', 'Asia', '154738', '4638', '108', '3', '2.99732451' ]  
[ 'Colombia', 'Colombia', 'COL', '51832231', 'Latin America and the Caribbean', '6085926', '139660', '117416', '2694', '2.294802796' ]  
[ 'Comoros', 'Comoros', 'COM', '902011', 'Africa', '8093', '160', '8972', '177', '1.977017175' ]  
[ 'Congo', 'Congo', 'COG', '5755689', 'Africa', '24071', '385', '4182', '67', '1.599435005' ]  
[ 'Cook Islands', 'Cook Islands', 'COK', '17592', 'Oceania', '2118', '0', '120396', '0', '0' ]  
[ 'Costa Rica', 'Costa Rica', 'CRI', '5175547', 'Latin America and the Caribbean', '839368', '8308', '162180', '1605', '0.98979232' ]  
[ 'Croatia', 'Croatia', 'HRV', '4060951', 'Europe', '1102730', '15601', '271545', '3842', '1.414761546' ]  
[ 'Cuba', 'Cuba', 'CUB', '11314513', 'Latin America and the Caribbean', '1092547', '8514', '96562', '752', '0.779279976' ]  
[ 'Curaçao', 'Curaçao', 'CUW', '165268', 'Latin America and the Caribbean', '40671', '267', '246091', '1616', '0.656487423' ]  
[ 'Cyprus', 'Cyprus', 'CYP', '1222745', 'Asia', '439964', '947', '359817', '774', '0.215244884' ]  
[ 'Czechia', 'Czech Republic', 'CZE', '10743762', 'Europe', '3830631', '39720', '356545', '3697', '1.036904886' ]  
[ 'Denmark', 'Denmark', 'DNK', '5827911', 'Europe', '2919428', '5762', '500939', '989', '0.19736743' ]  
[ 'Djibouti', 'Djibouti', 'DJI', '1013146', 'Africa', '15590', '189', '15388', '187', '1.212315587' ]  
[ 'Dominica', 'Dominica', 'DMA', '72299', 'Latin America and the Caribbean', '11891', '63', '164470', '871', '0.529812463' ]  
[ 'Dominican Republic', 'Dominican Republic', 'DOM', '11038333', 'Latin America and the Caribbean', '578130', '4375', '52375', '396', '0.756750212' ]  
[ 'Democratic Republic of the Congo', 'Democratic Republic of the Congo', 'COD', '94323344', 'Africa', '86748', '1337', '920', '14', '1.541245908' ]  
[ 'Ecuador', 'Ecuador', 'ECU', '18111933', 'Latin America and the Caribbean', '859890', '35421', '47476', '1956', '4.119247811' ]  
[ 'Egypt', 'Egypt', 'EGY', '105711844', 'Africa', '505264', '24417', '4780', '231', '4.832523196' ]  
[ 'El Salvador', 'El Salvador', 'SLV', '6543499', 'Latin America and the Caribbean', '161570', '4120', '24692', '630', '2.549978338' ]  
[ 'Equatorial Guinea', 'Equatorial Guinea', 'GNQ', '1483588', 'Africa', '15903', '183', '10719', '123', '1.150726278' ]  
[ 'Eritrea', 'Eritrea', 'ERI', '3632329', 'Africa', '9728', '103', '2678', '28', '1.058799342' ]  
[ 'Estonia', 'Estonia', 'EST', '1328097', 'Europe', '558706', '2468', '420682',

'1858', '0.441735009']  
[['Eswatini', 'Eswatini', 'SWZ', '1181191', 'Africa', '69851', '1394', '59136',  
'1180', '1.995676511']]  
[['Ethiopia', 'Ethiopia', 'ETH', '119945147', 'Africa', '469819', '7504', '3917',  
'63', '1.597210841']]  
[['Faeroe Islands', 'Faeroe Islands', 'FRO', '49188', 'Europe', '34237', '28',  
'696044', '569', '0.081782866']]  
[['Falkland Islands', 'Falkland Islands (Malvinas)', 'FLK', '3657', 'Latin America  
and the Caribbean', '123', '0', '33634', '0', '0']]  
[['Fiji', 'Fiji', 'FJI', '907817', 'Oceania', '64422', '834', '70964', '919',  
'1.294588805']]  
[['Finland', 'Finland', 'FIN', '5555788', 'Europe', '889626', '3178', '160126',  
'572', '0.357228768']]  
[['France', 'France', 'FRA', '65526369', 'Europe', '25997852', '142506', '396754',  
'2175', '0.548145285']]  
[['French Guiana', 'French Guiana', 'GUF', '312224', 'Latin America and the  
Caribbean', '79075', '394', '253264', '1262', '0.498261144']]  
[['French Polynesia', 'French Polynesia', 'PYF', '283751', 'Oceania', '72318',  
'646', '254864', '2277', '0.893276916']]  
[['Gabon', 'Gabon', 'GAB', '2317612', 'Africa', '47586', '303', '20532', '131',  
'0.636741899']]  
[['Gambia', 'Gambia', 'GMB', '2535418', 'Africa', '11988', '365', '4728', '144',  
'3.044711378']]  
[['Georgia', 'Georgia', 'GEO', '3975762', 'Asia', '1649222', '16756', '414819',  
'4215', '1.015994208']]  
[['Germany', 'Germany', 'DEU', '84252947', 'Europe', '21646375', '130563', '256921',  
'1550', '0.603163347']]  
[['Ghana', 'Ghana', 'GHA', '32207812', 'Africa', '160971', '1445', '4998', '45',  
'0.897677221']]  
[['Gibraltar', 'Gibraltar', 'GIB', '33673', 'Europe', '16979', '101', '504232',  
'2999', '0.594852465']]  
[['Greece', 'Greece', 'GRC', '10333930', 'Europe', '3077711', '27684', '297826',  
'2679', '0.899499661']]  
[['Greenland', 'Greenland', 'GRL', '56942', 'Northern America', '11971', '21',  
'210231', '369', '0.175423941']]  
[['Grenada', 'Grenada', 'GRD', '113436', 'Latin America and the Caribbean', '14024',  
'218', '123629', '1922', '1.554478038']]  
[['Guadeloupe', 'Guadeloupe', 'GLP', '400244', 'Latin America and the Caribbean',  
'130705', '843', '326563', '2106', '0.64496385']]  
[['Guatemala', 'Guatemala', 'GTM', '18495493', 'Latin America and the Caribbean',  
'830745', '17325', '44916', '937', '2.085477493']]  
[['Guinea', 'Guinea', 'GIN', '13755881', 'Africa', '36459', '440', '2650', '32',  
'1.206835075']]  
[['Guinea-Bissau', 'Guinea-Bissau', 'GNB', '2049374', 'Africa', '8151', '170',  
'3977', '83', '2.085633665']]  
[['Guyana', 'Guyana', 'GUY', '793196', 'Latin America and the Caribbean', '63272',  
'1226', '79768', '1546', '1.93766595']]  
[['Haiti', 'Haiti', 'HTI', '11645833', 'Latin America and the Caribbean', '30549',  
'833', '2623', '72', '2.726766834']]  
[['Honduras', 'Honduras', 'HND', '10180299', 'Latin America and the Caribbean',  
'421062', '10880', '41360', '1069', '2.583942507']]  
[['Hong Kong', 'China, Hong Kong Special Administrative Region', 'HKG', '7603455',  
'Asia', '1171422', '8172', '154064', '1075', '0.69761367']]  
[['Hungary', 'Hungary', 'HUN', '9617409', 'Europe', '1854198', '45510', '192796',  
'4732', '2.454430433']]  
[['Iceland', 'Iceland', 'ISL', '345120', 'Europe', '181830', '101', '526860', '293',  
'0.055546389']]  
[['India', 'India', 'IND', '1403754381', 'Asia', '43029044', '521388', '30653',  
'371', '1.211711792']]  
[['Indonesia', 'Indonesia', 'IDN', '278586508', 'Asia', '6019981', '155288',  
'21609', '557', '2.579543025']]  
[['Iran', 'United Kingdom', 'IRN', '85874667', 'Asia', '7167646', '140315', '83466',  
'1634', '1.95761621']]  
[['Iraq', 'Iraq', 'IRQ', '41801625', 'Asia', '2320260', '25173', '55506', '602',  
'1.084921517']]  
[['Ireland', 'Ireland', 'IRL', '5034333', 'Europe', '1471210', '6786', '292235',  
'1348', '0.461252982']]  
[['Isle of Man', 'Isle of Man', 'IMN', '85821', 'Europe', '28416', '84', '331108',  
'979', '0.295608108']]  
[['Israel', 'Israel', 'ISR', '9326000', 'Asia', '3943153', '10530', '422813',  
'1129', '0.267045179']]  
[['Italy', 'Italy', 'ITA', '60306185', 'Europe', '14846514', '159784', '246186',  
'2650', '1.076239176']]  
[['Ivory Coast', 'Côte d'Ivoire', 'CIV', '27520953', 'Africa', '81761', '796',  
'2971', '29', '0.973569306']]  
[['Jamaica', 'Jamaica', 'JAM', '29837941', 'Latin America and the Caribbean',  
'128811', '2893', '43170', '970', '2.245926202']]  
[['Japan', 'Japan', 'JPN', '125798669', 'Asia', '6653841', '28248', '52893', '225',  
'0.424536745']]  
[['Jordan', 'Jordan', 'JOR', '10380442', 'Asia', '1689314', '14003', '162740',  
'1349', '0.828916353']]  
[['Kazakhstan', 'Kazakhstan', 'KAZ', '19169833', 'Asia', '1305188', '13660',  
'68086', '713', '1.046592522']]  
[['Kenya', 'Kenya', 'KEN', '55843563', 'Africa', '323454', '5648', '5792', '101',  
'1.746152467']]  
[['Kiribati', 'Kiribati', 'KIR', '122656', 'Oceania', '3067', '13', '25005', '106',  
'0.423866971']]

[ 'Kuwait', 'Kuwait', 'KWT', '4381108', 'Asia', '629525', '2554', '143691', '583', '0.405702712' ]  
[ 'Kyrgyzstan', 'Kyrgyzstan', 'KGZ', '6712569', 'Asia', '200968', '2991', '29939', '446', '1.488296644' ]  
[ 'Laos', "Lao People's Democratic Republic", 'LAO', '7460338', 'Asia', '183560', '679', '24605', '91', '0.369906298' ]  
[ 'Latvia', 'Latvia', 'LVA', '1849698', 'Europe', '802534', '5643', '433873', '3051', '0.703147779' ]  
[ 'Lebanon', 'Lebanon', 'LBN', '6771939', 'Asia', '1092995', '10315', '161401', '1523', '0.943737163' ]  
[ 'Lesotho', 'Lesotho', 'LSO', '2171978', 'Africa', '32910', '697', '15152', '321', '2.117897296' ]  
[ 'Liberia', 'Liberia', 'LBR', '5265647', 'Africa', '7400', '294', '1405', '56', '3.972972973' ]  
[ 'Libya', 'Libya', 'LBY', '7034832', 'Africa', '501738', '6419', '71322', '912', '1.279352969' ]  
[ 'Liechtenstein', 'Liechtenstein', 'LIE', '38320', 'Europe', '16429', '84', '428732', '2192', '0.51129101' ]  
[ 'Lithuania', 'Lithuania', 'LTU', '2655811', 'Europe', '1030966', '8907', '388193', '3354', '0.863947017' ]  
[ 'Luxembourg', 'Luxembourg', 'LUX', '643801', 'Europe', '216979', '1037', '337028', '1611', '0.477926435' ]  
[ 'Macao', 'China, Macao Special Administrative Region', 'MAC', '664828', 'Asia', '82', '0', '123', '0', '0' ]  
[ 'Madagascar', 'Madagascar', 'MDG', '28936285', 'Africa', '64050', '1388', '2213', '48', '2.167056987' ]  
[ 'Malawi', 'Malawi', 'MWI', '19994654', 'Africa', '85664', '2626', '4284', '131', '3.065465073' ]  
[ 'Malaysia', 'Malaysia', 'MYS', '33091831', 'Asia', '4246467', '35099', '128324', '1061', '0.826545926' ]  
[ 'Maldives', 'Maldives', 'MDV', '557204', 'Asia', '176993', '298', '317645', '535', '0.168368241' ]  
[ 'Mali', 'Mali', 'MLI', '21271006', 'Africa', '30495', '728', '1434', '34', '2.387276603' ]  
[ 'Malta', 'Malta', 'MLT', '443602', 'Europe', '81596', '641', '183940', '1445', '0.785577724' ]  
[ 'Marshall Islands', 'Marshall Islands', 'MHL', '59889', 'Oceania', '7', '0', '117', '0', '0' ]  
[ 'Martinique', 'Martinique', 'MTQ', '374756', 'Latin America and the Caribbean', '141415', '909', '377352', '2426', '0.642788954' ]  
[ 'Mauritania', 'Mauritania', 'MRT', '4863443', 'Africa', '58670', '982', '12063', '202', '1.673768536' ]  
[ 'Mauritius', 'Mauritius', 'MUS', '1275463', 'Africa', '36628', '968', '28717', '759', '2.642786939' ]  
[ 'Mayotte', 'Mayotte', 'MYT', '284330', 'Africa', '36891', '187', '129747', '658', '0.506898702' ]  
[ 'Mexico', 'Mexico', 'MEX', '131303955', 'Latin America and the Caribbean', '5665376', '323212', '43147', '2462', '15.705040583' ]  
[ 'Micronesia', 'Micronesia (Federated States of)', 'FSM', '117134', 'Oceania', '1', '0', '9', '0', '0' ]  
[ 'Moldova', 'Republic of Moldova', 'MDA', '4017550', 'Europe', '514199', '11446', '127988', '2849', '2.225986437' ]  
[ 'Monaco', 'Monaco', 'MCO', '39729', 'Europe', '10842', '54', '272899', '1359', '0.498063088' ]  
[ 'Mongolia', 'Mongolia', 'MNG', '3370682', 'Asia', '468610', '2177', '139025', '646', '0.464565417' ]  
[ 'Montenegro', 'Montenegro', 'MNE', '628205', 'Europe', '233326', '2705', '371417', '4306', '1.15932215' ]  
[ 'Montserrat', 'Montserrat', 'MSR', '4997', 'Latin America and the Caribbean', '175', '2', '35021', '400', '1.142857143' ]  
[ 'Morocco', 'Morocco', 'MAR', '37676342', 'Africa', '1163526', '16060', '30882', '426', '1.380287162' ]  
[ 'Mozambique', 'Mozambique', 'MOZ', '32787052', 'Africa', '225266', '2200', '6871', '67', '0.976623192' ]  
[ 'Myanmar', 'Myanmar', 'MMR', '55048340', 'Asia', '611875', '19433', '11115', '353', '3.175975485' ]  
[ 'Namibia', 'Namibia', 'NAM', '2621429', 'Africa', '157646', '4019', '60137', '1533', '2.549382794' ]  
[ 'Nepal', 'Nepal', 'NPL', '30053867', 'Asia', '978475', '11951', '32557', '398', '1.221390429' ]  
[ 'Netherlands', 'Netherlands', 'NLD', '17201245', 'Europe', '7908701', '22016', '459775', '1280', '0.278376942' ]  
[ 'New Caledonia', 'New Caledonia', 'NCL', '290302', 'Oceania', '60294', '311', '207694', '1071', '0.515805884' ]  
[ 'New Zealand', 'New Zealand', 'NZL', '5002100', 'Oceania', '693219', '350', '138586', '70', '0.050489095' ]  
[ 'Nicaragua', 'Nicaragua', 'NIC', '6762511', 'Latin America and the Caribbean', '18434', '224', '2726', '33', '1.215145926' ]  
[ 'Niger', 'Niger', 'NER', '25738714', 'Africa', '8811', '308', '342', '12', '3.495630462' ]  
[ 'Nigeria', 'Nigeria', 'NGA', '215077352', 'Africa', '255468', '3142', '1188', '15', '1.229899635' ]  
[ 'Niue', 'Niue', 'NIU', '1645', 'Oceania', '7', '0', '4255', '0', '0' ]  
[ 'North Macedonia', 'The former Yugoslav Republic of Macedonia', 'MKD', '2083224', 'Europe', '306670', '9228', '147209', '4430', '3.009097727' ]  
[ 'Norway', 'Norway', 'NOR', '5495449', 'Europe', '1408708', '2518', '256341', '458', '0.178745347' ]

[ 'Oman', 'Oman', 'OMN', '5333815', 'Asia', '388468', '4251', '72831', '797', '1.094298629' ]  
[ 'Pakistan', 'Pakistan', 'PAK', '228397520', 'Asia', '1525466', '30361', '6679', '133', '1.990277069' ]  
[ 'Palau', 'Palau', 'PLW', '18245', 'Oceania', '4042', '6', '221540', '329', '0.148441366' ]  
[ 'Palestine', 'State of Palestine', 'WBG', '5308883', 'Asia', '581236', '5351', '109484', '1008', '0.920624325' ]  
[ 'Panama', 'Panama', 'PAN', '4433639', 'Latin America and the Caribbean', '765213', '8170', '172593', '1843', '1.067676582' ]  
[ 'Papua New Guinea', 'Papua New Guinea', 'PNG', '9243590', 'Oceania', '42203', '640', '4566', '69', '1.516479871' ]  
[ 'Paraguay', 'Paraguay', 'PRY', '7285892', 'Latin America and the Caribbean', '648353', '18731', '88987', '2571', '2.889012621' ]  
[ 'Peru', 'Peru', 'PER', '33775745', 'Latin America and the Caribbean', '3548559', '212328', '105062', '6286', '5.983499218' ]  
[ 'Philippines', 'Philippines', 'PHL', '112133868', 'Asia', '3679485', '59343', '32813', '529', '1.612807227' ]  
[ 'Poland', 'Poland', 'POL', '37774045', 'Europe', '5969621', '115345', '158035', '3054', '1.932199716' ]  
[ 'Portugal', 'Portugal', 'PRT', '10144662', 'Europe', '3604114', '21693', '355272', '2138', '0.601895501' ]  
[ 'Qatar', 'Qatar', 'QAT', '2807805', 'Asia', '361819', '677', '128862', '241', '0.18711013' ]  
[ 'Réunion', 'RÃ©union', 'REU', '906497', 'Africa', '336945', '709', '371700', '782', '0.210420098' ]  
[ 'Romania', 'Romania', 'ROU', '19013049', 'Europe', '2860094', '65090', '150428', '3423', '2.275799327' ]  
[ 'Russia', 'Russian Federation', 'RUS', '146044010', 'Europe', '17896866', '369708', '122544', '2531', '2.065769504' ]  
[ 'Rwanda', 'Rwanda', 'RWA', '13513881', 'Africa', '129728', '1458', '9600', '108', '1.123889985' ]  
[ 'S. Korea', 'Republic of Korea', 'KOR', '51346429', 'Asia', '13874216', '17235', '270208', '336', '0.124223235' ]  
[ 'Saint Helena', 'Saint Helena', 'SHN', '6109', 'Africa', '2', '0', '327', '0', '0' ]  
[ 'Saint Kitts and Nevis', 'Saint Kitts and Nevis', 'KNA', '53858', 'Latin America and the Caribbean', '5549', '43', '103030', '798', '0.774914399' ]  
[ 'Saint Lucia', 'Saint Lucia', 'LCA', '185096', 'Latin America and the Caribbean', '22964', '365', '124065', '1972', '1.589444348' ]  
[ 'Saint Martin', 'Saint Martin', 'MAF', '39820', 'Latin America and the Caribbean', '10107', '63', '253817', '1582', '0.623330365' ]  
[ 'Saint Pierre Miquelon', '\xa0Saint Pierre and Miquelon', 'SPM', '5744', 'Northern America', '1957', '1', '340703', '174', '0.05109862' ]  
[ 'Samoa', 'Samoa', 'WSM', '200722', 'Oceania', '2285', '1', '11384', '5', '0.043763676' ]  
[ 'San Marino', 'San Marino', 'SMR', '34056', 'Europe', '15181', '113', '445766', '3318', '0.744351492' ]  
[ 'Sao Tome and Principe', 'Sao Tome and Principe', 'STP', '226281', 'Africa', '5945', '73', '26273', '323', '1.227922624' ]  
[ 'Saudi Arabia', 'Saudi Arabia', 'SAU', '35762746', 'Asia', '751076', '9048', '21002', '253', '1.204671698' ]  
[ 'Senegal', 'Senegal', 'SEN', '17515750', 'Africa', '85919', '1965', '4905', '112', '2.287037791' ]  
[ 'Serbia', 'Serbia', 'SRB', '8675762', 'Europe', '1980722', '15825', '228305', '1824', '0.79895109' ]  
[ 'Seychelles', 'Seychelles', 'SYC', '99413', 'Africa', '40421', '164', '406597', '1650', '0.405729695' ]  
[ 'Sierra Leone', 'Sierra Leone', 'SLE', '8260822', 'Africa', '7674', '125', '929', '15', '1.628876727' ]  
[ 'Singapore', 'Singapore', 'SGP', '5930887', 'Asia', '1109744', '1276', '187113', '215', '0.114981473' ]  
[ 'Sint Maarten', 'Sint Maarten', 'SXM', '43728', 'Latin America and the Caribbean', '9766', '86', '223335', '1967', '0.880606185' ]  
[ 'Slovakia', 'Slovakia', 'SVK', '5464272', 'Europe', '1725487', '19417', '315776', '3553', '1.125305493' ]  
[ 'Slovenia', 'Slovenia', 'SVN', '2079438', 'Europe', '973892', '6501', '468344', '3126', '0.667527816' ]  
[ 'Solomon Islands', 'Solomon Islands', 'SLB', '716351', 'Oceania', '11470', '133', '16012', '186', '1.159546643' ]  
[ 'Somalia', 'Somalia', 'SOM', '16668781', 'Africa', '26400', '1348', '1584', '81', '5.106060606' ]  
[ 'South Africa', 'South Africa', 'ZAF', '60617532', 'Africa', '3722954', '100050', '61417', '1651', '2.687382116' ]  
[ 'South Sudan', 'South Sudan', 'SSD', '11423439', 'Africa', '17278', '138', '1513', '12', '0.798703554' ]  
[ 'Spain', 'Spain', 'ESP', '46786482', 'Europe', '11551574', '102541', '246900', '2192', '0.887679895' ]  
[ 'Sri Lanka', 'Sri Lanka', 'LKA', '21570428', 'Asia', '661991', '16481', '30690', '764', '2.489610886' ]  
[ 'St. Barth', 'Saint Barthélemy', 'BLM', '9930', 'Latin America and the Caribbean', '4150', '6', '417925', '604', '0.144578313' ]  
[ 'St. Vincent Grenadines', 'Saint Vincent and the Grenadines', 'VCT', '111557', 'Latin America and the Caribbean', '6746', '106', '60471', '950', '1.571301512' ]  
[ 'Sudan', 'Sudan', 'SDN', '45640385', 'Africa', '61955', '4907', '1357', '108', '7.920264708' ]  
[ 'Suriname', 'Suriname', 'SUR', '595833', 'Latin America and the Caribbean',

```

['79232', '1325', '132977', '2224', '1.67230412'],
['Sweden', 'Sweden', 'SWE', '10209507', 'Europe', '2487852', '18331', '243680',
'1795', '0.736820357'],
['Switzerland', 'Switzerland', 'CHE', '8765420', 'Europe', '3490876', '13715',
'398255', '1565', '0.392881328'],
['Syria', 'Syrian Arab Republic', 'SYR', '18244381', 'Asia', '55711', '3144',
'3054', '172', '5.64340974'],
['Taiwan', 'China, Taiwan Province of China', 'TWN', '23892241', 'Asia', '24310',
'853', '1017', '36', '3.508844097'],
['Tajikistan', 'Tajikistan', 'TJK', '9912437', 'Asia', '17388', '124', '1754',
'13', '0.713135496'],
['Tanzania', 'United Republic of Tanzania', 'TZA', '62710097', 'Africa', '33815',
'800', '539', '13', '2.365813988'],
['Thailand', 'Thailand', 'THA', '70106601', 'Asia', '3711595', '25418', '52942',
'363', '0.684826874'],
['Timor-Leste', 'Timor-Leste', 'TLS', '1362386', 'Asia', '22832', '130', '16759',
'95', '0.569376314'],
['Togo', 'Togo', 'TGO', '8618172', 'Africa', '36944', '272', '4287', '32',
'0.736249459'],
['Tonga', 'Tonga', 'TON', '107792', 'Oceania', '7127', '9', '66118', '83',
'0.126280342'],
['Trinidad and Tobago', 'Trinidad and Tobago', 'TT0', '1407422', 'Latin America and
the Caribbean', '138425', '3756', '98354', '2669', '2.713382698'],
['Tunisia', 'Tunisia', 'TUN', '12035092', 'Africa', '1035884', '28323', '86072',
'2353', '2.734186453'],
['Turkey', 'Turkey', 'TUR', '85927644', 'Asia', '14894731', '98157', '173340',
'1142', '0.659004852'],
['Turks and Caicos Islands', 'Turks and Caicos Islands', 'TCAÃ\xao', '39634', 'Latin
America and the Caribbean', '5910', '36', '149114', '908', '0.609137056'],
['UAE', 'United Arab Emirates', 'ARE', '10099567', 'Asia', '892170', '2302',
'88337', '228', '0.258022574'],
['Uganda', 'Uganda', 'UGA', '48267221', 'Africa', '163936', '3595', '3396', '74',
'2.192928948'],
['UK', 'United Kingdom of Great Britain and Northern Ireland', 'GBR', '68510300',
'Europe', '21216874', '165570', '309689', '2417', '0.780369436'],
['Ukraine', 'Ukraine', 'UKR', '43273831', 'Europe', '4968881', '107980', '114824',
'2495', '2.173125096'],
['Uruguay', 'Uruguay', 'URY', '3494806', 'Latin America and the Caribbean',
'889513', '7166', '254524', '2050', '0.805609362'],
['USA', 'United States of America', 'USA', '334400597', 'Northern America',
'81839052', '1008222', '244734', '3015', '1.231957086'],
['Uzbekistan', 'Uzbekistan', 'UZB', '34318156', 'Asia', '237853', '1637', '6931',
'48', '0.688240216'],
['Vanuatu', 'Vanuatu', 'VUT', '319701', 'Oceania', '4107', '2', '12846', '6',
'0.048697346'],
['Vatican City', 'Holy See', 'VAT', '805', 'Europe', '29', '0', '36025', '0', '0'],
['Venezuela', 'Venezuela (Bolivarian Republic of)', 'VEN', '28294895', 'Latin
America and the Caribbean', '520843', '5686', '18408', '201', '1.091691738'],
['Vietnam', 'Viet Nam', 'VNM', '98871712', 'Asia', '9818328', '42600', '99304',
'431', '0.433882429'],
['Wallis and Futuna', 'Wallis and Futuna Islands', 'WLF', '10894', 'Oceania',
'454', '7', '41674', '643', '1.54185022'],
['Western Sahara', 'Western Sahara', 'ESHÃ\xao', '623031', 'Africa', '10', '1',
'16', '2', '10'],
['Yemen', 'Yemen', 'YEM', '30975258', 'Asia', '11806', '2143', '381', '69',
'18.15178723'],
['Zambia', 'Zambia', 'ZMB', '19284482', 'Africa', '317076', '3967', '16442', '206',
'1.251119605'],
['Zimbabwe', 'Zimbabwe', 'ZWE', '15241601', 'Africa', '246525', '5446', '16174',
'357', '2.209106581']

```

```

with open(local_file1, 'r') as file:
 csv_file = csv.DictReader(file)
 for row in csv_file:
 print(dict(row))

```

```
{"Country": "Afghanistan", "Other names": "Afghanistan", "ISO 3166-1 alpha-3 CODE": "AFG", "Population": "40462186", "Continent": "Asia", "Total Cases": "177827", "Total Deaths": "7671", "Tot\x00Cases//1M pop": "4395", "Tot\x00Deaths/1M pop": "190", "Death percentage": "4.313743132"}, {"Country": "Albania", "Other names": "Albania", "ISO 3166-1 alpha-3 CODE": "ALB", "Population": "2872296", "Continent": "Europe", "Total Cases": "273870", "Total Deaths": "3492", "Tot\x00Cases//1M pop": "95349", "Tot\x00Deaths/1M pop": "1216", "Death percentage": "1.275057509"}, {"Country": "Algeria", "Other names": "Algeria", "ISO 3166-1 alpha-3 CODE": "DZA", "Population": "45236699", "Continent": "Africa", "Total Cases": "265691", "Total Deaths": "6874", "Tot\x00Cases//1M pop": "5873", "Tot\x00Deaths/1M pop": "152", "Death percentage": "2.587215976"}, {"Country": "Andorra", "Other names": "Andorra", "ISO 3166-1 alpha-3 CODE": "AND", "Population": "77481", "Continent": "Europe", "Total Cases": "40024", "Total Deaths": "153", "Tot\x00Cases//1M pop": "516565", "Tot\x00Deaths/1M pop": "1975", "Death percentage": "0.382270638"}, {"Country": "Angola", "Other names": "Angola", "ISO 3166-1 alpha-3 CODE": "AGO", "Population": "34654212", "Continent": "Africa", "Total Cases": "99194", "Total Deaths": "1900", "Tot\x00Cases//1M pop": "2862", "Tot\x00Deaths/1M pop": "55", "Death percentage": "6.671052631"}]
```

```
'Death percentage': '1.915438434'}
{'Country': 'Anguilla', 'Other names': 'Anguilla', 'ISO 3166-1 alpha-3 CODE': 'AIA', 'Population': '15237', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '2700', 'Total Deaths': '9', 'Tot\x00Cases//1M pop': '177200', 'Tot\x00Deaths/1M pop': '591', 'Death percentage': '0.333333333'}
{'Country': 'Antigua and Barbuda', 'Other names': 'Antigua and Barbuda', 'ISO 3166-1 alpha-3 CODE': 'ATG', 'Population': '99348', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '7493', 'Total Deaths': '135', 'Tot\x00Cases//1M pop': '75422', 'Tot\x00Deaths/1M pop': '1359', 'Death percentage': '1.801681569'}
{'Country': 'Argentina', 'Other names': 'Argentina', 'ISO 3166-1 alpha-3 CODE': 'ARG', 'Population': '45921761', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '9041124', 'Total Deaths': '128065', 'Tot\x00Cases//1M pop': '196881', 'Tot\x00Deaths/1M pop': '2789', 'Death percentage': '1.416472111'}
{'Country': 'Armenia', 'Other names': 'Armenia', 'ISO 3166-1 alpha-3 CODE': 'ARM', 'Population': '2972939', 'Continent': 'Asia', 'Total Cases': '422574', 'Total Deaths': '8617', 'Tot\x00Cases//1M pop': '142140', 'Tot\x00Deaths/1M pop': '2898', 'Death percentage': '2.039169471'}
{'Country': 'Aruba', 'Other names': 'Aruba', 'ISO 3166-1 alpha-3 CODE': 'ABW', 'Population': '107560', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '34051', 'Total Deaths': '212', 'Tot\x00Cases//1M pop': '316577', 'Tot\x00Deaths/1M pop': '1971', 'Death percentage': '0.622595518'}
{'Country': 'Australia', 'Other names': 'Australia', 'ISO 3166-1 alpha-3 CODE': 'AUS', 'Population': '26017767', 'Continent': 'Oceania', 'Total Cases': '4680816', 'Total Deaths': '6384', 'Tot\x00Cases//1M pop': '179908', 'Tot\x00Deaths/1M pop': '245', 'Death percentage': '0.136386476'}
{'Country': 'Austria', 'Other names': 'Austria', 'ISO 3166-1 alpha-3 CODE': 'AUT', 'Population': '9096360', 'Continent': 'Europe', 'Total Cases': '3887355', 'Total Deaths': '15985', 'Tot\x00Cases//1M pop': '427353', 'Tot\x00Deaths/1M pop': '1757', 'Death percentage': '0.411205048'}
{'Country': 'Azerbaijan', 'Other names': 'Azerbaijan', 'ISO 3166-1 alpha-3 CODE': 'AZE', 'Population': '10299156', 'Continent': 'Asia', 'Total Cases': '792061', 'Total Deaths': '9697', 'Tot\x00Cases//1M pop': '76905', 'Tot\x00Deaths/1M pop': '942', 'Death percentage': '1.224274393'}
{'Country': 'Bahamas', 'Other names': 'Bahamas', 'ISO 3166-1 alpha-3 CODE': 'BHM', 'Population': '399822', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '33295', 'Total Deaths': '788', 'Tot\x00Cases//1M pop': '83275', 'Tot\x00Deaths/1M pop': '1971', 'Death percentage': '2.36672173'}
{'Country': 'Bahrain', 'Other names': 'Bahrain', 'ISO 3166-1 alpha-3 CODE': 'BHR', 'Population': '1804995', 'Continent': 'Asia', 'Total Cases': '556241', 'Total Deaths': '1471', 'Tot\x00Cases//1M pop': '308168', 'Tot\x00Deaths/1M pop': '815', 'Death percentage': '0.264453717'}
{'Country': 'Bangladesh', 'Other names': 'Bangladesh', 'ISO 3166-1 alpha-3 CODE': 'BGD', 'Population': '167561502', 'Continent': 'Asia', 'Total Cases': '1951770', 'Total Deaths': '29122', 'Tot\x00Cases//1M pop': '11648', 'Tot\x00Deaths/1M pop': '174', 'Death percentage': '1.492081546'}
{'Country': 'Barbados', 'Other names': 'Barbados', 'ISO 3166-1 alpha-3 CODE': 'BRB', 'Population': '287991', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '59938', 'Total Deaths': '375', 'Tot\x00Cases//1M pop': '208125', 'Tot\x00Deaths/1M pop': '1302', 'Death percentage': '0.625646501'}
{'Country': 'Belarus', 'Other names': 'Belarus', 'ISO 3166-1 alpha-3 CODE': 'BLR', 'Population': '9443882', 'Continent': 'Europe', 'Total Cases': '965322', 'Total Deaths': '6844', 'Tot\x00Cases//1M pop': '102217', 'Tot\x00Deaths/1M pop': '725', 'Death percentage': '0.708986224'}
{'Country': 'Belgium', 'Other names': 'Belgium', 'ISO 3166-1 alpha-3 CODE': 'BEL', 'Population': '11677924', 'Continent': 'Europe', 'Total Cases': '3851048', 'Total Deaths': '30826', 'Tot\x00Cases//1M pop': '329772', 'Tot\x00Deaths/1M pop': '2640', 'Death percentage': '0.800457434'}
{'Country': 'Belize', 'Other names': 'Belize', 'ISO 3166-1 alpha-3 CODE': 'BLZ', 'Population': '410260', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '57289', 'Total Deaths': '656', 'Tot\x00Cases//1M pop': '139641', 'Tot\x00Deaths/1M pop': '1599', 'Death percentage': '1.14507148'}
{'Country': 'Benin', 'Other names': 'Benin', 'ISO 3166-1 alpha-3 CODE': 'BEN', 'Population': '12678649', 'Continent': 'Africa', 'Total Cases': '26952', 'Total Deaths': '163', 'Tot\x00Cases//1M pop': '2126', 'Tot\x00Deaths/1M pop': '13', 'Death percentage': '0.604778866'}
{'Country': 'Bermuda', 'Other names': 'Bermuda', 'ISO 3166-1 alpha-3 CODE': 'BMU', 'Population': '61875', 'Continent': 'Northern America', 'Total Cases': '12564', 'Total Deaths': '128', 'Tot\x00Cases//1M pop': '203055', 'Tot\x00Deaths/1M pop': '2069', 'Death percentage': '1.018783827'}
{'Country': 'Bhutan', 'Other names': 'Bhutan', 'ISO 3166-1 alpha-3 CODE': 'BTN', 'Population': '786480', 'Continent': 'Asia', 'Total Cases': '31437', 'Total Deaths': '12', 'Tot\x00Cases//1M pop': '39972', 'Tot\x00Deaths/1M pop': '15', 'Death percentage': '0.038171581'}
{'Country': 'Bolivia', 'Other names': 'Bolivia (Plurinational State of)', 'ISO 3166-1 alpha-3 CODE': 'BOL', 'Population': '11951714', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '902448', 'Total Deaths': '21896', 'Tot\x00Cases//1M pop': '75508', 'Tot\x00Deaths/1M pop': '1832', 'Death percentage': '2.426289382'}
{'Country': 'Bosnia and Herzegovina', 'Other names': 'Bosnia and Herzegovina', 'ISO 3166-1 alpha-3 CODE': 'BIH', 'Population': '3245097', 'Continent': 'Europe', 'Total Cases': '375693', 'Total Deaths': '15719', 'Tot\x00Cases//1M pop': '115773', 'Tot\x00Deaths/1M pop': '4844', 'Death percentage': '4.184001299'}
{'Country': 'Botswana', 'Other names': 'Botswana', 'ISO 3166-1 alpha-3 CODE': 'BWA', 'Population': '2434708', 'Continent': 'Africa', 'Total Cases': '305526', 'Total Deaths': '2686', 'Tot\x00Cases//1M pop': '125488', 'Tot\x00Deaths/1M pop': '1103', 'Death percentage': '0.879139582'}
{'Country': 'Brazil', 'Other names': 'Brazil', 'ISO 3166-1 alpha-3 CODE': 'BRA',
```

'Population': '215204501', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '29999816', 'Total Deaths': '660269', 'Tot\xxa0Cases//1M pop': '139401', 'Tot\xxa0Deaths/1M pop': '3068', 'Death percentage': '2.200910166'}  
{'Country': 'British Virgin Islands', 'Other names': 'British Virgin Islands', 'ISO 3166-1 alpha-3 CODE': 'VGB', 'Population': '30583', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '6155', 'Total Deaths': '62', 'Tot\xxa0Cases//1M pop': '201256', 'Tot\xxa0Deaths/1M pop': '2027', 'Death percentage': '1.007311129'}  
{'Country': 'Brunei ', 'Other names': 'Brunei Darussalam', 'ISO 3166-1 alpha-3 CODE': 'BRN', 'Population': '444812', 'Continent': 'Asia', 'Total Cases': '135974', 'Total Deaths': '213', 'Tot\xxa0Cases//1M pop': '305689', 'Tot\xxa0Deaths/1M pop': '479', 'Death percentage': '0.156647594'}  
{'Country': 'Bulgaria', 'Other names': 'Bulgaria', 'ISO 3166-1 alpha-3 CODE': 'BGR', 'Population': '6856886', 'Continent': 'Europe', 'Total Cases': '1140679', 'Total Deaths': '36568', 'Tot\xxa0Cases//1M pop': '166355', 'Tot\xxa0Deaths/1M pop': '5333', 'Death percentage': '3.205809873'}  
{'Country': 'Burkina Faso', 'Other names': 'Burkina Faso', 'ISO 3166-1 alpha-3 CODE': 'BFA', 'Population': '21905848', 'Continent': 'Africa', 'Total Cases': '20853', 'Total Deaths': '382', 'Tot\xxa0Cases//1M pop': '952', 'Tot\xxa0Deaths/1M pop': '17', 'Death percentage': '1.831870714'}  
{'Country': 'Burundi', 'Other names': 'Burundi', 'ISO 3166-1 alpha-3 CODE': 'BDI', 'Population': '12510155', 'Continent': 'Africa', 'Total Cases': '38519', 'Total Deaths': '38', 'Tot\xxa0Cases//1M pop': '3079', 'Tot\xxa0Deaths/1M pop': '3', 'Death percentage': '0.098652613'}  
{'Country': 'Cabo Verde', 'Other names': 'Cabo Verde', 'ISO 3166-1 alpha-3 CODE': 'CPV', 'Population': '566557', 'Continent': 'Africa', 'Total Cases': '55960', 'Total Deaths': '401', 'Tot\xxa0Cases//1M pop': '98772', 'Tot\xxa0Deaths/1M pop': '708', 'Death percentage': '0.716583274'}  
{'Country': 'Cambodia', 'Other names': 'Cambodia', 'ISO 3166-1 alpha-3 CODE': 'KHM', 'Population': '17123941', 'Continent': 'Asia', 'Total Cases': '135747', 'Total Deaths': '3054', 'Tot\xxa0Cases//1M pop': '7927', 'Tot\xxa0Deaths/1M pop': '178', 'Death percentage': '2.249773476'}  
{'Country': 'Cameroon', 'Other names': 'Cameroon', 'ISO 3166-1 alpha-3 CODE': 'CMR', 'Population': '27701805', 'Continent': 'Africa', 'Total Cases': '119544', 'Total Deaths': '1927', 'Tot\xxa0Cases//1M pop': '4315', 'Tot\xxa0Deaths/1M pop': '70', 'Death percentage': '1.611958777'}  
{'Country': 'Canada', 'Other names': 'Canada', 'ISO 3166-1 alpha-3 CODE': 'CAN', 'Population': '38321435', 'Continent': 'Northern America', 'Total Cases': '3499226', 'Total Deaths': '37690', 'Tot\xxa0Cases//1M pop': '91312', 'Tot\xxa0Deaths/1M pop': '984', 'Death percentage': '1.077095335'}  
{'Country': 'CAR', 'Other names': 'Central African Republic', 'ISO 3166-1 alpha-3 CODE': 'CAF', 'Population': '4976719', 'Continent': 'Africa', 'Total Cases': '14649', 'Total Deaths': '113', 'Tot\xxa0Cases//1M pop': '2944', 'Tot\xxa0Deaths/1M pop': '23', 'Death percentage': '0.771383712'}  
{'Country': 'Caribbean Netherlands', 'Other names': 'Bonaire, Sint Eustatius and Saba', 'ISO 3166-1 alpha-3 CODE': 'BES', 'Population': '26650', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '8574', 'Total Deaths': '33', 'Tot\xxa0Cases//1M pop': '321726', 'Tot\xxa0Deaths/1M pop': '1238', 'Death percentage': '0.384884535'}  
{'Country': 'Cayman Islands', 'Other names': 'Cayman Islands', 'ISO 3166-1 alpha-3 CODE': 'CYM', 'Population': '67073', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '20606', 'Total Deaths': '24', 'Tot\xxa0Cases//1M pop': '307218', 'Tot\xxa0Deaths/1M pop': '358', 'Death percentage': '0.116470931'}  
{'Country': 'Chad', 'Other names': 'Chad', 'ISO 3166-1 alpha-3 CODE': 'TCD', 'Population': '17250246', 'Continent': 'Africa', 'Total Cases': '7308', 'Total Deaths': '191', 'Tot\xxa0Cases//1M pop': '424', 'Tot\xxa0Deaths/1M pop': '11', 'Death percentage': '2.613574165'}  
{'Country': 'Channel Islands', 'Other names': 'Guernsey', 'ISO 3166-1 alpha-3 CODE': 'GGY', 'Population': '176668', 'Continent': 'Europe', 'Total Cases': '69036', 'Total Deaths': '156', 'Tot\xxa0Cases//1M pop': '390767', 'Tot\xxa0Deaths/1M pop': '883', 'Death percentage': '0.22596906'}  
{'Country': 'Chile', 'Other names': 'Chile', 'ISO 3166-1 alpha-3 CODE': 'CHL', 'Population': '19403451', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '3486653', 'Total Deaths': '56750', 'Tot\xxa0Cases//1M pop': '179692', 'Tot\xxa0Deaths/1M pop': '2925', 'Death percentage': '1.627635443'}  
{'Country': 'China', 'Other names': 'China', 'ISO 3166-1 alpha-3 CODE': 'CHN', 'Population': '1439323776', 'Continent': 'Asia', 'Total Cases': '154738', 'Total Deaths': '4638', 'Tot\xxa0Cases//1M pop': '108', 'Tot\xxa0Deaths/1M pop': '3', 'Death percentage': '2.99732451'}  
{'Country': 'Colombia', 'Other names': 'Colombia', 'ISO 3166-1 alpha-3 CODE': 'COL', 'Population': '51832231', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '6085926', 'Total Deaths': '139660', 'Tot\xxa0Cases//1M pop': '117416', 'Tot\xxa0Deaths/1M pop': '2694', 'Death percentage': '2.294802796'}  
{'Country': 'Comoros', 'Other names': 'Comoros', 'ISO 3166-1 alpha-3 CODE': 'COM', 'Population': '902011', 'Continent': 'Africa', 'Total Cases': '8093', 'Total Deaths': '160', 'Tot\xxa0Cases//1M pop': '8972', 'Tot\xxa0Deaths/1M pop': '177', 'Death percentage': '1.977017175'}  
{'Country': 'Congo', 'Other names': 'Congo', 'ISO 3166-1 alpha-3 CODE': 'COG', 'Population': '5755689', 'Continent': 'Africa', 'Total Cases': '24071', 'Total Deaths': '385', 'Tot\xxa0Cases//1M pop': '4182', 'Tot\xxa0Deaths/1M pop': '67', 'Death percentage': '1.599435005'}  
{'Country': 'Cook Islands', 'Other names': 'Cook Islands', 'ISO 3166-1 alpha-3 CODE': 'COK', 'Population': '17592', 'Continent': 'Oceania', 'Total Cases': '2118', 'Total Deaths': '0', 'Tot\xxa0Cases//1M pop': '120396', 'Tot\xxa0Deaths/1M pop': '0', 'Death percentage': '0'}  
{'Country': 'Costa Rica', 'Other names': 'Costa Rica', 'ISO 3166-1 alpha-3 CODE': 'CRI', 'Population': '5175547', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '839368', 'Total Deaths': '8308', 'Tot\xxa0Cases//1M pop': '162180',

```
'Tot\x{a0}Deaths/1M pop': '1605', 'Death percentage': '0.98979232'}
{'Country': 'Croatia', 'Other names': 'Croatia', 'ISO 3166-1 alpha-3 CODE': 'HRV',
'Population': '4060951', 'Continent': 'Europe', 'Total Cases': '1102730', 'Total
Deaths': '15601', 'Tot\x{a0}Cases//1M pop': '271545', 'Tot\x{a0}Deaths/1M pop': '3842',
'Death percentage': '1.414761546'}
{'Country': 'Cuba', 'Other names': 'Cuba', 'ISO 3166-1 alpha-3 CODE': 'CUB',
'Population': '11314513', 'Continent': 'Latin America and the Caribbean', 'Total
Cases': '1092547', 'Total Deaths': '8514', 'Tot\x{a0}Cases//1M pop': '96562',
'Tot\x{a0}Deaths/1M pop': '752', 'Death percentage': '0.779279976'}
{'Country': 'Curaçao', 'Other names': 'Curaçao', 'ISO 3166-1 alpha-3 CODE': 'CUW',
'Population': '165268', 'Continent': 'Latin America and the Caribbean', 'Total
Cases': '40671', 'Total Deaths': '267', 'Tot\x{a0}Cases//1M pop': '246091',
'Tot\x{a0}Deaths/1M pop': '1616', 'Death percentage': '0.656487423'}
{'Country': 'Cyprus', 'Other names': 'Cyprus', 'ISO 3166-1 alpha-3 CODE': 'CYP',
'Population': '1222745', 'Continent': 'Asia', 'Total Cases': '439964', 'Total
Deaths': '947', 'Tot\x{a0}Cases//1M pop': '359817', 'Tot\x{a0}Deaths/1M pop': '774',
'Death percentage': '0.215244884'}
{'Country': 'Czechia', 'Other names': 'Czech Republic', 'ISO 3166-1 alpha-3 CODE':
'CZE', 'Population': '10743762', 'Continent': 'Europe', 'Total Cases': '3830631',
'Total Deaths': '39720', 'Tot\x{a0}Cases//1M pop': '356545', 'Tot\x{a0}Deaths/1M pop':
'3697', 'Death percentage': '1.036904886'}
{'Country': 'Denmark', 'Other names': 'Denmark', 'ISO 3166-1 alpha-3 CODE': 'DNK',
'Population': '5827911', 'Continent': 'Europe', 'Total Cases': '2919428', 'Total
Deaths': '5762', 'Tot\x{a0}Cases//1M pop': '500939', 'Tot\x{a0}Deaths/1M pop': '989',
'Death percentage': '0.19736743'}
{'Country': 'Djibouti', 'Other names': 'Djibouti', 'ISO 3166-1 alpha-3 CODE':
'DJI', 'Population': '1013146', 'Continent': 'Africa', 'Total Cases': '15590',
'Total Deaths': '189', 'Tot\x{a0}Cases//1M pop': '15388', 'Tot\x{a0}Deaths/1M pop':
'187', 'Death percentage': '1.212315587'}
{'Country': 'Dominica', 'Other names': 'Dominica', 'ISO 3166-1 alpha-3 CODE':
'DMA', 'Population': '72299', 'Continent': 'Latin America and the Caribbean',
'Total Cases': '11891', 'Total Deaths': '63', 'Tot\x{a0}Cases//1M pop': '164470',
'Tot\x{a0}Deaths/1M pop': '871', 'Death percentage': '0.529812463'}
{'Country': 'Dominican Republic', 'Other names': 'Dominican Republic', 'ISO 3166-1
alpha-3 CODE': 'DOM', 'Population': '11038333', 'Continent': 'Latin America and the
Caribbean', 'Total Cases': '578130', 'Total Deaths': '4375', 'Tot\x{a0}Cases//1M
pop': '52375', 'Tot\x{a0}Deaths/1M pop': '396', 'Death percentage': '0.756750212'}
{'Country': 'Democratic Republic of the Congo', 'Other names': 'Democratic Republic
of the Congo', 'ISO 3166-1 alpha-3 CODE': 'COD', 'Population': '94323344',
'Continent': 'Africa', 'Total Cases': '86748', 'Total Deaths': '1337',
'Tot\x{a0}Cases//1M pop': '920', 'Tot\x{a0}Deaths/1M pop': '14', 'Death percentage':
'1.541245908'}
{'Country': 'Ecuador', 'Other names': 'Ecuador', 'ISO 3166-1 alpha-3 CODE': 'ECU',
'Population': '18111933', 'Continent': 'Latin America and the Caribbean', 'Total
Cases': '859890', 'Total Deaths': '35421', 'Tot\x{a0}Cases//1M pop': '47476',
'Tot\x{a0}Deaths/1M pop': '1956', 'Death percentage': '4.119247811'}
{'Country': 'Egypt', 'Other names': 'Egypt', 'ISO 3166-1 alpha-3 CODE': 'EGY',
'Population': '105711844', 'Continent': 'Africa', 'Total Cases': '505264', 'Total
Deaths': '24417', 'Tot\x{a0}Cases//1M pop': '4780', 'Tot\x{a0}Deaths/1M pop': '231',
'Death percentage': '4.832523196'}
{'Country': 'El Salvador', 'Other names': 'El Salvador', 'ISO 3166-1 alpha-3 CODE':
'SLV', 'Population': '6543499', 'Continent': 'Latin America and the Caribbean',
'Total Cases': '161570', 'Total Deaths': '4120', 'Tot\x{a0}Cases//1M pop': '24692',
'Tot\x{a0}Deaths/1M pop': '630', 'Death percentage': '2.549978338'}
{'Country': 'Equatorial Guinea', 'Other names': 'Equatorial Guinea', 'ISO 3166-1
alpha-3 CODE': 'GNQ', 'Population': '1483588', 'Continent': 'Africa', 'Total
Cases': '15903', 'Total Deaths': '183', 'Tot\x{a0}Cases//1M pop': '10719',
'Tot\x{a0}Deaths/1M pop': '123', 'Death percentage': '1.150726278'}
{'Country': 'Eritrea', 'Other names': 'Eritrea', 'ISO 3166-1 alpha-3 CODE': 'ERI',
'Population': '3632329', 'Continent': 'Africa', 'Total Cases': '9728', 'Total
Deaths': '103', 'Tot\x{a0}Cases//1M pop': '2678', 'Tot\x{a0}Deaths/1M pop': '28',
'Death percentage': '1.058799342'}
{'Country': 'Estonia', 'Other names': 'Estonia', 'ISO 3166-1 alpha-3 CODE': 'EST',
'Population': '1328097', 'Continent': 'Europe', 'Total Cases': '558706', 'Total
Deaths': '2468', 'Tot\x{a0}Cases//1M pop': '420682', 'Tot\x{a0}Deaths/1M pop': '1858',
'Death percentage': '0.441735009'}
{'Country': 'Eswatini', 'Other names': 'Eswatini', 'ISO 3166-1 alpha-3 CODE':
'SWZ', 'Population': '1181191', 'Continent': 'Africa', 'Total Cases': '69851',
'Total Deaths': '1394', 'Tot\x{a0}Cases//1M pop': '59136', 'Tot\x{a0}Deaths/1M pop':
'1180', 'Death percentage': '1.995676511'}
{'Country': 'Ethiopia', 'Other names': 'Ethiopia', 'ISO 3166-1 alpha-3 CODE':
'ETH', 'Population': '119945147', 'Continent': 'Africa', 'Total Cases': '469819',
'Total Deaths': '7504', 'Tot\x{a0}Cases//1M pop': '3917', 'Tot\x{a0}Deaths/1M pop':
'63', 'Death percentage': '1.597210841'}
{'Country': 'Faeroe Islands', 'Other names': 'Faeroe Islands', 'ISO 3166-1 alpha-3
CODE': 'FRO', 'Population': '49188', 'Continent': 'Europe', 'Total Cases': '34237',
'Total Deaths': '28', 'Tot\x{a0}Cases//1M pop': '696044', 'Tot\x{a0}Deaths/1M pop':
'569', 'Death percentage': '0.081782866'}
{'Country': 'Falkland Islands', 'Other names': 'Falkland Islands (Malvinas)', 'ISO
3166-1 alpha-3 CODE': 'FLK', 'Population': '3657', 'Continent': 'Latin America and
the Caribbean', 'Total Cases': '123', 'Total Deaths': '0', 'Tot\x{a0}Cases//1M pop':
'33634', 'Tot\x{a0}Deaths/1M pop': '0', 'Death percentage': '0'}
{'Country': 'Fiji', 'Other names': 'Fiji', 'ISO 3166-1 alpha-3 CODE': 'FJI',
'Population': '907817', 'Continent': 'Oceania', 'Total Cases': '64422', 'Total
Deaths': '834', 'Tot\x{a0}Cases//1M pop': '70964', 'Tot\x{a0}Deaths/1M pop': '919',
'Death percentage': '1.294588805'}
{'Country': 'Finland', 'Other names': 'Finland', 'ISO 3166-1 alpha-3 CODE': 'FIN',
```

'Population': '5555788', 'Continent': 'Europe', 'Total Cases': '889626', 'Total Deaths': '3178', 'Tot\x00Cases//1M pop': '160126', 'Tot\x00Deaths/1M pop': '572', 'Death percentage': '0.357228768'}  
{'Country': 'France', 'Other names': 'France', 'ISO 3166-1 alpha-3 CODE': 'FRA', 'Population': '65526369', 'Continent': 'Europe', 'Total Cases': '25997852', 'Total Deaths': '142506', 'Tot\x00Cases//1M pop': '396754', 'Tot\x00Deaths/1M pop': '2175', 'Death percentage': '0.548145285'}  
{'Country': 'French Guiana', 'Other names': 'French Guiana', 'ISO 3166-1 alpha-3 CODE': 'GUF', 'Population': '312224', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '79075', 'Total Deaths': '394', 'Tot\x00Cases//1M pop': '253264', 'Tot\x00Deaths/1M pop': '1262', 'Death percentage': '0.498261144'}  
{'Country': 'French Polynesia', 'Other names': 'French Polynesia', 'ISO 3166-1 alpha-3 CODE': 'PYF', 'Population': '283751', 'Continent': 'Oceania', 'Total Cases': '72318', 'Total Deaths': '646', 'Tot\x00Cases//1M pop': '254864', 'Tot\x00Deaths/1M pop': '2277', 'Death percentage': '0.893276916'}  
{'Country': 'Gabon', 'Other names': 'Gabon', 'ISO 3166-1 alpha-3 CODE': 'GAB', 'Population': '2317612', 'Continent': 'Africa', 'Total Cases': '47586', 'Total Deaths': '303', 'Tot\x00Cases//1M pop': '20532', 'Tot\x00Deaths/1M pop': '131', 'Death percentage': '0.636741899'}  
{'Country': 'Gambia', 'Other names': 'Gambia', 'ISO 3166-1 alpha-3 CODE': 'GMB', 'Population': '2535418', 'Continent': 'Africa', 'Total Cases': '11988', 'Total Deaths': '365', 'Tot\x00Cases//1M pop': '4728', 'Tot\x00Deaths/1M pop': '144', 'Death percentage': '3.044711378'}  
{'Country': 'Georgia', 'Other names': 'Georgia', 'ISO 3166-1 alpha-3 CODE': 'GEO', 'Population': '3975762', 'Continent': 'Asia', 'Total Cases': '1649222', 'Total Deaths': '16756', 'Tot\x00Cases//1M pop': '414819', 'Tot\x00Deaths/1M pop': '4215', 'Death percentage': '1.015994208'}  
{'Country': 'Germany', 'Other names': 'Germany', 'ISO 3166-1 alpha-3 CODE': 'DEU', 'Population': '84252947', 'Continent': 'Europe', 'Total Cases': '21646375', 'Total Deaths': '130563', 'Tot\x00Cases//1M pop': '256921', 'Tot\x00Deaths/1M pop': '1550', 'Death percentage': '0.603163347'}  
{'Country': 'Ghana', 'Other names': 'Ghana', 'ISO 3166-1 alpha-3 CODE': 'GHA', 'Population': '32207812', 'Continent': 'Africa', 'Total Cases': '160971', 'Total Deaths': '1445', 'Tot\x00Cases//1M pop': '4998', 'Tot\x00Deaths/1M pop': '45', 'Death percentage': '0.897677221'}  
{'Country': 'Gibraltar', 'Other names': 'Gibraltar', 'ISO 3166-1 alpha-3 CODE': 'GIB', 'Population': '33673', 'Continent': 'Europe', 'Total Cases': '16979', 'Total Deaths': '101', 'Tot\x00Cases//1M pop': '504232', 'Tot\x00Deaths/1M pop': '2999', 'Death percentage': '0.594852465'}  
{'Country': 'Greece', 'Other names': 'Greece', 'ISO 3166-1 alpha-3 CODE': 'GRC', 'Population': '10333930', 'Continent': 'Europe', 'Total Cases': '3077711', 'Total Deaths': '27684', 'Tot\x00Cases//1M pop': '297826', 'Tot\x00Deaths/1M pop': '2679', 'Death percentage': '0.899499661'}  
{'Country': 'Greenland', 'Other names': 'Greenland', 'ISO 3166-1 alpha-3 CODE': 'GRL', 'Population': '56942', 'Continent': 'Northern America', 'Total Cases': '11971', 'Total Deaths': '21', 'Tot\x00Cases//1M pop': '210231', 'Tot\x00Deaths/1M pop': '369', 'Death percentage': '0.175423941'}  
{'Country': 'Grenada', 'Other names': 'Grenada', 'ISO 3166-1 alpha-3 CODE': 'GRD', 'Population': '113436', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '14024', 'Total Deaths': '218', 'Tot\x00Cases//1M pop': '123629', 'Tot\x00Deaths/1M pop': '1922', 'Death percentage': '1.554478038'}  
{'Country': 'Guadeloupe', 'Other names': 'Guadeloupe', 'ISO 3166-1 alpha-3 CODE': 'GLP', 'Population': '400244', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '130705', 'Total Deaths': '843', 'Tot\x00Cases//1M pop': '326563', 'Tot\x00Deaths/1M pop': '2106', 'Death percentage': '0.64496385'}  
{'Country': 'Guatemala', 'Other names': 'Guatemala', 'ISO 3166-1 alpha-3 CODE': 'GTM', 'Population': '18495493', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '830745', 'Total Deaths': '17325', 'Tot\x00Cases//1M pop': '44916', 'Tot\x00Deaths/1M pop': '937', 'Death percentage': '2.085477493'}  
{'Country': 'Guinea', 'Other names': 'Guinea', 'ISO 3166-1 alpha-3 CODE': 'GIN', 'Population': '137555881', 'Continent': 'Africa', 'Total Cases': '36459', 'Total Deaths': '440', 'Tot\x00Cases//1M pop': '2650', 'Tot\x00Deaths/1M pop': '32', 'Death percentage': '1.206835075'}  
{'Country': 'Guinea-Bissau', 'Other names': 'Guinea-Bissau', 'ISO 3166-1 alpha-3 CODE': 'GNB', 'Population': '2049374', 'Continent': 'Africa', 'Total Cases': '8151', 'Total Deaths': '170', 'Tot\x00Cases//1M pop': '3977', 'Tot\x00Deaths/1M pop': '83', 'Death percentage': '2.085633665'}  
{'Country': 'Guyana', 'Other names': 'Guyana', 'ISO 3166-1 alpha-3 CODE': 'GUY', 'Population': '793196', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '63272', 'Total Deaths': '1226', 'Tot\x00Cases//1M pop': '79768', 'Tot\x00Deaths/1M pop': '1546', 'Death percentage': '1.937666595'}  
{'Country': 'Haiti', 'Other names': 'Haiti', 'ISO 3166-1 alpha-3 CODE': 'HTI', 'Population': '11645833', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '30549', 'Total Deaths': '833', 'Tot\x00Cases//1M pop': '2623', 'Tot\x00Deaths/1M pop': '72', 'Death percentage': '2.726766834'}  
{'Country': 'Honduras', 'Other names': 'Honduras', 'ISO 3166-1 alpha-3 CODE': 'HND', 'Population': '10180299', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '421062', 'Total Deaths': '10880', 'Tot\x00Cases//1M pop': '41360', 'Tot\x00Deaths/1M pop': '1069', 'Death percentage': '2.583942507'}  
{'Country': 'Hong Kong', 'Other names': 'China, Hong Kong Special Administrative Region', 'ISO 3166-1 alpha-3 CODE': 'HKG', 'Population': '7603455', 'Continent': 'Asia', 'Total Cases': '1171422', 'Total Deaths': '8172', 'Tot\x00Cases//1M pop': '154064', 'Tot\x00Deaths/1M pop': '1075', 'Death percentage': '0.69761367'}  
{'Country': 'Hungary', 'Other names': 'Hungary', 'ISO 3166-1 alpha-3 CODE': 'HUN', 'Population': '9617409', 'Continent': 'Europe', 'Total Cases': '1854198', 'Total Deaths': '45510', 'Tot\x00Cases//1M pop': '192796', 'Tot\x00Deaths/1M pop': '4732', 'Death percentage': '2.454430433'}

```
{"Country": "Iceland", "Other names": "Iceland", "ISO 3166-1 alpha-3 CODE": "ISL", "Population": "345120", "Continent": "Europe", "Total Cases": "181830", "Total Deaths": "101", "Tot\xa0Cases//1M pop": "526860", "Tot\xa0Deaths/1M pop": "293", "Death percentage": "0.055546389"} {"Country": "India", "Other names": "India", "ISO 3166-1 alpha-3 CODE": "IND", "Population": "1403754381", "Continent": "Asia", "Total Cases": "43029044", "Total Deaths": "521388", "Tot\xa0Cases//1M pop": "30653", "Tot\xa0Deaths/1M pop": "371", "Death percentage": "1.211711792"} {"Country": "Indonesia", "Other names": "Indonesia", "ISO 3166-1 alpha-3 CODE": "IDN", "Population": "278586508", "Continent": "Asia", "Total Cases": "6019981", "Total Deaths": "155288", "Tot\xa0Cases//1M pop": "21609", "Tot\xa0Deaths/1M pop": "557", "Death percentage": "2.579543025"} {"Country": "Iran", "Other names": "United Kingdom", "ISO 3166-1 alpha-3 CODE": "IRN", "Population": "85874667", "Continent": "Asia", "Total Cases": "7167646", "Total Deaths": "140315", "Tot\xa0Cases//1M pop": "83466", "Tot\xa0Deaths/1M pop": "1634", "Death percentage": "1.95761621"} {"Country": "Iraq", "Other names": "Iraq", "ISO 3166-1 alpha-3 CODE": "IRQ", "Population": "41801625", "Continent": "Asia", "Total Cases": "2320260", "Total Deaths": "25173", "Tot\xa0Cases//1M pop": "55506", "Tot\xa0Deaths/1M pop": "602", "Death percentage": "1.084921517"} {"Country": "Ireland", "Other names": "Ireland", "ISO 3166-1 alpha-3 CODE": "IRL", "Population": "5034333", "Continent": "Europe", "Total Cases": "1471210", "Total Deaths": "6786", "Tot\xa0Cases//1M pop": "292235", "Tot\xa0Deaths/1M pop": "1348", "Death percentage": "0.461252982"} {"Country": "Isle of Man", "Other names": "Isle of Man", "ISO 3166-1 alpha-3 CODE": "IMN", "Population": "85821", "Continent": "Europe", "Total Cases": "28416", "Total Deaths": "84", "Tot\xa0Cases//1M pop": "331108", "Tot\xa0Deaths/1M pop": "979", "Death percentage": "0.295608108"} {"Country": "Israel", "Other names": "Israel", "ISO 3166-1 alpha-3 CODE": "ISR", "Population": "9326000", "Continent": "Asia", "Total Cases": "3943153", "Total Deaths": "10530", "Tot\xa0Cases//1M pop": "422813", "Tot\xa0Deaths/1M pop": "1129", "Death percentage": "0.267045179"} {"Country": "Italy", "Other names": "Italy", "ISO 3166-1 alpha-3 CODE": "ITA", "Population": "60306185", "Continent": "Europe", "Total Cases": "14846514", "Total Deaths": "159784", "Tot\xa0Cases//1M pop": "246186", "Tot\xa0Deaths/1M pop": "2650", "Death percentage": "1.076239176"} {"Country": "Ivory Coast", "Other names": "C\u00e2te d'Ivoire", "ISO 3166-1 alpha-3 CODE": "CIV", "Population": "127520953", "Continent": "Africa", "Total Cases": "81761", "Total Deaths": "796", "Tot\xa0Cases//1M pop": "2971", "Tot\xa0Deaths/1M pop": "29", "Death percentage": "0.973569306"} {"Country": "Jamaica", "Other names": "Jamaica", "ISO 3166-1 alpha-3 CODE": "JAM", "Population": "2983794", "Continent": "Latin America and the Caribbean", "Total Cases": "128811", "Total Deaths": "2893", "Tot\xa0Cases//1M pop": "43170", "Tot\xa0Deaths/1M pop": "970", "Death percentage": "2.245926202"} {"Country": "Japan", "Other names": "Japan", "ISO 3166-1 alpha-3 CODE": "JPN", "Population": "125798669", "Continent": "Asia", "Total Cases": "6653841", "Total Deaths": "28248", "Tot\xa0Cases//1M pop": "52893", "Tot\xa0Deaths/1M pop": "225", "Death percentage": "0.424536745"} {"Country": "Jordan", "Other names": "Jordan", "ISO 3166-1 alpha-3 CODE": "JOR", "Population": "10380442", "Continent": "Asia", "Total Cases": "1689314", "Total Deaths": "14003", "Tot\xa0Cases//1M pop": "162740", "Tot\xa0Deaths/1M pop": "1349", "Death percentage": "0.828916353"} {"Country": "Kazakhstan", "Other names": "Kazakhstan", "ISO 3166-1 alpha-3 CODE": "KAZ", "Population": "19169833", "Continent": "Asia", "Total Cases": "1305188", "Total Deaths": "13660", "Tot\xa0Cases//1M pop": "68086", "Tot\xa0Deaths/1M pop": "713", "Death percentage": "1.046592522"} {"Country": "Kenya", "Other names": "Kenya", "ISO 3166-1 alpha-3 CODE": "KEN", "Population": "55843563", "Continent": "Africa", "Total Cases": "323454", "Total Deaths": "5648", "Tot\xa0Cases//1M pop": "5792", "Tot\xa0Deaths/1M pop": "101", "Death percentage": "1.746152467"} {"Country": "Kiribati", "Other names": "Kiribati", "ISO 3166-1 alpha-3 CODE": "KIR", "Population": "122656", "Continent": "Oceania", "Total Cases": "3067", "Total Deaths": "13", "Tot\xa0Cases//1M pop": "25005", "Tot\xa0Deaths/1M pop": "106", "Death percentage": "0.423866971"} {"Country": "Kuwait", "Other names": "Kuwait", "ISO 3166-1 alpha-3 CODE": "KWT", "Population": "4381108", "Continent": "Asia", "Total Cases": "629525", "Total Deaths": "2554", "Tot\xa0Cases//1M pop": "143691", "Tot\xa0Deaths/1M pop": "583", "Death percentage": "0.405702712"} {"Country": "Kyrgyzstan", "Other names": "Kyrgyzstan", "ISO 3166-1 alpha-3 CODE": "KGZ", "Population": "6712569", "Continent": "Asia", "Total Cases": "200968", "Total Deaths": "2991", "Tot\xa0Cases//1M pop": "29939", "Tot\xa0Deaths/1M pop": "446", "Death percentage": "1.488296644"} {"Country": "Laos", "Other names": "Lao People's Democratic Republic", "ISO 3166-1 alpha-3 CODE": "LAO", "Population": "7460338", "Continent": "Asia", "Total Cases": "183560", "Total Deaths": "679", "Tot\xa0Cases//1M pop": "24605", "Tot\xa0Deaths/1M pop": "91", "Death percentage": "0.369906298"} {"Country": "Latvia", "Other names": "Latvia", "ISO 3166-1 alpha-3 CODE": "LVA", "Population": "1849698", "Continent": "Europe", "Total Cases": "802534", "Total Deaths": "5643", "Tot\xa0Cases//1M pop": "433873", "Tot\xa0Deaths/1M pop": "3051", "Death percentage": "0.703147779"} {"Country": "Lebanon", "Other names": "Lebanon", "ISO 3166-1 alpha-3 CODE": "LBN", "Population": "6771939", "Continent": "Asia", "Total Cases": "1092995", "Total Deaths": "10315", "Tot\xa0Cases//1M pop": "161401", "Tot\xa0Deaths/1M pop": "1523", "Death percentage": "0.943737163"} {"Country": "Lesotho", "Other names": "Lesotho", "ISO 3166-1 alpha-3 CODE": "LSO", "Population": "2171978", "Continent": "Africa", "Total Cases": "32910", "Total Deaths": "697", "Tot\xa0Cases//1M pop": "15152", "Tot\xa0Deaths/1M pop": "321", "Death percentage": "0.943737163"}
```

```
'Death percentage': '2.117897296'}
{'Country': 'Liberia', 'Other names': 'Liberia', 'ISO 3166-1 alpha-3 CODE': 'LBR',
'Population': '5265647', 'Continent': 'Africa', 'Total Cases': '7400', 'Total
Deaths': '294', 'Tot\x00Cases//1M pop': '1405', 'Tot\x00Deaths/1M pop': '56',
'Death percentage': '3.972972973'}
{'Country': 'Libya', 'Other names': 'Libya', 'ISO 3166-1 alpha-3 CODE': 'LBY',
'Population': '7034832', 'Continent': 'Africa', 'Total Cases': '501738', 'Total
Deaths': '6419', 'Tot\x00Cases//1M pop': '71322', 'Tot\x00Deaths/1M pop': '912',
'Death percentage': '1.279352969'}
{'Country': 'Liechtenstein', 'Other names': 'Liechtenstein', 'ISO 3166-1 alpha-3
CODE': 'LIE', 'Population': '38320', 'Continent': 'Europe', 'Total Cases': '16429',
'Total Deaths': '84', 'Tot\x00Cases//1M pop': '428732', 'Tot\x00Deaths/1M pop':
'2192', 'Death percentage': '0.51129101'}
{'Country': 'Lithuania', 'Other names': 'Lithuania', 'ISO 3166-1 alpha-3 CODE':
'LTU', 'Population': '2655811', 'Continent': 'Europe', 'Total Cases': '1030966',
'Total Deaths': '8907', 'Tot\x00Cases//1M pop': '388193', 'Tot\x00Deaths/1M pop':
'3354', 'Death percentage': '0.863947017'}
{'Country': 'Luxembourg', 'Other names': 'Luxembourg', 'ISO 3166-1 alpha-3 CODE':
'LUX', 'Population': '643801', 'Continent': 'Europe', 'Total Cases': '216979',
'Total Deaths': '1037', 'Tot\x00Cases//1M pop': '337028', 'Tot\x00Deaths/1M pop':
'1611', 'Death percentage': '0.477926435'}
{'Country': 'Macao', 'Other names': 'China, Macao Special Administrative Region',
'ISO 3166-1 alpha-3 CODE': 'MAC', 'Population': '664828', 'Continent': 'Asia',
'Total Cases': '82', 'Total Deaths': '0', 'Tot\x00Cases//1M pop': '123',
'Tot\x00Deaths/1M pop': '0', 'Death percentage': '0'}
{'Country': 'Madagascar', 'Other names': 'Madagascar', 'ISO 3166-1 alpha-3 CODE':
'MDG', 'Population': '28936285', 'Continent': 'Africa', 'Total Cases': '64050',
'Total Deaths': '1388', 'Tot\x00Cases//1M pop': '2213', 'Tot\x00Deaths/1M pop':
'48', 'Death percentage': '2.167056987'}
{'Country': 'Malawi', 'Other names': 'Malawi', 'ISO 3166-1 alpha-3 CODE': 'MWI',
'Population': '19994654', 'Continent': 'Africa', 'Total Cases': '85664', 'Total
Deaths': '2626', 'Tot\x00Cases//1M pop': '4284', 'Tot\x00Deaths/1M pop': '131',
'Death percentage': '3.065465073'}
{'Country': 'Malaysia', 'Other names': 'Malaysia', 'ISO 3166-1 alpha-3 CODE':
'MYS', 'Population': '33091831', 'Continent': 'Asia', 'Total Cases': '4246467',
'Total Deaths': '35099', 'Tot\x00Cases//1M pop': '128324', 'Tot\x00Deaths/1M pop':
'1061', 'Death percentage': '0.826545926'}
{'Country': 'Maldives', 'Other names': 'Maldives', 'ISO 3166-1 alpha-3 CODE':
'MDV', 'Population': '557204', 'Continent': 'Asia', 'Total Cases': '176993', 'Total
Deaths': '298', 'Tot\x00Cases//1M pop': '317645', 'Tot\x00Deaths/1M pop': '535',
'Death percentage': '0.168368241'}
{'Country': 'Mali', 'Other names': 'Mali', 'ISO 3166-1 alpha-3 CODE': 'MLI',
'Population': '21271006', 'Continent': 'Africa', 'Total Cases': '30495', 'Total
Deaths': '728', 'Tot\x00Cases//1M pop': '1434', 'Tot\x00Deaths/1M pop': '34',
'Death percentage': '2.387276603'}
{'Country': 'Malta', 'Other names': 'Malta', 'ISO 3166-1 alpha-3 CODE': 'MLT',
'Population': '443602', 'Continent': 'Europe', 'Total Cases': '81596', 'Total
Deaths': '641', 'Tot\x00Cases//1M pop': '183940', 'Tot\x00Deaths/1M pop': '1445',
'Death percentage': '0.785577724'}
{'Country': 'Marshall Islands', 'Other names': 'Marshall Islands', 'ISO 3166-1
alpha-3 CODE': 'MHL', 'Population': '59889', 'Continent': 'Oceania', 'Total Cases':
'7', 'Total Deaths': '0', 'Tot\x00Cases//1M pop': '117', 'Tot\x00Deaths/1M pop':
'0', 'Death percentage': '0'}
{'Country': 'Martinique', 'Other names': 'Martinique', 'ISO 3166-1 alpha-3 CODE':
'MTQ', 'Population': '374756', 'Continent': 'Latin America and the Caribbean',
'Total Cases': '141415', 'Total Deaths': '909', 'Tot\x00Cases//1M pop': '377352',
'Tot\x00Deaths/1M pop': '2426', 'Death percentage': '0.642788954'}
{'Country': 'Mauritania', 'Other names': 'Mauritania', 'ISO 3166-1 alpha-3 CODE':
'MRT', 'Population': '4863443', 'Continent': 'Africa', 'Total Cases': '58670',
'Total Deaths': '982', 'Tot\x00Cases//1M pop': '12063', 'Tot\x00Deaths/1M pop':
'202', 'Death percentage': '1.673768536'}
{'Country': 'Mauritius', 'Other names': 'Mauritius', 'ISO 3166-1 alpha-3 CODE':
'MUS', 'Population': '1275463', 'Continent': 'Africa', 'Total Cases': '36628',
'Total Deaths': '968', 'Tot\x00Cases//1M pop': '28717', 'Tot\x00Deaths/1M pop':
'759', 'Death percentage': '2.642786939'}
{'Country': 'Mayotte', 'Other names': 'Mayotte', 'ISO 3166-1 alpha-3 CODE':
'MYT', 'Population': '284330', 'Continent': 'Africa', 'Total Cases': '36891',
'Total Deaths': '187', 'Tot\x00Cases//1M pop': '129747', 'Tot\x00Deaths/1M pop':
'658', 'Death percentage': '0.506898702'}
{'Country': 'Mexico', 'Other names': 'Mexico', 'ISO 3166-1 alpha-3 CODE': 'MEX',
'Population': '131303955', 'Continent': 'Latin America and the Caribbean', 'Total
Cases': '5665376', 'Total Deaths': '323212', 'Tot\x00Cases//1M pop': '43147',
'Tot\x00Deaths/1M pop': '2462', 'Death percentage': '5.705040583'}
{'Country': 'Micronesia', 'Other names': 'Micronesia (Federated States of)', 'ISO
3166-1 alpha-3 CODE': 'FSM', 'Population': '117134', 'Continent': 'Oceania', 'Total
Cases': '1', 'Total Deaths': '0', 'Tot\x00Cases//1M pop': '9', 'Tot\x00Deaths/1M
pop': '0', 'Death percentage': '0'}
{'Country': 'Moldova', 'Other names': 'Republic of Moldova', 'ISO 3166-1 alpha-3
CODE': 'MDA', 'Population': '4017550', 'Continent': 'Europe', 'Total Cases':
'514199', 'Total Deaths': '11446', 'Tot\x00Cases//1M pop': '127988',
'Tot\x00Deaths/1M pop': '2849', 'Death percentage': '2.225986437'}
{'Country': 'Monaco', 'Other names': 'Monaco', 'ISO 3166-1 alpha-3 CODE': 'MCO',
'Population': '39729', 'Continent': 'Europe', 'Total Cases': '10842', 'Total
Deaths': '54', 'Tot\x00Cases//1M pop': '272899', 'Tot\x00Deaths/1M pop': '1359',
'Death percentage': '0.498063088'}
{'Country': 'Mongolia', 'Other names': 'Mongolia', 'ISO 3166-1 alpha-3 CODE':
'MNG', 'Population': '3370682', 'Continent': 'Asia', 'Total Cases': '468610',
```

'Total Deaths': '2177', 'Tot\xa0Cases//1M pop': '139025', 'Tot\xa0Deaths/1M pop': '646', 'Death percentage': '0.464565417'}

{'Country': 'Montenegro', 'Other names': '', 'ISO 3166-1 alpha-3 CODE': 'MNE', 'Population': '628205', 'Continent': 'Europe', 'Total Cases': '233326', 'Total Deaths': '2705', 'Tot\xa0Cases//1M pop': '371417', 'Tot\xa0Deaths/1M pop': '4306', 'Death percentage': '1.15932215'}

{'Country': 'Montserrat', 'Other names': 'Montserrat', 'ISO 3166-1 alpha-3 CODE': 'MSR', 'Population': '4997', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '175', 'Total Deaths': '2', 'Tot\xa0Cases//1M pop': '35021', 'Tot\xa0Deaths/1M pop': '400', 'Death percentage': '1.142857143'}

{'Country': 'Morocco', 'Other names': 'Morocco', 'ISO 3166-1 alpha-3 CODE': 'MAR', 'Population': '37676342', 'Continent': 'Africa', 'Total Cases': '1163526', 'Total Deaths': '16060', 'Tot\xa0Cases//1M pop': '30882', 'Tot\xa0Deaths/1M pop': '426', 'Death percentage': '1.380287162'}

{'Country': 'Mozambique', 'Other names': 'Mozambique', 'ISO 3166-1 alpha-3 CODE': 'MOZ', 'Population': '32787052', 'Continent': 'Africa', 'Total Cases': '225266', 'Total Deaths': '2200', 'Tot\xa0Cases//1M pop': '6871', 'Tot\xa0Deaths/1M pop': '67', 'Death percentage': '0.976623192'}

{'Country': 'Myanmar', 'Other names': 'Myanmar', 'ISO 3166-1 alpha-3 CODE': 'MMR', 'Population': '55048340', 'Continent': 'Asia', 'Total Cases': '611875', 'Total Deaths': '19433', 'Tot\xa0Cases//1M pop': '11115', 'Tot\xa0Deaths/1M pop': '353', 'Death percentage': '3.175975485'}

{'Country': 'Namibia', 'Other names': 'Namibia', 'ISO 3166-1 alpha-3 CODE': 'NAM', 'Population': '2621429', 'Continent': 'Africa', 'Total Cases': '157646', 'Total Deaths': '4019', 'Tot\xa0Cases//1M pop': '60137', 'Tot\xa0Deaths/1M pop': '1533', 'Death percentage': '2.549382794'}

{'Country': 'Nepal', 'Other names': 'Nepal', 'ISO 3166-1 alpha-3 CODE': 'NPL', 'Population': '30053867', 'Continent': 'Asia', 'Total Cases': '978475', 'Total Deaths': '11951', 'Tot\xa0Cases//1M pop': '32557', 'Tot\xa0Deaths/1M pop': '398', 'Death percentage': '1.221390429'}

{'Country': 'Netherlands', 'Other names': 'Netherlands', 'ISO 3166-1 alpha-3 CODE': 'NLD', 'Population': '17201245', 'Continent': 'Europe', 'Total Cases': '7908701', 'Total Deaths': '22016', 'Tot\xa0Cases//1M pop': '459775', 'Tot\xa0Deaths/1M pop': '1280', 'Death percentage': '0.278376942'}

{'Country': 'New Caledonia', 'Other names': 'New Caledonia', 'ISO 3166-1 alpha-3 CODE': 'NCL', 'Population': '290302', 'Continent': 'Oceania', 'Total Cases': '60294', 'Total Deaths': '311', 'Tot\xa0Cases//1M pop': '207694', 'Tot\xa0Deaths/1M pop': '1071', 'Death percentage': '0.515805884'}

{'Country': 'New Zealand', 'Other names': 'New Zealand', 'ISO 3166-1 alpha-3 CODE': 'NZL', 'Population': '5002100', 'Continent': 'Oceania', 'Total Cases': '693219', 'Total Deaths': '350', 'Tot\xa0Cases//1M pop': '138586', 'Tot\xa0Deaths/1M pop': '70', 'Death percentage': '0.050489095'}

{'Country': 'Nicaragua', 'Other names': 'Nicaragua', 'ISO 3166-1 alpha-3 CODE': 'NIC', 'Population': '6762511', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '18434', 'Total Deaths': '224', 'Tot\xa0Cases//1M pop': '2726', 'Tot\xa0Deaths/1M pop': '33', 'Death percentage': '1.215145926'}

{'Country': 'Niger', 'Other names': 'Niger', 'ISO 3166-1 alpha-3 CODE': 'NER', 'Population': '25738714', 'Continent': 'Africa', 'Total Cases': '8811', 'Total Deaths': '308', 'Tot\xa0Cases//1M pop': '342', 'Tot\xa0Deaths/1M pop': '12', 'Death percentage': '3.495630462'}

{'Country': 'Nigeria', 'Other names': 'Nigeria', 'ISO 3166-1 alpha-3 CODE': 'NGA', 'Population': '215077352', 'Continent': 'Africa', 'Total Cases': '255468', 'Total Deaths': '3142', 'Tot\xa0Cases//1M pop': '1188', 'Tot\xa0Deaths/1M pop': '15', 'Death percentage': '1.229899635'}

{'Country': 'Niue', 'Other names': 'Niue', 'ISO 3166-1 alpha-3 CODE': 'NIU', 'Population': '1645', 'Continent': 'Oceania', 'Total Cases': '7', 'Total Deaths': '0', 'Tot\xa0Cases//1M pop': '4255', 'Tot\xa0Deaths/1M pop': '0', 'Death percentage': '0'}

{'Country': 'North Macedonia', 'Other names': 'The former Yugoslav Republic of Macedonia', 'ISO 3166-1 alpha-3 CODE': 'MKD', 'Population': '2083224', 'Continent': 'Europe', 'Total Cases': '306670', 'Total Deaths': '9228', 'Tot\xa0Cases//1M pop': '147209', 'Tot\xa0Deaths/1M pop': '4430', 'Death percentage': '3.009097727'}

{'Country': 'Norway', 'Other names': 'Norway', 'ISO 3166-1 alpha-3 CODE': 'NOR', 'Population': '5495449', 'Continent': 'Europe', 'Total Cases': '1408708', 'Total Deaths': '2518', 'Tot\xa0Cases//1M pop': '256341', 'Tot\xa0Deaths/1M pop': '458', 'Death percentage': '0.178745347'}

{'Country': 'Oman', 'Other names': 'Oman', 'ISO 3166-1 alpha-3 CODE': 'OMN', 'Population': '5333815', 'Continent': 'Asia', 'Total Cases': '388468', 'Total Deaths': '4251', 'Tot\xa0Cases//1M pop': '72831', 'Tot\xa0Deaths/1M pop': '797', 'Death percentage': '1.094298629'}

{'Country': 'Pakistan', 'Other names': 'Pakistan', 'ISO 3166-1 alpha-3 CODE': 'PAK', 'Population': '228397520', 'Continent': 'Asia', 'Total Cases': '1525466', 'Total Deaths': '30361', 'Tot\xa0Cases//1M pop': '6679', 'Tot\xa0Deaths/1M pop': '133', 'Death percentage': '1.990277069'}

{'Country': 'Palau', 'Other names': 'Palau', 'ISO 3166-1 alpha-3 CODE': 'PLW', 'Population': '18245', 'Continent': 'Oceania', 'Total Cases': '4042', 'Total Deaths': '6', 'Tot\xa0Cases//1M pop': '221540', 'Tot\xa0Deaths/1M pop': '329', 'Death percentage': '0.148441366'}

{'Country': 'Palestine', 'Other names': 'State of Palestine', 'ISO 3166-1 alpha-3 CODE': 'WBG', 'Population': '5308883', 'Continent': 'Asia', 'Total Cases': '581236', 'Total Deaths': '5351', 'Tot\xa0Cases//1M pop': '109484', 'Tot\xa0Deaths/1M pop': '1008', 'Death percentage': '0.920624325'}

{'Country': 'Panama', 'Other names': 'Panama', 'ISO 3166-1 alpha-3 CODE': 'PAN', 'Population': '4433639', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '765213', 'Total Deaths': '8170', 'Tot\xa0Cases//1M pop': '172593', 'Tot\xa0Deaths/1M pop': '1843', 'Death percentage': '1.067676582'}

{'Country': 'Papua New Guinea', 'Other names': 'Papua New Guinea', 'ISO 3166-1

alpha-3 CODE': 'PNG', 'Population': '9243590', 'Continent': 'Oceania', 'Total Cases': '42203', 'Total Deaths': '640', 'Tot\xxa0Cases//1M pop': '4566', 'Tot\xxa0Deaths/1M pop': '69', 'Death percentage': '1.516479871'}  
{'Country': 'Paraguay', 'Other names': 'Paraguay', 'ISO 3166-1 alpha-3 CODE': 'PRY', 'Population': '7285892', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '648353', 'Total Deaths': '18731', 'Tot\xxa0Cases//1M pop': '88987', 'Tot\xxa0Deaths/1M pop': '2571', 'Death percentage': '2.889012621'}  
{'Country': 'Peru', 'Other names': 'Peru', 'ISO 3166-1 alpha-3 CODE': 'PER', 'Population': '33775745', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '3548559', 'Total Deaths': '212328', 'Tot\xxa0Cases//1M pop': '105062', 'Tot\xxa0Deaths/1M pop': '6286', 'Death percentage': '5.983499218'}  
{'Country': 'Philippines', 'Other names': 'Philippines', 'ISO 3166-1 alpha-3 CODE': 'PHL', 'Population': '112133868', 'Continent': 'Asia', 'Total Cases': '3679485', 'Total Deaths': '59343', 'Tot\xxa0Cases//1M pop': '32813', 'Tot\xxa0Deaths/1M pop': '529', 'Death percentage': '1.612807227'}  
{'Country': 'Poland', 'Other names': 'Poland', 'ISO 3166-1 alpha-3 CODE': 'POL', 'Population': '37774045', 'Continent': 'Europe', 'Total Cases': '5969621', 'Total Deaths': '115345', 'Tot\xxa0Cases//1M pop': '158035', 'Tot\xxa0Deaths/1M pop': '3054', 'Death percentage': '1.932199716'}  
{'Country': 'Portugal', 'Other names': 'Portugal', 'ISO 3166-1 alpha-3 CODE': 'PRT', 'Population': '10144662', 'Continent': 'Europe', 'Total Cases': '3604114', 'Total Deaths': '21693', 'Tot\xxa0Cases//1M pop': '355272', 'Tot\xxa0Deaths/1M pop': '2138', 'Death percentage': '0.601895501'}  
{'Country': 'Qatar', 'Other names': 'Qatar', 'ISO 3166-1 alpha-3 CODE': 'QAT', 'Population': '2807805', 'Continent': 'Asia', 'Total Cases': '361819', 'Total Deaths': '677', 'Tot\xxa0Cases//1M pop': '128862', 'Tot\xxa0Deaths/1M pop': '241', 'Death percentage': '0.18711013'}  
{'Country': 'Réunion', 'Other names': 'Réunion', 'ISO 3166-1 alpha-3 CODE': 'REU', 'Population': '906497', 'Continent': 'Africa', 'Total Cases': '336945', 'Total Deaths': '7091', 'Tot\xxa0Cases//1M pop': '371700', 'Tot\xxa0Deaths/1M pop': '782', 'Death percentage': '0.210420098'}  
{'Country': 'Romania', 'Other names': 'Romania', 'ISO 3166-1 alpha-3 CODE': 'ROU', 'Population': '19013049', 'Continent': 'Europe', 'Total Cases': '2860094', 'Total Deaths': '65090', 'Tot\xxa0Cases//1M pop': '150428', 'Tot\xxa0Deaths/1M pop': '3423', 'Death percentage': '2.275799327'}  
{'Country': 'Russia', 'Other names': 'Russian Federation', 'ISO 3166-1 alpha-3 CODE': 'RUS', 'Population': '146044010', 'Continent': 'Europe', 'Total Cases': '17896866', 'Total Deaths': '369708', 'Tot\xxa0Cases//1M pop': '122544', 'Tot\xxa0Deaths/1M pop': '2531', 'Death percentage': '2.065769504'}  
{'Country': 'Rwanda', 'Other names': 'Rwanda', 'ISO 3166-1 alpha-3 CODE': 'RWA', 'Population': '13513881', 'Continent': 'Africa', 'Total Cases': '129728', 'Total Deaths': '1458', 'Tot\xxa0Cases//1M pop': '9600', 'Tot\xxa0Deaths/1M pop': '108', 'Death percentage': '1.123889985'}  
{'Country': 'S. Korea', 'Other names': 'Republic of Korea', 'ISO 3166-1 alpha-3 CODE': 'KOR', 'Population': '51346429', 'Continent': 'Asia', 'Total Cases': '13874216', 'Total Deaths': '17235', 'Tot\xxa0Cases//1M pop': '270208', 'Tot\xxa0Deaths/1M pop': '336', 'Death percentage': '0.124223235'}  
{'Country': 'Saint Helena', 'Other names': 'Saint Helena', 'ISO 3166-1 alpha-3 CODE': 'SHN', 'Population': '6109', 'Continent': 'Africa', 'Total Cases': '2', 'Total Deaths': '0', 'Tot\xxa0Cases//1M pop': '327', 'Tot\xxa0Deaths/1M pop': '0', 'Death percentage': '0'}  
{'Country': 'Saint Kitts and Nevis', 'Other names': 'Saint Kitts and Nevis', 'ISO 3166-1 alpha-3 CODE': 'KNA', 'Population': '53858', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '5549', 'Total Deaths': '43', 'Tot\xxa0Cases//1M pop': '103030', 'Tot\xxa0Deaths/1M pop': '798', 'Death percentage': '0.774914399'}  
{'Country': 'Saint Lucia', 'Other names': 'Saint Lucia', 'ISO 3166-1 alpha-3 CODE': 'LCA', 'Population': '185096', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '22964', 'Total Deaths': '365', 'Tot\xxa0Cases//1M pop': '124065', 'Tot\xxa0Deaths/1M pop': '1972', 'Death percentage': '1.589444348'}  
{'Country': 'Saint Martin', 'Other names': 'Saint Martin', 'ISO 3166-1 alpha-3 CODE': 'MAF', 'Population': '39820', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '10107', 'Total Deaths': '63', 'Tot\xxa0Cases//1M pop': '253817', 'Tot\xxa0Deaths/1M pop': '1582', 'Death percentage': '0.623330365'}  
{'Country': 'Saint Pierre Miquelon', 'Other names': '\xa0Saint Pierre and Miquelon', 'ISO 3166-1 alpha-3 CODE': 'SPM', 'Population': '5744', 'Continent': 'Northern America', 'Total Cases': '1957', 'Total Deaths': '1', 'Tot\xxa0Cases//1M pop': '340703', 'Tot\xxa0Deaths/1M pop': '174', 'Death percentage': '0.05109862'}  
{'Country': 'Samoa', 'Other names': 'Samoa', 'ISO 3166-1 alpha-3 CODE': 'WSM', 'Population': '200722', 'Continent': 'Oceania', 'Total Cases': '2285', 'Total Deaths': '1', 'Tot\xxa0Cases//1M pop': '11384', 'Tot\xxa0Deaths/1M pop': '5', 'Death percentage': '0.043763676'}  
{'Country': 'San Marino', 'Other names': 'San Marino', 'ISO 3166-1 alpha-3 CODE': 'SMR', 'Population': '34056', 'Continent': 'Europe', 'Total Cases': '15181', 'Total Deaths': '113', 'Tot\xxa0Cases//1M pop': '445766', 'Tot\xxa0Deaths/1M pop': '3318', 'Death percentage': '0.744351492'}  
{'Country': 'Sao Tome and Principe', 'Other names': 'Sao Tome and Principe', 'ISO 3166-1 alpha-3 CODE': 'STP', 'Population': '226281', 'Continent': 'Africa', 'Total Cases': '5945', 'Total Deaths': '73', 'Tot\xxa0Cases//1M pop': '26273', 'Tot\xxa0Deaths/1M pop': '323', 'Death percentage': '1.227922624'}  
{'Country': 'Saudi Arabia', 'Other names': 'Saudi Arabia', 'ISO 3166-1 alpha-3 CODE': 'SAU', 'Population': '35762746', 'Continent': 'Asia', 'Total Cases': '751076', 'Total Deaths': '9048', 'Tot\xxa0Cases//1M pop': '21002', 'Tot\xxa0Deaths/1M pop': '253', 'Death percentage': '1.204671698'}  
{'Country': 'Senegal', 'Other names': 'Senegal', 'ISO 3166-1 alpha-3 CODE': 'SEN', 'Population': '17515750', 'Continent': 'Africa', 'Total Cases': '85919', 'Total Deaths': '1965', 'Tot\xxa0Cases//1M pop': '4905', 'Tot\xxa0Deaths/1M pop': '112', 'Death percentage': '2.287037791'}

{'Country': 'Serbia', 'Other names': 'Serbia', 'ISO 3166-1 alpha-3 CODE': 'SRB', 'Population': '8675762', 'Continent': 'Europe', 'Total Cases': '1980722', 'Total Deaths': '15825', 'Tot\xa0Cases//1M pop': '228305', 'Tot\xa0Deaths/1M pop': '1824', 'Death percentage': '0.79895109'}  
{'Country': 'Seychelles', 'Other names': 'Seychelles', 'ISO 3166-1 alpha-3 CODE': 'SYC', 'Population': '99413', 'Continent': 'Africa', 'Total Cases': '40421', 'Total Deaths': '164', 'Tot\xa0Cases//1M pop': '406597', 'Tot\xa0Deaths/1M pop': '1650', 'Death percentage': '0.405729695'}  
{'Country': 'Sierra Leone', 'Other names': 'Sierra Leone', 'ISO 3166-1 alpha-3 CODE': 'SLE', 'Population': '8260822', 'Continent': 'Africa', 'Total Cases': '7674', 'Total Deaths': '125', 'Tot\xa0Cases//1M pop': '929', 'Tot\xa0Deaths/1M pop': '151', 'Death percentage': '1.628876727'}  
{'Country': 'Singapore', 'Other names': 'Singapore', 'ISO 3166-1 alpha-3 CODE': 'SGP', 'Population': '5930887', 'Continent': 'Asia', 'Total Cases': '1109744', 'Total Deaths': '1276', 'Tot\xa0Cases//1M pop': '187113', 'Tot\xa0Deaths/1M pop': '215', 'Death percentage': '0.114981473'}  
{'Country': 'Sint Maarten', 'Other names': 'Sint Maarten', 'ISO 3166-1 alpha-3 CODE': 'SXM', 'Population': '43728', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '9766', 'Total Deaths': '86', 'Tot\xa0Cases//1M pop': '223335', 'Tot\xa0Deaths/1M pop': '1967', 'Death percentage': '0.8800606185'}  
{'Country': 'Slovakia', 'Other names': 'Slovakia', 'ISO 3166-1 alpha-3 CODE': 'SVK', 'Population': '5464272', 'Continent': 'Europe', 'Total Cases': '1725487', 'Total Deaths': '19417', 'Tot\xa0Cases//1M pop': '315776', 'Tot\xa0Deaths/1M pop': '3553', 'Death percentage': '1.125305493'}  
{'Country': 'Slovenia', 'Other names': 'Slovenia', 'ISO 3166-1 alpha-3 CODE': 'SVN', 'Population': '2079438', 'Continent': 'Europe', 'Total Cases': '973892', 'Total Deaths': '6501', 'Tot\xa0Cases//1M pop': '468344', 'Tot\xa0Deaths/1M pop': '3126', 'Death percentage': '0.667527816'}  
{'Country': 'Solomon Islands', 'Other names': 'Solomon Islands', 'ISO 3166-1 alpha-3 CODE': 'SLB', 'Population': '716351', 'Continent': 'Oceania', 'Total Cases': '11470', 'Total Deaths': '133', 'Tot\xa0Cases//1M pop': '16012', 'Tot\xa0Deaths/1M pop': '186', 'Death percentage': '1.159546643'}  
{'Country': 'Somalia', 'Other names': 'Somalia', 'ISO 3166-1 alpha-3 CODE': 'SOM', 'Population': '16668781', 'Continent': 'Africa', 'Total Cases': '26400', 'Total Deaths': '1348', 'Tot\xa0Cases//1M pop': '1584', 'Tot\xa0Deaths/1M pop': '81', 'Death percentage': '5.106060606'}  
{'Country': 'South Africa', 'Other names': 'South Africa', 'ISO 3166-1 alpha-3 CODE': 'ZAF', 'Population': '60617532', 'Continent': 'Africa', 'Total Cases': '3722954', 'Total Deaths': '100050', 'Tot\xa0Cases//1M pop': '61417', 'Tot\xa0Deaths/1M pop': '1651', 'Death percentage': '2.687382116'}  
{'Country': 'South Sudan', 'Other names': 'South Sudan', 'ISO 3166-1 alpha-3 CODE': 'SSD', 'Population': '11423439', 'Continent': 'Africa', 'Total Cases': '17278', 'Total Deaths': '138', 'Tot\xa0Cases//1M pop': '1513', 'Tot\xa0Deaths/1M pop': '12', 'Death percentage': '0.798703554'}  
{'Country': 'Spain', 'Other names': 'Spain', 'ISO 3166-1 alpha-3 CODE': 'ESP', 'Population': '46786482', 'Continent': 'Europe', 'Total Cases': '11551574', 'Total Deaths': '102541', 'Tot\xa0Cases//1M pop': '246900', 'Tot\xa0Deaths/1M pop': '2192', 'Death percentage': '0.887679895'}  
{'Country': 'Sri Lanka', 'Other names': 'Sri Lanka', 'ISO 3166-1 alpha-3 CODE': 'LKA', 'Population': '21570428', 'Continent': 'Asia', 'Total Cases': '661991', 'Total Deaths': '16481', 'Tot\xa0Cases//1M pop': '30690', 'Tot\xa0Deaths/1M pop': '764', 'Death percentage': '2.489610886'}  
{'Country': 'St. Barth', 'Other names': 'Saint Barthélemy', 'ISO 3166-1 alpha-3 CODE': 'BLM', 'Population': '9930', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '4150', 'Total Deaths': '6', 'Tot\xa0Cases//1M pop': '417925', 'Tot\xa0Deaths/1M pop': '604', 'Death percentage': '0.144578313'}  
{'Country': 'St. Vincent Grenadines', 'Other names': 'Saint Vincent and the Grenadines', 'ISO 3166-1 alpha-3 CODE': 'VCT', 'Population': '111557', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '6746', 'Total Deaths': '106', 'Tot\xa0Cases//1M pop': '60471', 'Tot\xa0Deaths/1M pop': '950', 'Death percentage': '1.571301512'}  
{'Country': 'Sudan', 'Other names': 'Sudan', 'ISO 3166-1 alpha-3 CODE': 'SDN', 'Population': '45640385', 'Continent': 'Africa', 'Total Cases': '61955', 'Total Deaths': '4907', 'Tot\xa0Cases//1M pop': '1357', 'Tot\xa0Deaths/1M pop': '108', 'Death percentage': '7.920264708'}  
{'Country': 'Suriname', 'Other names': 'Suriname', 'ISO 3166-1 alpha-3 CODE': 'SUR', 'Population': '595833', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '79232', 'Total Deaths': '1325', 'Tot\xa0Cases//1M pop': '132977', 'Tot\xa0Deaths/1M pop': '2224', 'Death percentage': '1.67230412'}  
{'Country': 'Sweden', 'Other names': 'Sweden', 'ISO 3166-1 alpha-3 CODE': 'SWE', 'Population': '10209507', 'Continent': 'Europe', 'Total Cases': '2487852', 'Total Deaths': '18331', 'Tot\xa0Cases//1M pop': '243680', 'Tot\xa0Deaths/1M pop': '1795', 'Death percentage': '0.736820357'}  
{'Country': 'Switzerland', 'Other names': 'Switzerland', 'ISO 3166-1 alpha-3 CODE': 'CHE', 'Population': '8765420', 'Continent': 'Europe', 'Total Cases': '3490876', 'Total Deaths': '13715', 'Tot\xa0Cases//1M pop': '398255', 'Tot\xa0Deaths/1M pop': '1565', 'Death percentage': '0.392881328'}  
{'Country': 'Syria', 'Other names': 'Syrian Arab Republic', 'ISO 3166-1 alpha-3 CODE': 'SYR', 'Population': '18244381', 'Continent': 'Asia', 'Total Cases': '55711', 'Total Deaths': '3144', 'Tot\xa0Cases//1M pop': '3054', 'Tot\xa0Deaths/1M pop': '172', 'Death percentage': '5.64340974'}  
{'Country': 'Taiwan', 'Other names': 'China, Taiwan Province of China', 'ISO 3166-1 alpha-3 CODE': 'TWN', 'Population': '23892241', 'Continent': 'Asia', 'Total Cases': '24310', 'Total Deaths': '853', 'Tot\xa0Cases//1M pop': '1017', 'Tot\xa0Deaths/1M pop': '36', 'Death percentage': '3.508844097'}  
{'Country': 'Tajikistan', 'Other names': 'Tajikistan', 'ISO 3166-1 alpha-3 CODE': 'TJK', 'Population': '9912437', 'Continent': 'Asia', 'Total Cases': '17388', 'Total Deaths': '13715', 'Tot\xa0Cases//1M pop': '3054', 'Tot\xa0Deaths/1M pop': '1565', 'Death percentage': '0.392881328'}  
{'Country': 'Togo', 'Other names': 'Togo', 'ISO 3166-1 alpha-3 CODE': 'TGO', 'Population': '8000000', 'Continent': 'Africa', 'Total Cases': '1109744', 'Total Deaths': '6501', 'Tot\xa0Cases//1M pop': '468344', 'Tot\xa0Deaths/1M pop': '3126', 'Death percentage': '0.667527816'}  
{'Country': 'Tonga', 'Other names': 'Tonga', 'ISO 3166-1 alpha-3 CODE': 'TON', 'Population': '104000', 'Continent': 'Oceania', 'Total Cases': '11470', 'Total Deaths': '133', 'Tot\xa0Cases//1M pop': '16012', 'Tot\xa0Deaths/1M pop': '186', 'Death percentage': '1.159546643'}  
{'Country': 'Trinidad and Tobago', 'Other names': 'Trinidad and Tobago', 'ISO 3166-1 alpha-3 CODE': 'TTO', 'Population': '1150000', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '3722954', 'Total Deaths': '100050', 'Tot\xa0Cases//1M pop': '61417', 'Tot\xa0Deaths/1M pop': '1651', 'Death percentage': '2.687382116'}  
{'Country': 'Tunisia', 'Other names': 'Tunisia', 'ISO 3166-1 alpha-3 CODE': 'TUN', 'Population': '11500000', 'Continent': 'Africa', 'Total Cases': '1725487', 'Total Deaths': '19417', 'Tot\xa0Cases//1M pop': '315776', 'Tot\xa0Deaths/1M pop': '3553', 'Death percentage': '1.125305493'}  
{'Country': 'Turkey', 'Other names': 'Turkey', 'ISO 3166-1 alpha-3 CODE': 'TUR', 'Population': '83000000', 'Continent': 'Asia', 'Total Cases': '26400', 'Total Deaths': '1348', 'Tot\xa0Cases//1M pop': '1584', 'Tot\xa0Deaths/1M pop': '81', 'Death percentage': '5.106060606'}  
{'Country': 'Uganda', 'Other names': 'Uganda', 'ISO 3166-1 alpha-3 CODE': 'UGA', 'Population': '40000000', 'Continent': 'Africa', 'Total Cases': '13715', 'Total Deaths': '13715', 'Tot\xa0Cases//1M pop': '1565', 'Tot\xa0Deaths/1M pop': '1565', 'Death percentage': '1.0'}  
{'Country': 'Ukraine', 'Other names': 'Ukraine', 'ISO 3166-1 alpha-3 CODE': 'UKR', 'Population': '45000000', 'Continent': 'Europe', 'Total Cases': '1980722', 'Total Deaths': '15825', 'Tot\xa0Cases//1M pop': '228305', 'Tot\xa0Deaths/1M pop': '1824', 'Death percentage': '0.79895109'}  
{'Country': 'United Arab Emirates', 'Other names': 'United Arab Emirates', 'ISO 3166-1 alpha-3 CODE': 'ARE', 'Population': '100000000', 'Continent': 'Asia', 'Total Cases': '1109744', 'Total Deaths': '6501', 'Tot\xa0Cases//1M pop': '468344', 'Tot\xa0Deaths/1M pop': '3126', 'Death percentage': '0.667527816'}  
{'Country': 'United Kingdom', 'Other names': 'United Kingdom', 'ISO 3166-1 alpha-3 CODE': 'GBR', 'Population': '66000000', 'Continent': 'Europe', 'Total Cases': '2079438', 'Total Deaths': '2079438', 'Tot\xa0Cases//1M pop': '246900', 'Tot\xa0Deaths/1M pop': '246900', 'Death percentage': '1.0'}  
{'Country': 'United States', 'Other names': 'United States', 'ISO 3166-1 alpha-3 CODE': 'USA', 'Population': '330000000', 'Continent': 'North America', 'Total Cases': '11470', 'Total Deaths': '133', 'Tot\xa0Cases//1M pop': '16012', 'Tot\xa0Deaths/1M pop': '186', 'Death percentage': '1.159546643'}  
{'Country': 'Uruguay', 'Other names': 'Uruguay', 'ISO 3166-1 alpha-3 CODE': 'URY', 'Population': '3500000', 'Continent': 'South America', 'Total Cases': '3722954', 'Total Deaths': '100050', 'Tot\xa0Cases//1M pop': '61417', 'Tot\xa0Deaths/1M pop': '1651', 'Death percentage': '2.687382116'}  
{'Country': 'Vanuatu', 'Other names': 'Vanuatu', 'ISO 3166-1 alpha-3 CODE': 'VUT', 'Population': '300000', 'Continent': 'Oceania', 'Total Cases': '11470', 'Total Deaths': '133', 'Tot\xa0Cases//1M pop': '16012', 'Tot\xa0Deaths/1M pop': '186', 'Death percentage': '1.159546643'}  
{'Country': 'Venezuela', 'Other names': 'Venezuela', 'ISO 3166-1 alpha-3 CODE': 'VEN', 'Population': '30000000', 'Continent': 'South America', 'Total Cases': '1725487', 'Total Deaths': '19417', 'Tot\xa0Cases//1M pop': '315776', 'Tot\xa0Deaths/1M pop': '3553', 'Death percentage': '1.125305493'}  
{'Country': 'Yemen', 'Other names': 'Yemen', 'ISO 3166-1 alpha-3 CODE': 'YEM', 'Population': '30000000', 'Continent': 'Africa', 'Total Cases': '13715', 'Total Deaths': '13715', 'Tot\xa0Cases//1M pop': '1565', 'Tot\xa0Deaths/1M pop': '1565', 'Death percentage': '1.0'}  
{'Country': 'Zambia', 'Other names': 'Zambia', 'ISO 3166-1 alpha-3 CODE': 'ZMB', 'Population': '15000000', 'Continent': 'Africa', 'Total Cases': '13715', 'Total Deaths': '13715', 'Tot\xa0Cases//1M pop': '1565', 'Tot\xa0Deaths/1M pop': '1565', 'Death percentage': '1.0'}

Deaths': '124', 'Tot\x00Cases//1M pop': '1754', 'Tot\x00Deaths/1M pop': '13', 'Death percentage': '0.713135496'}  
{'Country': 'Tanzania', 'Other names': 'United Republic of Tanzania', 'ISO 3166-1 alpha-3 CODE': 'TZA', 'Population': '62710097', 'Continent': 'Africa', 'Total Cases': '33815', 'Total Deaths': '800', 'Tot\x00Cases//1M pop': '539', 'Tot\x00Deaths/1M pop': '13', 'Death percentage': '2.365813988'}  
{'Country': 'Thailand', 'Other names': 'Thailand', 'ISO 3166-1 alpha-3 CODE': 'THA', 'Population': '70106601', 'Continent': 'Asia', 'Total Cases': '3711595', 'Total Deaths': '25418', 'Tot\x00Cases//1M pop': '52942', 'Tot\x00Deaths/1M pop': '363', 'Death percentage': '0.684826874'}  
{'Country': 'Timor-Leste', 'Other names': 'Timor-Leste', 'ISO 3166-1 alpha-3 CODE': 'TLS', 'Population': '1362386', 'Continent': 'Asia', 'Total Cases': '22832', 'Total Deaths': '130', 'Tot\x00Cases//1M pop': '16759', 'Tot\x00Deaths/1M pop': '95', 'Death percentage': '0.569376314'}  
{'Country': 'Togo', 'Other names': 'Togo', 'ISO 3166-1 alpha-3 CODE': 'TGO', 'Population': '8618172', 'Continent': 'Africa', 'Total Cases': '36944', 'Total Deaths': '272', 'Tot\x00Cases//1M pop': '4287', 'Tot\x00Deaths/1M pop': '32', 'Death percentage': '0.736249459'}  
{'Country': 'Tonga', 'Other names': 'Tonga', 'ISO 3166-1 alpha-3 CODE': 'TON', 'Population': '107792', 'Continent': 'Oceania', 'Total Cases': '7127', 'Total Deaths': '9', 'Tot\x00Cases//1M pop': '66118', 'Tot\x00Deaths/1M pop': '83', 'Death percentage': '0.126280342'}  
{'Country': 'Trinidad and Tobago', 'Other names': 'Trinidad and Tobago', 'ISO 3166-1 alpha-3 CODE': 'TTO', 'Population': '1407422', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '1384251', 'Total Deaths': '3756', 'Tot\x00Cases//1M pop': '98354', 'Tot\x00Deaths/1M pop': '2669', 'Death percentage': '2.713382698'}  
{'Country': 'Tunisia', 'Other names': 'Tunisia', 'ISO 3166-1 alpha-3 CODE': 'TUN', 'Population': '12035092', 'Continent': 'Africa', 'Total Cases': '1035884', 'Total Deaths': '28323', 'Tot\x00Cases//1M pop': '86072', 'Tot\x00Deaths/1M pop': '2353', 'Death percentage': '2.734186453'}  
{'Country': 'Turkey', 'Other names': 'Turkey', 'ISO 3166-1 alpha-3 CODE': 'TUR', 'Population': '85927644', 'Continent': 'Asia', 'Total Cases': '14894731', 'Total Deaths': '98157', 'Tot\x00Cases//1M pop': '173340', 'Tot\x00Deaths/1M pop': '1142', 'Death percentage': '0.659004852'}  
{'Country': 'Turks and Caicos', 'Other names': 'Turks and Caicos Islands', 'ISO 3166-1 alpha-3 CODE': 'TCAÂ\x00', 'Population': '39634', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '5910', 'Total Deaths': '36', 'Tot\x00Cases//1M pop': '149114', 'Tot\x00Deaths/1M pop': '908', 'Death percentage': '0.609137056'}  
{'Country': 'UAE', 'Other names': 'United Arab Emirates', 'ISO 3166-1 alpha-3 CODE': 'ARE', 'Population': '10099567', 'Continent': 'Asia', 'Total Cases': '892170', 'Total Deaths': '2302', 'Tot\x00Cases//1M pop': '88337', 'Tot\x00Deaths/1M pop': '228', 'Death percentage': '0.258022574'}  
{'Country': 'Uganda', 'Other names': 'Uganda', 'ISO 3166-1 alpha-3 CODE': 'UGA', 'Population': '48267221', 'Continent': 'Africa', 'Total Cases': '163936', 'Total Deaths': '3595', 'Tot\x00Cases//1M pop': '3396', 'Tot\x00Deaths/1M pop': '74', 'Death percentage': '2.192928948'}  
{'Country': 'UK', 'Other names': 'United Kingdom of Great Britain and Northern Ireland', 'ISO 3166-1 alpha-3 CODE': 'GBR', 'Population': '68510300', 'Continent': 'Europe', 'Total Cases': '21216874', 'Total Deaths': '165570', 'Tot\x00Cases//1M pop': '309689', 'Tot\x00Deaths/1M pop': '2417', 'Death percentage': '0.780369436'}  
{'Country': 'Ukraine', 'Other names': 'Ukraine', 'ISO 3166-1 alpha-3 CODE': 'UKR', 'Population': '43273831', 'Continent': 'Europe', 'Total Cases': '4968881', 'Total Deaths': '107980', 'Tot\x00Cases//1M pop': '114824', 'Tot\x00Deaths/1M pop': '2495', 'Death percentage': '2.173125096'}  
{'Country': 'Uruguay', 'Other names': 'Uruguay', 'ISO 3166-1 alpha-3 CODE': 'URY', 'Population': '3494806', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '889513', 'Total Deaths': '7166', 'Tot\x00Cases//1M pop': '254524', 'Tot\x00Deaths/1M pop': '2050', 'Death percentage': '0.805609362'}  
{'Country': 'USA', 'Other names': 'United States of America', 'ISO 3166-1 alpha-3 CODE': 'USA', 'Population': '334400597', 'Continent': 'Northern America', 'Total Cases': '81839052', 'Total Deaths': '1008222', 'Tot\x00Cases//1M pop': '244734', 'Tot\x00Deaths/1M pop': '3015', 'Death percentage': '1.231957086'}  
{'Country': 'Uzbekistan', 'Other names': 'Uzbekistan', 'ISO 3166-1 alpha-3 CODE': 'UZB', 'Population': '34318156', 'Continent': 'Asia', 'Total Cases': '237853', 'Total Deaths': '1637', 'Tot\x00Cases//1M pop': '6931', 'Tot\x00Deaths/1M pop': '48', 'Death percentage': '0.688240216'}  
{'Country': 'Vanuatu', 'Other names': 'Vanuatu', 'ISO 3166-1 alpha-3 CODE': 'VUT', 'Population': '319701', 'Continent': 'Oceania', 'Total Cases': '4107', 'Total Deaths': '2', 'Tot\x00Cases//1M pop': '12846', 'Tot\x00Deaths/1M pop': '6', 'Death percentage': '0.048697346'}  
{'Country': 'Vatican City', 'Other names': 'Holy See', 'ISO 3166-1 alpha-3 CODE': 'VAT', 'Population': '805', 'Continent': 'Europe', 'Total Cases': '29', 'Total Deaths': '0', 'Tot\x00Cases//1M pop': '36025', 'Tot\x00Deaths/1M pop': '0', 'Death percentage': '0'}  
{'Country': 'Venezuela', 'Other names': 'Venezuela (Bolivarian Republic of)', 'ISO 3166-1 alpha-3 CODE': 'VEN', 'Population': '28294895', 'Continent': 'Latin America and the Caribbean', 'Total Cases': '520843', 'Total Deaths': '5686', 'Tot\x00Cases//1M pop': '18408', 'Tot\x00Deaths/1M pop': '201', 'Death percentage': '1.091691738'}  
{'Country': 'Vietnam', 'Other names': 'Viet Nam', 'ISO 3166-1 alpha-3 CODE': 'VNM', 'Population': '98871712', 'Continent': 'Asia', 'Total Cases': '9818328', 'Total Deaths': '42600', 'Tot\x00Cases//1M pop': '99304', 'Tot\x00Deaths/1M pop': '431', 'Death percentage': '0.433882429'}  
{'Country': 'Wallis and Futuna', 'Other names': 'Wallis and Futuna Islands', 'ISO 3166-1 alpha-3 CODE': 'WLF', 'Population': '10894', 'Continent': 'Oceania', 'Total Cases': '454', 'Total Deaths': '7', 'Tot\x00Cases//1M pop': '41674', 'Tot\x00Deaths/1M pop': '2'}

```
'Tot\x00Deaths/1M pop': '643', 'Death percentage': '1.54185022'}
{'Country': 'Western Sahara', 'Other names': 'Western Sahara', 'ISO 3166-1 alpha-3 CODE': 'ESH\x00', 'Population': '623031', 'Continent': 'Africa', 'Total Cases': '10', 'Total Deaths': '1', 'Tot\x00Cases//1M pop': '16', 'Tot\x00Deaths/1M pop': '2', 'Death percentage': '10'}
{'Country': 'Yemen', 'Other names': 'Yemen', 'ISO 3166-1 alpha-3 CODE': 'YEM', 'Population': '30975258', 'Continent': 'Asia', 'Total Cases': '11806', 'Total Deaths': '2143', 'Tot\x00Cases//1M pop': '381', 'Tot\x00Deaths/1M pop': '69', 'Death percentage': '18.15178723'}
{'Country': 'Zambia', 'Other names': 'Zambia', 'ISO 3166-1 alpha-3 CODE': 'ZMB', 'Population': '19284482', 'Continent': 'Africa', 'Total Cases': '317076', 'Total Deaths': '3967', 'Tot\x00Cases//1M pop': '16442', 'Tot\x00Deaths/1M pop': '206', 'Death percentage': '1.251119605'}
{'Country': 'Zimbabwe', 'Other names': 'Zimbabwe', 'ISO 3166-1 alpha-3 CODE': 'ZWE', 'Population': '15241601', 'Continent': 'Africa', 'Total Cases': '246525', 'Total Deaths': '5446', 'Tot\x00Cases//1M pop': '16174', 'Tot\x00Deaths/1M pop': '357', 'Death percentage': '2.209106581'}
```

```
with open(local_file1, 'r') as file:
 reader = csv.reader(file, delimiter = ';', quoting=csv.QUOTE_ALL)
 for row in reader:
 print(row)
```

```
['Country,Other names,ISO 3166-1 alpha-3 CODE,Population,Continent,Total
Cases,Total Deaths,Tot\x00C0Cases//1M pop,Tot\x00C0Deaths/1M pop,Death percentage']
['Afghanistan,Afghanistan,AFG,40462186,Asia,177827,7671,4395,190,4.313743132']
['Albania,Albania,ALB,2872296,Europe,273870,3492,95349,1216,1.275057509']
['Algeria,Algeria,DZA,45236699,Africa,265691,6874,5873,152,2.587215976']
['Andorra,Andorra,AND,77481,Europe,40024,153,516565,1975,0.382270638']
['Angola,Angola,AGO,34654212,Africa,99194,1900,2862,55,1.915438434']
['Anguilla,Anguilla,AIA,15237,Latin America and the
Caribbean,2700,9,177200,591,0.333333333']
['Antigua and Barbuda,Antigua and Barbuda,ATG,99348,Latin America and the
Caribbean,7493,135,75422,1359,1.801681569']
['Argentina,Argentina,ARG,45921761,Latin America and the
Caribbean,9041124,128065,196881,2789,1.416472111']
['Armenia,Armenia,ARM,2972939,Asia,422574,8617,142140,2898,2.039169471']
['Aruba,Aruba,ABW,107560,Latin America and the
Caribbean,34051,212,316577,1971,0.622595518']
['Australia,Australia,AUS,26017767,Oceania,4680816,6384,179908,245,0.136386476']
['Austria,Austria,AUT,9096360,Europe,3887355,15985,427353,1757,0.411205048']
['Azerbaijan,Azerbaijan,AZE,10299156,Asia,792061,9697,76905,942,1.224274393']
['Bahamas,Bahamas,BHM,399822,Latin America and the
Caribbean,33295,788,83275,1971,2.36672173']
['Bahrain,Bahrain,BHR,1804995,Asia,556241,1471,308168,815,0.264453717']
['Bangladesh,Bangladesh,BGD,167561502,Asia,1951770,29122,11648,174,1.492081546']
['Barbados,Barbados,BRB,287991,Latin America and the
Caribbean,59938,375,208125,1302,0.625646501']
['Belarus,Belarus,BLR,9443882,Europe,965322,6844,102217,725,0.708986224']
['Belgium,Belgium,BEL,11677924,Europe,3851048,30826,329772,2640,0.800457434']
['Belize,Belize,BLZ,410260,Latin America and the
Caribbean,57289,656,139641,1599,1.14507148']
['Benin,Benin,BEN,12678649,Africa,26952,163,2126,13,0.604778866']
['Bermuda,Bermuda,BMU,61875,Northern America,12564,128,203055,2069,1.018783827']
['Bhutan,Bhutan,BTN,786480,Asia,31437,12,39972,15,0.038171581']
['Bolivia,Bolivia (Plurinational State of),BOL,11951714,Latin America and the
Caribbean,902448,21896,75508,1832,2.426289382']
['Bosnia and Herzegovina,Bosnia and
Herzegovina,BIH,3245097,Europe,375693,15719,115773,4844,4.184001299']
['Botswana,Botswana,BWA,2434708,Africa,305526,2686,125488,1103,0.879139582']
['Brazil,Brazil,BRA,215204501,Latin America and the
Caribbean,29999816,660269,139401,3068,2.200910166']
['British Virgin Islands,British Virgin Islands,VGB,30583,Latin America and the
Caribbean,6155,62,201256,2027,1.007311129']
['Brunei ,Brunei Darussalam,BRN,444812,Asia,135974,213,305689,479,0.156647594']
['Bulgaria,Bulgaria,BGR,6856886,Europe,1140679,36568,166355,5333,3.205809873']
['Burkina Faso,Burkina Faso,BFA,21905848,Africa,20853,382,952,17,1.831870714']
['Burundi,Burundi,BDI,12510155,Africa,38519,38,3079,3,0.098652613']
['Cabo Verde,Cabo Verde,CPV,566557,Africa,55960,401,98772,708,0.716583274']
['Cambodia,Cambodia,KHM,17123941,Asia,135747,3054,7927,178,2.249773476']
['Cameroon,Cameroon,CMR,27701805,Africa,119544,1927,4315,70,1.611958777']
['Canada,Canada,CAN,38321435,Northern America,3499226,37690,91312,984,1.077095335']
['CAR,Central African Republic,CAF,4976719,Africa,14649,113,2944,23,0.771383712']
['Caribbean Netherlands,"Bonaire, Sint Eustatius and Saba",BES,26650,Latin America
and the Caribbean,8574,33,321726,1238,0.384884535']
['Cayman Islands,Cayman Islands,CYM,67073,Latin America and the
Caribbean,20606,24,307218,358,0.116470931']
['Chad,Chad,TCD,17250246,Africa,7308,191,424,11,2.613574165']
['Channel Islands,Guernsey,GGY,176668,Europe,69036,156,390767,883,0.22596906']
['Chile,Chile,CHL,19403451,Latin America and the
Caribbean,3486653,56750,179692,2925,1.627635443']
['China,China,CHN,1439323776,Asia,154738,4638,108,3,2.99732451']
['Colombia,Colombia,COL,51832231,Latin America and the
Caribbean,6085926,139660,117416,2694,2.294802796']
['Comoros,Comoros,COM,902011,Africa,8093,160,8972,177,1.977017175']
```

[ 'Congo, Congo, COG, 5755689, Africa, 24071, 385, 4182, 67, 1.599435005' ]  
[ 'Cook Islands, Cook Islands, COK, 17592, Oceania, 2118, 0, 120396, 0, 0' ]  
[ 'Costa Rica, Costa Rica, CRI, 5175547, Latin America and the Caribbean, 839368, 8308, 162180, 1605, 0.98979232' ]  
[ 'Croatia, Croatia, HRV, 4060951, Europe, 1102730, 15601, 271545, 3842, 1.414761546' ]  
[ 'Cuba, Cuba, CUB, 11314513, Latin America and the Caribbean, 1092547, 8514, 96562, 752, 0.779279976' ]  
[ 'Curaçao, Curaçao, CUW, 165268, Latin America and the Caribbean, 40671, 267, 246091, 1616, 0.656487423' ]  
[ 'Cyprus, Cyprus, CYP, 1222745, Asia, 439964, 947, 359817, 774, 0.215244884' ]  
[ 'Czechia, Czech Republic, CZE, 10743762, Europe, 3830631, 39720, 356545, 3697, 1.036904886' ]  
[ 'Denmark, Denmark, DNK, 5827913, Europe, 2919428, 5762, 500939, 989, 0.19736743' ]  
[ 'Djibouti, Djibouti, DJI, 1013146, Africa, 15590, 189, 15388, 187, 1.212315587' ]  
[ 'Dominica, Dominica, DMA, 72299, Latin America and the Caribbean, 11891, 63, 164470, 871, 0.529812463' ]  
[ 'Dominican Republic, Dominican Republic, DOM, 11038333, Latin America and the Caribbean, 578130, 4375, 52375, 396, 0.756750212' ]  
[ 'Democratic Republic of the Congo, Democratic Republic of the Congo, COD, 9432344, Africa, 86748, 1337, 920, 14, 1.541245908' ]  
[ 'Ecuador, Ecuador, ECU, 18111933, Latin America and the Caribbean, 859890, 35421, 47476, 1956, 4.119247811' ]  
[ 'Egypt, Egypt, EGY, 105711844, Africa, 505264, 24417, 4780, 231, 4.832523196' ]  
[ 'El Salvador, El Salvador, SLV, 6543499, Latin America and the Caribbean, 161570, 4120, 24692, 630, 2.549978338' ]  
[ 'Equatorial Guinea, Equatorial Guinea, GNQ, 1483588, Africa, 15903, 183, 10719, 123, 1.150726278' ]  
[ 'Eritrea, Eritrea, ERI, 3632329, Africa, 9728, 103, 2678, 28, 1.058799342' ]  
[ 'Estonia, Estonia, EST, 1328097, Europe, 558706, 2468, 420682, 1858, 0.441735009' ]  
[ 'Eswatini, Eswatini, SWZ, 1181191, Africa, 69851, 1394, 59136, 1180, 1.995676511' ]  
[ 'Ethiopia, Ethiopia, ETH, 119945147, Africa, 469819, 7504, 3917, 63, 1.597210841' ]  
[ 'Faeroe Islands, Faeroe Islands, FRO, 49188, Europe, 34237, 28, 696044, 569, 0.081782866' ]  
[ 'Falkland Islands, Falkland Islands (Malvinas), FLK, 3657, Latin America and the Caribbean, 123, 0, 33634, 0, 0' ]  
[ 'Fiji, Fiji, FJI, 907817, Oceania, 64422, 834, 70964, 919, 1.294588805' ]  
[ 'Finland, Finland, FIN, 5555788, Europe, 889626, 3178, 160126, 572, 0.357228768' ]  
[ 'France, France, FRA, 65526369, Europe, 25997852, 142506, 396754, 2175, 0.548145285' ]  
[ 'French Guiana, French Guiana, GUF, 312224, Latin America and the Caribbean, 79075, 394, 253264, 1262, 0.498261144' ]  
[ 'French Polynesia, French Polynesia, PYF, 283751, Oceania, 72318, 646, 254864, 2277, 0.893276916' ]  
[ 'Gabon, Gabon, GAB, 2317612, Africa, 47586, 303, 20532, 131, 0.636741899' ]  
[ 'Gambia, Gambia, GMB, 2535418, Africa, 11988, 365, 4728, 144, 3.044711378' ]  
[ 'Georgia, Georgia, GEO, 3975762, Asia, 1649222, 16756, 414819, 4215, 1.015994208' ]  
[ 'Germany, Germany, DEU, 84252947, Europe, 21646375, 130563, 256921, 1550, 0.603163347' ]  
[ 'Ghana, Ghana, GHA, 32207812, Africa, 160971, 1445, 4998, 45, 0.897677221' ]  
[ 'Gibraltar, Gibraltar, GIB, 33673, Europe, 16979, 101, 504232, 2999, 0.594852465' ]  
[ 'Greece, Greece, GRC, 10333930, Europe, 3077711, 27684, 297826, 2679, 0.899499661' ]  
[ 'Greenland, Greenland, GRL, 56942, Northern America, 11971, 21, 210231, 369, 0.175423941' ]  
[ 'Grenada, Grenada, GRD, 113436, Latin America and the Caribbean, 14024, 218, 123629, 1922, 1.554478038' ]  
[ 'Guadeloupe, Guadeloupe, GLP, 400244, Latin America and the Caribbean, 130705, 843, 326563, 2106, 0.64496385' ]  
[ 'Guatemala, Guatemala, GTM, 18495493, Latin America and the Caribbean, 830745, 17325, 44916, 937, 2.085477493' ]  
[ 'Guinea, Guinea, GIN, 13755881, Africa, 36459, 440, 2650, 32, 1.206835075' ]  
[ 'Guinea-Bissau, Guinea-Bissau, GNB, 2049374, Africa, 8151, 170, 3977, 83, 2.085633665' ]  
[ 'Guyana, Guyana, GUY, 793196, Latin America and the Caribbean, 63272, 1226, 79768, 1546, 1.93766595' ]  
[ 'Haiti, Haiti, HTI, 11645833, Latin America and the Caribbean, 30549, 833, 2623, 72, 2.726766834' ]  
[ 'Honduras, Honduras, HND, 10180299, Latin America and the Caribbean, 421062, 10880, 41360, 1069, 2.583942507' ]  
[ 'Hong Kong, "China, Hong Kong Special Administrative Region", HKG, 7603455, Asia, 1171422, 8172, 154064, 1075, 0.69761367' ]  
[ 'Hungary, Hungary, HUN, 9617409, Europe, 1854198, 45510, 192796, 4732, 2.454430433' ]  
[ 'Iceland, Iceland, ISL, 345120, Europe, 181830, 101, 526860, 293, 0.055546389' ]  
[ 'India, India, IND, 1403754381, Asia, 43029044, 521388, 30653, 371, 1.211711792' ]  
[ 'Indonesia, Indonesia, IDN, 278586508, Asia, 6019981, 155288, 21609, 557, 2.579543025' ]  
[ 'Iran, United Kingdom, IRN, 85874667, Asia, 7167646, 140315, 83466, 1634, 1.95761621' ]  
[ 'Iraq, Iraq, IRQ, 41801625, Asia, 2320260, 25173, 55506, 602, 1.084921517' ]  
[ 'Ireland, Ireland, IRL, 5034333, Europe, 1471210, 6786, 292235, 1348, 0.461252982' ]  
[ 'Isle of Man, Isle of Man, IMN, 85821, Europe, 28416, 84, 331108, 979, 0.295608108' ]  
[ 'Israel, Israel, ISR, 9326000, Asia, 3943153, 10530, 422813, 1129, 0.267045179' ]  
[ 'Italy, Italy, ITA, 60306185, Europe, 14846514, 159784, 246186, 2650, 1.076239176' ]  
[ "Ivory Coast, Côte d'Ivoire, CIV, 27520953, Africa, 81761, 796, 2971, 29, 0.973569306" ]  
[ 'Jamaica, Jamaica, JAM, 2983794, Latin America and the Caribbean, 128811, 2893, 43170, 970, 2.245926202' ]  
[ 'Japan, Japan, JPN, 125798669, Asia, 6653841, 28248, 52893, 225, 0.424536745' ]  
[ 'Jordan, Jordan, JOR, 10380442, Asia, 1689314, 14003, 162740, 1349, 0.828916353' ]  
[ 'Kazakhstan, Kazakhstan, KAZ, 19169833, Asia, 1305188, 13660, 68086, 713, 1.046592522' ]  
[ 'Kenya, Kenya, KEN, 55843563, Africa, 323454, 5648, 5792, 101, 1.746152467' ]  
[ 'Kiribati, Kiribati, KIR, 122656, Oceania, 3067, 13, 25005, 106, 0.423866971' ]  
[ 'Kuwait, Kuwait, KWT, 4381108, Asia, 629525, 2554, 143691, 583, 0.405702712' ]  
[ 'Kyrgyzstan, Kyrgyzstan, KGZ, 6712569, Asia, 200968, 2991, 29939, 446, 1.488296644' ]  
[ "Laos, Lao People's Democratic Republic, LAO, 7460338, Asia, 183560, 679, 24605, 91, 0.369906298" ]

[ 'Latvia, Latvia, LVA, 1849698, Europe, 802534, 5643, 433873, 3051, 0.703147779' ]  
[ 'Lebanon, Lebanon, LBN, 6771939, Asia, 1092995, 10315, 161401, 1523, 0.943737163' ]  
[ 'Lesotho, Lesotho, LSO, 2171978, Africa, 32910, 697, 15152, 321, 2.117897298' ]  
[ 'Liberia, Liberia, LBR, 5265647, Africa, 7400, 294, 1405, 56, 3.972972973' ]  
[ 'Libya, Libya, LBY, 7034832, Africa, 501738, 6419, 71322, 912, 1.279352969' ]  
[ 'Liechtenstein, Liechtenstein, LIE, 38320, Europe, 16429, 84, 428732, 2192, 0.51129101' ]  
[ 'Lithuania, Lithuania, LTU, 2655811, Europe, 1030966, 8907, 388193, 3354, 0.863947017' ]  
[ 'Luxembourg, Luxembourg, LUX, 643801, Europe, 216979, 1037, 337028, 1611, 0.477926435' ]  
[ 'Macao, "China, Macao Special Administrative Region", MAC, 664828, Asia, 82, 0, 123, 0, 0' ]  
[ 'Madagascar, Madagascar, MDG, 28936285, Africa, 64050, 1388, 2213, 48, 2.167056987' ]  
[ 'Malawi, Malawi, MWI, 19994654, Africa, 85664, 2626, 4284, 131, 3.065465073' ]  
[ 'Malaysia, Malaysia, MYS, 33091831, Asia, 4246467, 35099, 128324, 1061, 0.826545926' ]  
[ 'Maldives, Maldives, MDV, 557204, Asia, 176993, 298, 317645, 535, 0.168368241' ]  
[ 'Mali, Mali, MLI, 21271006, Africa, 30495, 728, 1434, 34, 2.387276603' ]  
[ 'Malta, Malta, MLT, 443602, Europe, 81596, 641, 183940, 1445, 0.785577724' ]  
[ 'Marshall Islands, Marshall Islands, MHL, 59889, Oceania, 7, 0, 117, 0, 0' ]  
[ 'Martinique, Martinique, MTQ, 374756, Latin America and the Caribbean, 141415, 909, 377352, 2426, 0.642788954' ]  
[ 'Mauritania, Mauritania, MRT, 4863443, Africa, 58670, 982, 12063, 202, 1.673768536' ]  
[ 'Mauritius, Mauritius, MUS, 1275463, Africa, 36628, 968, 28717, 759, 2.642786939' ]  
[ 'Mayotte, Mayotte, MYT\xa0, 284330, Africa, 36891, 187, 129747, 658, 0.506898702' ]  
[ 'Mexico, Mexico, MEX, 131303955, Latin America and the Caribbean, 5665376, 323212, 43147, 2462, 5.705040583' ]  
[ 'Micronesia, Micronesia (Federated States of), FSM, 117134, Oceania, 1, 0, 9, 0, 0' ]  
[ 'Moldova, Republic of Moldova, MDA, 4017550, Europe, 514199, 11446, 127988, 2849, 2.225986437' ]  
[ 'Monaco, Monaco, MCO, 39729, Europe, 10842, 54, 272899, 1359, 0.498063088' ]  
[ 'Mongolia, Mongolia, MNG, 3370682, Asia, 468610, 2177, 139025, 646, 0.464565417' ]  
[ 'Montenegro, , MNE, 628205, Europe, 233326, 2705, 371417, 4306, 1.15932215' ]  
[ 'Montserrat, Montserrat, MSR, 4997, Latin America and the Caribbean, 175, 2, 35021, 400, 1.142857143' ]  
[ 'Morocco, Morocco, MAR, 37676342, Africa, 1163526, 16060, 30882, 426, 1.380287162' ]  
[ 'Mozambique, Mozambique, MOZ, 32787052, Africa, 225266, 2200, 6871, 67, 0.976623192' ]  
[ 'Myanmar, Myanmar, MMR, 55048340, Asia, 611875, 19433, 11115, 353, 3.175975485' ]  
[ 'Namibia, Namibia, NAM, 2621429, Africa, 157646, 4019, 60137, 1533, 2.549382794' ]  
[ 'Nepal, Nepal, NPL, 30053867, Asia, 978475, 11951, 32557, 398, 1.221390429' ]  
[ 'Netherlands, Netherlands, NLD, 17201245, Europe, 7908701, 22016, 459775, 1280, 0.278376942' ]  
[ 'New Caledonia, New Caledonia, NCL, 290302, Oceania, 60294, 311, 207694, 1071, 0.515805884' ]  
[ 'New Zealand, New Zealand, NZL, 5002100, Oceania, 693219, 350, 138586, 70, 0.050489095' ]  
[ 'Nicaragua, Nicaragua, NIC, 6762511, Latin America and the Caribbean, 18434, 224, 2726, 33, 1.215145926' ]  
[ 'Niger, Niger, NER, 25738714, Africa, 8811, 308, 342, 12, 3.495630462' ]  
[ 'Nigeria, Nigeria, NGA, 215077352, Africa, 255468, 3142, 1188, 15, 1.229899635' ]  
[ 'Niue, Niue, NIU, 1645, Oceania, 7, 0, 4255, 0, 0' ]  
[ 'North Macedonia, The former Yugoslav Republic of Macedonia, MKD, 2083224, Europe, 306670, 9228, 147209, 4430, 3.009097727' ]  
[ 'Norway, Norway, NOR, 5495449, Europe, 1408708, 2518, 256341, 458, 0.178745347' ]  
[ 'Oman, Oman, OMN, 5333815, Asia, 388468, 4251, 72831, 797, 1.094298629' ]  
[ 'Pakistan, Pakistan, PAK, 228397520, Asia, 1525466, 30361, 6679, 133, 1.990277069' ]  
[ 'Palau, Palau, PLW, 18245, Oceania, 4042, 6, 221540, 329, 0.148441366' ]  
[ 'Palestine, State of Palestine, WBG, 5308883, Asia, 581236, 5351, 109484, 1008, 0.920624325' ]  
[ 'Panama, Panama, PAN, 4433639, Latin America and the Caribbean, 765213, 8170, 172593, 1843, 1.067676582' ]  
[ 'Papua New Guinea, Papua New Guinea, PNG, 9243590, Oceania, 42203, 640, 4566, 69, 1.516479871' ]  
[ 'Paraguay, Paraguay, PRY, 7285892, Latin America and the Caribbean, 648353, 18731, 88987, 2571, 2.889012621' ]  
[ 'Peru, Peru, PER, 33775745, Latin America and the Caribbean, 3548559, 212328, 105062, 6286, 5.983499218' ]  
[ 'Philippines, Philippines, PHL, 112133868, Asia, 3679485, 59343, 32813, 529, 1.612807227' ]  
[ 'Poland, Poland, POL, 37774045, Europe, 5969621, 115345, 158035, 3054, 1.932199716' ]  
[ 'Portugal, Portugal, PRT, 10144662, Europe, 3604114, 21693, 355272, 2138, 0.601895501' ]  
[ 'Qatar, Qatar, QAT, 2807805, Asia, 361819, 677, 128862, 241, 0.18711013' ]  
[ 'Réunion, RÃ©union, REU, 906497, Africa, 336945, 709, 371700, 782, 0.210420098' ]  
[ 'Romania, Romania, ROU, 19013049, Europe, 2860094, 65090, 150428, 3423, 2.275799327' ]  
[ 'Russia, Russian Federation, RUS, 146044010, Europe, 17896866, 369708, 122544, 2531, 2.065769504' ]  
[ 'Rwanda, Rwanda, RWA, 13513881, Africa, 129728, 1458, 9600, 108, 1.123889985' ]  
[ 'S. Korea, Republic of Korea, KOR, 51346429, Asia, 13874216, 17235, 270208, 336, 0.124223235' ]  
[ 'Saint Helena, Saint Helena, SHN, 6109, Africa, 2, 0, 327, 0, 0' ]  
[ 'Saint Kitts and Nevis, Saint Kitts and Nevis, KNA, 53858, Latin America and the Caribbean, 5549, 43, 103030, 798, 0.774914399' ]  
[ 'Saint Lucia, Saint Lucia, LCA, 185096, Latin America and the Caribbean, 22964, 365, 124065, 1972, 1.589444348' ]  
[ 'Saint Martin, Saint Martin, MAF, 39820, Latin America and the Caribbean, 10107, 63, 253817, 1582, 0.623330365' ]  
[ 'Saint Pierre Miquelon, \xa0Saint Pierre and Miquelon, SPM, 5744, Northern America, 1957, 1, 340703, 174, 0.05109862' ]  
[ 'Samoa, Samoa, WSM, 200722, Oceania, 2285, 1, 11384, 5, 0.043763676' ]  
[ 'San Marino, San Marino, SMR, 34056, Europe, 15181, 113, 445766, 3318, 0.744351492' ]  
[ 'Sao Tome and Principe, Sao Tome and Principe, STP, 226281, Africa, 5945, 73, 26273, 323, 1.227922624' ]  
[ 'Saudi Arabia, Saudi Arabia, SAU, 35762746, Asia, 751076, 9048, 21002, 253, 1.204671698' ]

```

['Senegal,Senegal,SEN,17515750,Africa,85919,1965,4905,112,2.287037791']
['Serbia,Serbia,SRB,8675762,Europe,1980722,15825,228305,1824,0.79895109']
['Seychelles,Seychelles,SYC,99413,Africa,40421,164,406597,1650,0.405729695']
['Sierra Leone,Sierra Leone,SLE,8260822,Africa,7674,125,929,15,1.628876727']
['Singapore,Singapore,SGP,5930887,Asia,1109744,1276,187113,215,0.114981473']
['Sint Maarten,Sint Maarten,SXM,43728,Latin America and the
Caribbean,9766,86,223335,1967,0.880606185']
['Slovakia,Slovakia,SVK,5464272,Europe,1725487,19417,315776,3553,1.125305493']
['Slovenia,Slovenia,SVN,2079438,Europe,973892,6501,468344,3126,0.667527816']
['Solomon Islands,Solomon
Islands,SLB,716351,Oceania,11470,133,16012,186,1.159546643']
['Somalia,Somalia,SOM,16668781,Africa,26400,1348,1584,81,5.106060606']
['South Africa,South
Africa,ZAF,60617532,Africa,3722954,100050,61417,1651,2.687382116']
['South Sudan,South Sudan,SSD,11423439,Africa,17278,138,1513,12,0.798703554']
['Spain,Spain,ESP,46786482,Europe,11551574,102541,246900,2192,0.887679895']
['Sri Lanka,Sri Lanka,LKA,21570428,Asia,661991,16481,30690,764,2.489610886']
['St. Barth ,Saint Barthélemy,BLM,9930,Latin America and the
Caribbean,4150,6,417925,604,0.144578313']
['St. Vincent Grenadines,Saint Vincent and the Grenadines,VCT,111557,Latin America
and the Caribbean,6746,106,60471,950,1.571301512']
['Sudan,Sudan,SDN,45640385,Africa,61955,4907,1357,108,7.920264708']
['Suriname,Suriname,SUR,595833,Latin America and the
Caribbean,79232,1325,132977,2224,1.67230412']
['Sweden,Sweden,SWE,10209507,Europe,2487852,18331,243680,1795,0.736820357']
['Switzerland,Switzerland,CHE,8765420,Europe,3490876,13715,398255,1565,0.392881328'
]
['Syria,Syrian Arab Republic,SYR,18244381,Asia,55711,3144,3054,172,5.64340974']
['Taiwan,"China, Taiwan Province of
China",TWN,23892241,Asia,24310,853,1017,36,3.508844097']
['Tajikistan,Tajikistan,TJK,9912437,Asia,17388,124,1754,13,0.713135496']
['Tanzania,United Republic of
Tanzania,TZA,62710097,Africa,33815,800,539,13,2.365813988']
['Thailand,Thailand,THA,70106601,Asia,3711595,25418,52942,363,0.684826874']
['Timor-Leste,Timor-Leste,TLS,1362386,Asia,22832,130,16759,95,0.569376314']
['Togo,Togo,TGO,8618172,Africa,36944,272,4287,32,0.736249459']
['Tonga,Tonga,TON,107792,Oceania,7127,9,66118,83,0.126280342']
['Trinidad and Tobago,Trinidad and Tobago,TT0,1407422,Latin America and the
Caribbean,138425,3756,98354,2669,2.713382698']
['Tunisia,Tunisia,TUN,12035092,Africa,1035884,28323,86072,2353,2.734186453']
['Turkey,Turkey,TUR,85927644,Asia,14894731,98157,173340,1142,0.659004852']
['Turks and Caicos,Turks and Caicos Islands,TCA\x0,39634,Latin America and the
Caribbean,5910,36,149114,908,0.609137056']
['UAE,United Arab Emirates,ARE,10099567,Asia,892170,2302,88337,228,0.258022574']
['Uganda,Uganda,UGA,48267221,Africa,163936,3595,3396,74,2.192928948']
['UK,United Kingdom of Great Britain and Northern
Ireland,GBR,68510300,Europe,21216874,165570,309689,2417,0.780369436']
['Ukraine,Ukraine,UKR,43273831,Europe,4968881,107980,114824,2495,2.173125096']
['Uruguay,Uruguay,URY,3494806,Latin America and the
Caribbean,889513,7166,254524,2050,0.805609362']
['USA,United States of America,USA,334400597,Northern
America,81839052,1008222,244734,3015,1.231957086']
['Uzbekistan,Uzbekistan,UZB,34318156,Asia,237853,1637,6931,48,0.688240216']
['Vanuatu,Vanuatu,VUT,319701,Oceania,4107,2,12846,6,0.048697346']
['Vatican City,Holy See,VAT,805,Europe,29,0,36025,0,0']
['Venezuela,Venezuela (Bolivarian Republic of),VEN,28294895,Latin America and the
Caribbean,520843,5686,18408,201,1.091691738']
['Vietnam,Viet Nam,VNM,98871712,Asia,9818328,42600,99304,431,0.433882429']
['Wallis and Futuna,Wallis and Futuna
Islands,WLF,10894,Oceania,454,7,41674,643,1.54185022']
['Western Sahara,Western Sahara,ESH\x0,623031,Africa,10,1,16,2,10']
['Yemen,Yemen,YEM,30975258,Asia,11806,2143,381,69,18.15178723']
['Zambia,Zambia,ZMB,19284482,Africa,317076,3967,16442,206,1.251119605']
['Zimbabwe,Zimbabwe,ZWE,15241601,Africa,246525,5446,16174,357,2.209106581']

```

## Εγγραφή σε CSV

```

with open('employee_file.csv', mode='w') as employee_file:
 employee_writer = csv.writer(employee_file, delimiter=',', quotechar='"',
 quoting=csv.QUOTE_ALL)

 employee_writer.writerow(['John Smith', 'Accounting', 'June'])
 employee_writer.writerow(['Erica Meyers', 'IT', 'March'])

```

Εγγραφή σε CSV από dictionary (λεξικό). Σε αυτή την περίπτωση πρέπει να ορίσουμε προκαταβολικά τα ονόματα των στηλών (fieldnames παράμετρος στην μέθοδο DictWriter) για να γίνει η ταυτοποίηση των κλειδιών από το λεξικό κατά την εγγραφή.

```

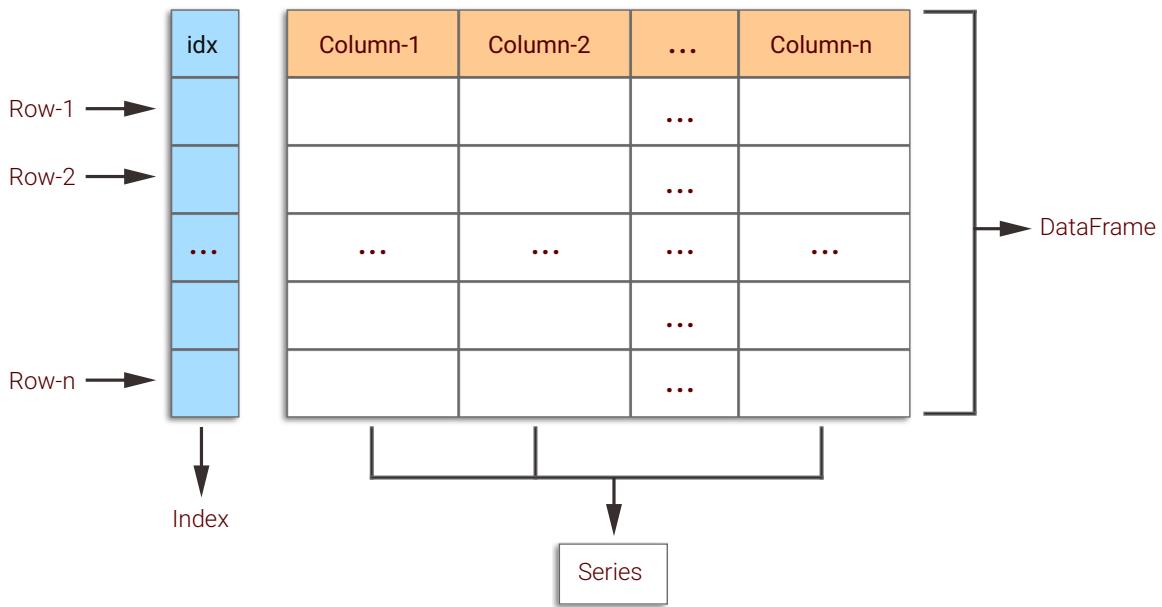
with open('employee_file2.csv', mode='w') as csv_file:
 writer = csv.DictWriter(csv_file, fieldnames= ['emp_name', 'dept',
'birth_month'],delimiter=';')

 writer.writeheader()
 writer.writerow({'emp_name': 'John Smith', 'dept': 'Accounting',
'birth_month': 'November'})
 writer.writerow({'emp_name': 'Erica Meyers', 'dept': 'IT', 'birth_month':
'March'})

```

## Η βιβλιοθήκη pandas

### Pandas Data structure



© w3resource.com

## Δημιουργία pandas Series object

```
s = pd.Series([2, 4, 6, 8, 10])
```

## Δημιουργία pandas Dataframe object. Η τελευτάι στήλη είναι από το προηγούμενο Series object.

```

df = pd.DataFrame({'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z':
[86,97,96,72,83]}, 'V':s);
df

```

	X	Y	Z	V
0	78	84	86	2
1	85	94	97	4
2	96	89	96	6
3	80	83	72	8
4	86	86	83	10

## Δημιουργία Index object

```

index = pd.Index(list('abcde'))
print(index)

```

```
Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
```

## Αλλαγή Index σε ένα datafram

```
df = df.set_index(index)
df
```

	X	Y	Z	V
a	78	84	86	2
b	85	94	97	4
c	96	89	96	6
d	80	83	72	8
e	86	86	83	10

Επιλογή στηλών από το dataframe df σε ένα νέο dataframe με το όνομα df\_subset

```
df_subset = df[['X', 'V']]
df_subset
```

	X	V
a	78	2
b	85	4
c	96	6
d	80	8
e	86	10

Ορισμός τιμών μια στήλης σε κενό (nan), με βάση κριτήρια

```
df.loc[df['Z'] >= 96, 'Z'] = np.nan
df
```

	X	Y	Z	V
a	78	84	86.0	2
b	85	94	NaN	4
c	96	89	NaN	6
d	80	83	72.0	8
e	86	86	83.0	10

Φιλτράρισμα δεδομένων

```
df_2 = df[df['V'] >= 6]
df_2
```

	X	Y	Z	V
c	96	89	NaN	6
d	80	83	72.0	8
e	86	86	83.0	10

ή αλλιώς:

```
df_2 = df.loc[df['V'] >= 8]
df_2
```

	X	Y	Z	V
d	80	83	72.0	8
e	86	86	83.0	10

με βάση συνθήκη όπου περιέχονται στην στήλη V τα στοιχεία της λίστας options

```
options=[4,10]
rslt_df = df[df['V'].isin(options)]
```

	X	Y	Z	V
b	85	94	NaN	4
e	86	86	83.0	10

Τροποποίηση κελιού

```
df.at['a', 'V'] = 1000
df
```

	X	Y	Z	V
a	78	84	86.0	1000
b	85	94	NaN	4
c	96	89	NaN	6
d	80	83	72.0	8
e	86	86	83.0	10

```
df.to_csv("leonidas.csv", sep=';', index=False)
```

Προσθήκη γραμμής

```
Νέα γραμμή σαν datafram
row_df = pd.DataFrame([pd.Series([1,2,8,99])], index = ["f"])
row_df.columns =['X', 'Y', 'Z', 'V']
row_df
```

	X	Y	Z	V
f	1	2	8	99

Εναλλακτικά φτιάχνουμε αν θελούμε την ιδια γραμμή με άλλη μέθοδο (μέσω ενός dictionary):

```
my_dictionary = {'X': [1], 'Y': [2], 'Z': [8], 'V': [99]}
row_df = pd.DataFrame.from_dict(my_dictionary)
row_df = row_df.set_index(pd.Index(['f']))
row_df
```

	X	Y	Z	V
f	1	2	8	99

```
Συννέωση
df1 = pd.concat([df, row_df], ignore_index=False)
df1
```

	X	Y	Z	V
a	78	84	86.0	1000
b	85	94	NaN	4
c	96	89	NaN	6
d	80	83	72.0	8
e	86	86	83.0	10
f	1	2	8.0	99

Αντιμετάθεση στηλών και γραμμών (transpose)

```
df2 = df1.T # transpose
df2
```

	a	b	c	d	e	f
X	78.0	85.0	96.0	80.0	86.0	1.0
Y	84.0	94.0	89.0	83.0	86.0	2.0
Z	86.0	NaN	NaN	72.0	83.0	8.0
V	1000.0	4.0	6.0	8.0	10.0	99.0

```
list(df2.columns)
```

```
['a', 'b', 'c', 'd', 'e', 'f']
```

```
list(df1.index)
```

```
['a', 'b', 'c', 'd', 'e', 'f']
```

Μπορούμε να διαβάσουμε ένα αρχείο CSV σαν pandas dataframe.

Ακολουθεί το παρακάτω παράδειγμα με δεδομένα Airbnb.

Λήψη αρχείου csv από το διαδίκτυο:

```
remote_url = 'http://data.insideairbnb.com/greece/attica/athens/2021-12-
23/visualisations/listings.csv'
Define the local filename to save data
local_file3 = 'listings.csv'
Download remote and save locally
request.urlretrieve(remote_url, local_file3)
```

```
('listings.csv', <http.client.HTTPMessage at 0x7f4b667d5420>)
```

Ανάγνωση του αρχείου σαν panda DataFrame

```
read the airbnb NYC listings csv file
airbnb = pd.read_csv(local_file3)
```

Εκτύπωση των πρώτων 10 γραμμών

```
airbnb.head()
```

	<b>id</b>	<b>name</b>	<b>host_id</b>	<b>host_name</b>	<b>neighbourhood_group</b>	<b>neighbourhood</b>	<b>lati</b>
<b>0</b>	10595	96m2, 3BR, 2BA, Metro, WI- FI etc...	37177	Emmanouil		NaN ΑΜΠΕΛΟΚΗΠΟΙ	37.9
<b>1</b>	10990	Athens Quality Apartments - Deluxe Apartment	37177	Emmanouil		NaN ΑΜΠΕΛΟΚΗΠΟΙ	37.9
<b>2</b>	10993	Athens Quality Apartments - Studio	37177	Emmanouil		NaN ΑΜΠΕΛΟΚΗΠΟΙ	37.9
<b>3</b>	10995	AQA-No2 1-bedroom, smart tv, fiber connection,	37177	Emmanouil		NaN ΑΜΠΕΛΟΚΗΠΟΙ	37.9
<b>4</b>	27262	54m2, 1-br, cable tv, wi-fi, metro	37177	Emmanouil		NaN ΑΜΠΕΛΟΚΗΠΟΙ	37.9

Εκτύπωση των 5 τελευταίων γραμμών

```
airbnb.tail(10)
```

	<b>id</b>	<b>name</b>	<b>host_id</b>	<b>host_name</b>	<b>neighbourhood_group</b>	<b>neighbo</b>
9663	53913979	Beautiful Apt. closed to Monastiraki!	35760684	Eleni	NaN	KEPAMI
9664	53914527	Cosy duplex apartment at Petralona	195674383	Selda	NaN	ΠΕΤΡΑ
9665	53914614	Classical vibes in this deluxe apt. in heart o...	35760684	Eleni	NaN	KEPAMI
9666	53915110	Family Room at the Trendy by Athens Prime Hotels	436756020	Paris	NaN	ΕΜΠΟΤΡΙΓΝΗ
9667	53916052	Modern, stylish and big apartment in Athens ce...	436763650	Styliani	NaN	ΚΩΝΣΤΑΝΤΠΛΑΤΕΙΑ Β
9668	53928165	About love AT015	428201678	Fenglin	NaN	ΚΩΝΣΤΑΝΤΠΛΑΤΕΙΑ Β
9669	53930376	Superior Queen Room by 33 Solonos suites	114156592	Bill & John	NaN	ΚΟΛΑΣ
9670	53932240	2 room sunny modern apartment, top floor + bal...	34801484	Daria	NaN	ΚΩΝΣΤΑΝΤΠΛΑΤΕΙΑ Β
9671	53934984	Διαμέρισμα στην πλατεία Αττικής, πλησίον του μ...	381802767	Hakob	NaN	ΣΤΑΥΡΟΣ
9672	53938913	A.P Athens	423806621	Anna	NaN	NEA KY

Μπορούμε να πάρουμε για όλες τις αριθμητικές στήλες περιγραφικά στατιστικά (mean, std, min, τεταρτημόρια κτλ) με την μέθοδο describe()

```
airbnb.describe()
```

	<b>id</b>	<b>host_id</b>	<b>neighbourhood_group</b>	<b>latitude</b>	<b>longitude</b>	
<b>count</b>	9.673000e+03	9.673000e+03		0.0	9673.000000	9673.000000 Ε
<b>mean</b>	3.255635e+07	1.475417e+08		NaN	37.979912	23.731762
<b>std</b>	1.366145e+07	1.188964e+08		NaN	0.013241	0.012369
<b>min</b>	1.059500e+04	3.717700e+04		NaN	37.950550	23.697700
<b>25%</b>	2.293145e+07	3.904215e+07		NaN	37.969230	23.723970
<b>50%</b>	3.335690e+07	1.304658e+08		NaN	37.978340	23.729810
<b>75%</b>	4.377216e+07	2.284115e+08		NaN	37.987960	23.737820
<b>max</b>	5.393891e+07	4.367636e+08		NaN	38.034497	23.780220 Ε

Ανάγνωση των δεδομένων μιας στήλης σαν pandas Series

```
Results for a single column
airbnb['name']
```

```
0 96m2, 3BR, 2BA, Metro, WI-FI etc...
1 Athens Quality Apartments - Deluxe Apartment
2 Athens Quality Apartments - Studio
3 AQA-No2 1-bedroom, smart tv, fiber connection,
 54m2, 1-br, cable tv, wi-fi, metro
 ...
9668 About love AT015
9669 Superior Queen Room by 33 Solonos suites
9670 2 room sunny modern apartment, top floor + bal...
9671 Διαμέρισμα στην πλατεία Αιτικής ,πλησίον του μ...
9672 A.P Athens
Name: name, Length: 9673, dtype: object
```

Επιλογή συγκεκριμένων στηλών από το dataframe

```
results for multiple columns
airbnb[['host_id', 'host_name']]
```

	host_id	host_name
0	37177	Emmanouil
1	37177	Emmanouil
2	37177	Emmanouil
3	37177	Emmanouil
4	37177	Emmanouil
...	...	...
9668	428201678	Fenglin
9669	114156592	Bill & John
9670	34801484	Daria
9671	381802767	Hakob
9672	423806621	Anna

9673 rows × 2 columns

Επιστροφή του τύπου δεδομένων για όλες τις στήλες

```
airbnb.dtypes
```

id	int64
name	object
host_id	int64
host_name	object
neighbourhood_group	float64
neighbourhood	object
latitude	float64
longitude	float64
room_type	object
price	int64
minimum_nights	int64
number_of_reviews	int64
last_review	object
reviews_per_month	float64
calculated_host_listings_count	int64
availability_365	int64
number_of_reviews_ltm	int64
license	object
dtype:	object

Μετατροπή της στήλης last\_review σε datetime64 object

```
airbnb['last_review'] = pd.to_datetime(airbnb['last_review'])
airbnb.dtypes
```

id		int64
name		object
host_id		int64
host_name		object
neighbourhood_group		float64
neighbourhood		object
latitude		float64
longitude		float64
room_type		object
price		int64
minimum_nights		int64
number_of_reviews		int64
last_review		datetime64[ns]
reviews_per_month		float64
calculated_host_listings_count		int64
availability_365		int64
number_of_reviews_ltm		int64
license		object
dtype:	object	

Μπορούμε να εξάγουμε το έτος από ένα datetime object σε μια νέα στήλη. Μετατροπή της στήλης year σε ακέραιο

```
extract the year from a datetime series
airbnb['year'] = airbnb['last_review'].dt.year

to integer
airbnb['year'] = airbnb['year'].astype('UInt16')

airbnb['year'].head()
```

0	2021
1	2021
2	2021
3	2021
4	2020

Name: year, dtype: UInt16

Το ίδιο μπορούμε να κάνουμε και με τον μήνα.

```
airbnb['month'] = airbnb['last_review'].dt.month
airbnb['month']=airbnb['month'].astype('UInt16')
airbnb.head()
```

	<b>id</b>	<b>name</b>	<b>host_id</b>	<b>host_name</b>	<b>neighbourhood_group</b>	<b>neighbourhood</b>	<b>lati</b>
<b>0</b>	10595	96m2, 3BR, 2BA, Metro, WI- FI etc...	37177	Emmanouil		NaN	ΑΜΠΕΛΟΚΗΠΟΙ 37.9
<b>1</b>	10990	Athens Quality Apartments - Deluxe Apartment	37177	Emmanouil		NaN	ΑΜΠΕΛΟΚΗΠΟΙ 37.9
<b>2</b>	10993	Athens Quality Apartments - Studio	37177	Emmanouil		NaN	ΑΜΠΕΛΟΚΗΠΟΙ 37.9
<b>3</b>	10995	AQA-No2 1-bedroom, smart tv, fiber connection,	37177	Emmanouil		NaN	ΑΜΠΕΛΟΚΗΠΟΙ 37.9
<b>4</b>	27262	54m2, 1-br, cable tv, wi-fi, metro	37177	Emmanouil		NaN	ΑΜΠΕΛΟΚΗΠΟΙ 37.9

Αν κάποια στήλη περιέχει κενά στην αρχή ή το τέλος της μπορούμε να τα αφαιρέσουμε ως εξής:

```
airbnb['name'] = airbnb['name'].str.strip()
airbnb['name'].head()
```

```
0 96m2, 3BR, 2BA, Metro, WI-FI etc...
1 Athens Quality Apartments - Deluxe Apartment
2 Athens Quality Apartments - Studio
3 AQA-No2 1-bedroom, smart tv, fiber connection,
4 54m2, 1-br, cable tv, wi-fi, metro
Name: name, dtype: object
```

Μετατροπή όλων των χαρακτήρων μιας στήλης σε πεζά:

```
airbnb['name_lower'] = airbnb['name'].str.lower()
airbnb['name_lower'].head()
```

```
0 96m2, 3br, 2ba, metro, wi-fi etc...
1 athens quality apartments - deluxe apartment
2 athens quality apartments - studio
3 aqa-no2 1-bedroom, smart tv, fiber connection,
4 54m2, 1-br, cable tv, wi-fi, metro
Name: name_lower, dtype: object
```

Δημιουργία μιας στήλης που στηρίζεται σε υπολογισμό μεταξύ άλλων στηλών (calculated column)

```
airbnb['min_revenue'] = airbnb['minimum_nights'] * airbnb['price']
airbnb[['minimum_nights', 'price', 'min_revenue']]
```

	minimum_nights	price	min_revenue
0	1	70	70
1	1	50	50
2	1	38	38
3	1	48	48
4	1	52	52
...	...	...	...
9668	2	40	80
9669	1	110	110
9670	4	33	132
9671	1	32	32
9672	1	40	40

9673 rows × 3 columns

Υπολογισμός του μέσου όρου μιας στήλης

```
airbnb['price'].mean()
```

```
86.91285020159206
```

Υπολογισμός του διαμέσου μιας στήλης

```
airbnb['price'].median()
```

```
50.0
```

Υπολογισμό μέσου όρου τιμής ανά τύπο δωματίου

```
airbnb[['room_type', 'price']].groupby('room_type', as_index=False).mean()
```

	room_type	price
0	Entire home/apt	84.221882
1	Hotel room	188.514706
2	Private room	100.025054
3	Shared room	61.857143

Υπολογισμό διαμέσου τιμής ανά τύπο δωματίου. Χρησιμοποιούμε την στήλη room\_type σαν index

```
airbnb[['room_type', 'price']].groupby('room_type', as_index=True).median()
```

room_type	price
Entire home/apt	51.0
Hotel room	105.0
Private room	38.0
Shared room	14.0

Υπολογισμό διαμέσου τιμής ανά τύπο δωματίου και έτος

```
airbnb[['room_type', 'year', 'price']].groupby(['room_type', 'year'], as_index=False).mean()
```

	room_type	year	price
0	Entire home/apt	2013	133.000000
1	Entire home/apt	2014	89.666667
2	Entire home/apt	2015	63.500000
3	Entire home/apt	2016	90.432432
4	Entire home/apt	2017	92.000000
5	Entire home/apt	2018	90.602094
6	Entire home/apt	2019	111.112040
7	Entire home/apt	2020	74.048780
8	Entire home/apt	2021	69.686602
9	Hotel room	2017	116.333333
10	Hotel room	2018	96.800000
11	Hotel room	2019	610.882353
12	Hotel room	2020	115.600000
13	Hotel room	2021	108.051282
14	Private room	2015	160.000000
15	Private room	2016	51.285714
16	Private room	2017	67.000000
17	Private room	2018	55.966667
18	Private room	2019	53.809524
19	Private room	2020	124.762712
20	Private room	2021	43.755376
21	Shared room	2017	42.000000
22	Shared room	2019	257.500000
23	Shared room	2020	391.666667
24	Shared room	2021	12.952381

Φιλτράρισμα γραμμών με βάση ένα κριτήριο (τιμή <1000) και προσάρτηση σε ένα νέο dataframe

```
get all rows with price < 1000
airbnb_under_1000 = airbnb[airbnb['price'] < 1000]
airbnb_under_1000.head()
```

	<b>id</b>	<b>name</b>	<b>host_id</b>	<b>host_name</b>	<b>neighbourhood_group</b>	<b>neighbourhood</b>	<b>lati</b>
<b>0</b>	10595	96m2, 3BR, 2BA, Metro, WI-FI etc...	37177	Emmanouil		NaN ΑΜΠΕΛΟΚΗΠΟΙ	37.9
<b>1</b>	10990	Athens Quality Apartments - Deluxe Apartment	37177	Emmanouil		NaN ΑΜΠΕΛΟΚΗΠΟΙ	37.9
<b>2</b>	10993	Athens Quality Apartments - Studio	37177	Emmanouil		NaN ΑΜΠΕΛΟΚΗΠΟΙ	37.9
<b>3</b>	10995	AQA-No2 1-bedroom, smart tv, fiber connection,	37177	Emmanouil		NaN ΑΜΠΕΛΟΚΗΠΟΙ	37.9
<b>4</b>	27262	54m2, 1-br, cable tv, wi-fi, metro	37177	Emmanouil		NaN ΑΜΠΕΛΟΚΗΠΟΙ	37.9

5 rows × 22 columns

Φιλτράρισμα με πολλαπλά κριτήρια (πινή <1000 και έτος 2020)

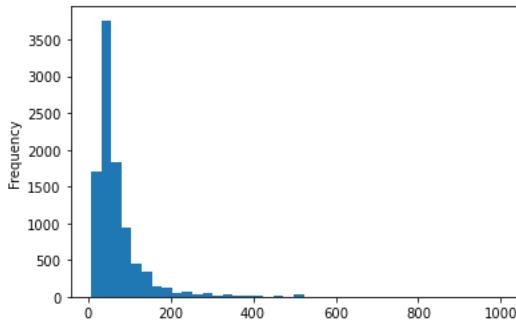
```
get all rows with price < 1000 and year equal to 2020
airbnb_2019_under_1000 = airbnb[(airbnb['price'] < 1000) & (airbnb['year'] == 2020)]
airbnb_2019_under_1000.head()
```

	<b>id</b>	<b>name</b>	<b>host_id</b>	<b>host_name</b>	<b>neighbourhood_group</b>	<b>neighbourhood</b>
<b>4</b>	27262	54m2, 1-br, cable tv, wi-fi, metro	37177	Emmanouil		NaN ΑΜΠΕΛΟΚΗΠΟΙ
<b>18</b>	138386	Apartment Dora , Service with style	675611	Yavor		NaN ΠΑΓΚΡΑΤΙ
<b>30</b>	281158	Center 1, Metro 30m, Satelite TV	640163	Nikos		NaN ΠΛΑΤΕΙΑ ΑΤΤΙΚΗΣ
<b>41</b>	469140	Our first love #01   Thiseo Residencies	2289590	Live In Athens		NaN ΘΗΣΕΙΟ
<b>42</b>	470219	Our Playground #02   Thiseo Residencies	2289590	Live In Athens		NaN ΘΗΣΕΙΟ

5 rows × 22 columns

Ιστόγραμμα για την στήλη price στον πίνακα airbnb\_under\_1000

```
ax = airbnb_under_1000['price'].plot.hist(bins=40)
```



## Βιβλιογραφία

pandas Tutorial for Beginners, <https://www.datacamp.com/tutorial/pandas>, Πρόσβαση: 18/05/2022

Reading and Writing CSV Files in Python, <https://realpython.com/python-csv/#reading-csv-files-with-csv>, Πρόσβαση: 18/05/2022

Reading CSV files in Python, <https://www.programiz.com/python-programming/reading-csv-files>, Πρόσβαση: 18/05/2022

COVID-19 Coronavirus Pandemic, <https://www.kaggle.com/datasets/rinichristy/covid19-coronavirus-pandemic>, Πρόσβαση: 18/05/2022 Pandas User Guide, [https://pandas.pydata.org/docs/user\\_guide/index.html#user-guide](https://pandas.pydata.org/docs/user_guide/index.html#user-guide), Πρόσβαση: 23/05/2022

McKinney, W., 2013. Python for data analysis. O'Reilly, Beijing.

## 9. Γεωεπεξεργασία διανυσματικών δεδομένων

Ειδική ενότητα για εκτέλεση στο Google Colab

```
έλεγχος αν το notebook τρέχει στο google colab
try:
 import google.colab
 IN_COLAB = True
except:
 IN_COLAB = False

αν το notebook τρέχει στο colab, mount το Google Drive και αλλαγή στο directory
που έχει γίνει clone το github repository.
εγκατάσταση απαραίτητων βιβλιοθηκών
if IN_COLAB:
 from google.colab import drive
 drive.mount('/content/drive')
 %cd /content/drive/MyDrive/Colab\ Notebooks/programming/notebooks
 !pip install rasterio geopandas folium matplotlib mapclassify
```

Ο χώρος και οι γεωμετρικές δομές του μπορούν να αναπαρασταθούν μέσω διανυσματικών δεδομένων (Vector). Οι τρεις βασικές δομές δεδομένων είναι:

- Τα σημεία
- Οι γραμμές
- Τα πολύγωνα

## Point (node, vertex)

INDIVIDUAL X,Y LOCATIONS

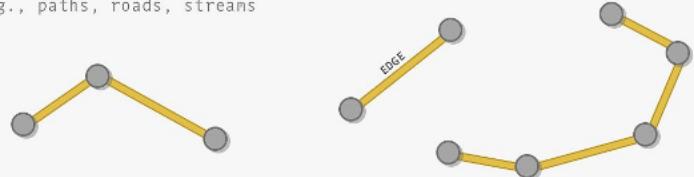
E.g., label, manhole, tower locations



## Line (chain)

2 OR MORE POINTS THAT ARE CONNECTED\*

E.g., paths, roads, streams

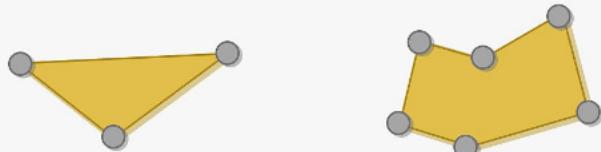


\* 2 connected points also known as edge or arc.

## Polygon (area segment)

3 OR MORE VERTICES THAT ARE CONNECTED AND CLOSED

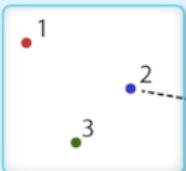
E.g., Land/water boundaries, buildings



<https://www.learndatasci.com>

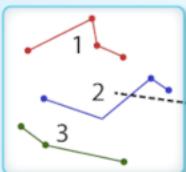
Επιπλέον αυτά τα γεωμετρικά δεδομένα συνοδεύονται και από περιγραφικά δεδομένα που αφορούν τις ιδιότητες ή τα χαρακτηριστικά αυτών των δεδομένων.

Example Attributes for Point Data



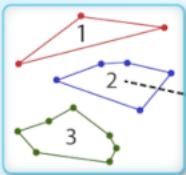
ID	Plot Size	Type	VegClass
1	40	Vegetation	Conifer
2	20	Vegetation	Deciduous
3	40	Vegetation	Conifer

Example Attributes for Line Data



ID	Type	Status	Maintenance
1	Road	Open	Year Round
2	Dirt Trail	Open	Summer
3	Road	Closed	Year Round

Example Attributes for Polygon Data

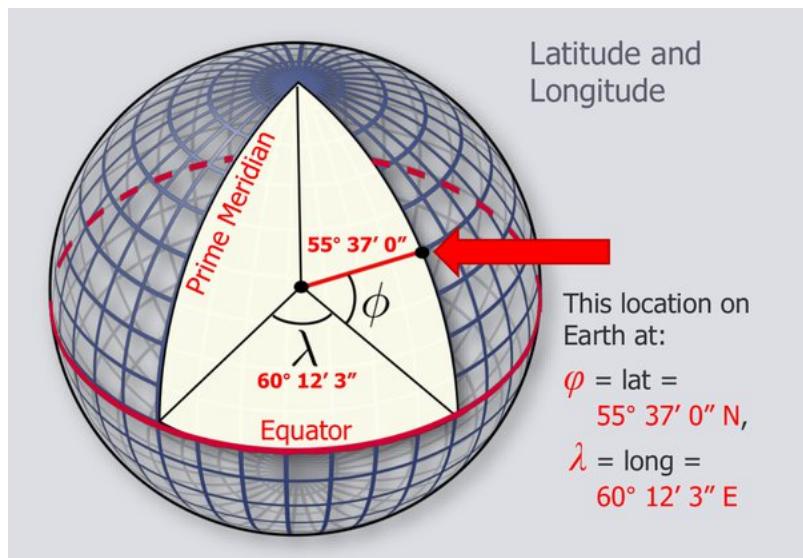


ID	Type	Class	Status
1	Herbaceous	Grassland	Protected
2	Herbaceous	Pasture	Open
3	Herbaceous / Woody	Grassland	Protected

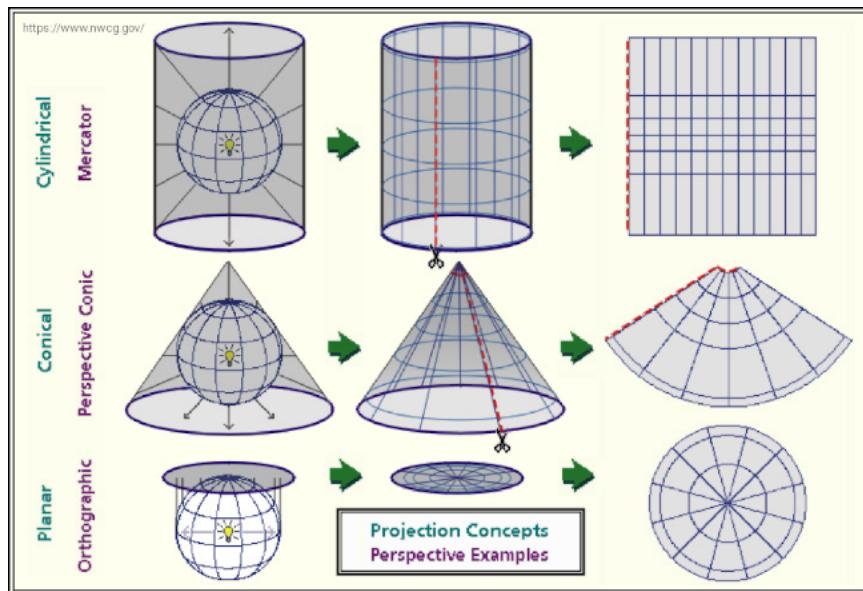
ΠΩΣ;

Στα Σ.Γ.Π. ο πιο συνηθισμένος τύπος αρχείων αποθήκευσης αυτών των δεδομένων είναι το shapefile. Πλέον έχουν αναπτυχθεί και άλλοι τύποι όπως geojson, geopackage και χωρικές βάσεις δεδομένων (geodatabases) όπως η Postgresql/Postgis.

Ο προσδιορισμός της γεωγραφικής θέσης στην υδρόγειο γίνεται μέσω ενός ζεύγους γεωγραφικών συντεταγμένων. Κάθε σημείο στον χώρο προσδιορίζεται γεωγραφικά από το γεωγραφικό μήκος ( $\lambda$ ) και το γεωγραφικό πλάτος ( $\phi$ ).



Για να αποδοθεί η τρισδιάστατη υδρόγειος σφαίρα σε ένα δυσδιάστατο σύστημα αναφορά χρησιμοποιείται ένα προβολικό σύστημα.



Κάθε προβολικό σύστημα της σφαίρας στο επίπεδο εισάγει μια σειρά παραμορφώσεων που αφορά το σχήμα των γεωμετρικών δομών, την κλίμακα, την έκταση και τις αποστάσεις. Ανάλογα το προβολικό σύστημα κάποιες από τις παραπάνω παραμορφώσεις εμφανίζονται σε μεγάλο βαθμό και άλλες όχι. Οπότε ανάλογα το είδος της έρευνας ο ερευνητής οφείλει να γνωρίζει τι είδους παραμορφώσεις εισάγει η κάθε προβολή και ανάλογα να επιλέγει την προβολή με τα λιγότερα σφάλματα.

## Python βιβλιοθήκες για διανυσματικά δεδομένα

Για την ανάγνωση, εγγραφή, επεξεργασία διανυσματικών δεδομένων στην Python έχουν καθιερωθεί μια σειρά βιβλιοθηκών. Η αρχαιότερη και βασική βιβλιοθήκη είναι η [GDAL/OGR](#). Επειδή η βιβλιοθήκη δεν είναι ιδιαίτερα συμβατή σε σχέση με τον τρόπο συγγραφής της Python και είναι επιρρεπής σε σφάλματα έχουν αναπτυχθεί πιο σύγχρονες βιβλιοθήκες όπως η βιβλιοθήκη [Fiona](#) που είναι ιδιαίτερα χρήσιμη για την ανάγνωση/εγγραφή διανυσματικών δεδομένων και η βιβλιοθήκη [Shapely](#), η οποία χρησιμοποιείται για επεξεργασία και ανάλυση δεδομένων.

## Η βιβλιοθήκη Shapely

## shapely from WKT

Μπορούμε να δημιουργήσουμε αντικείμενα shapely που αναπαριστούν σημεία ή γραμμές ή πολύγωνα μέσω WKT. Η γλώσσα [WKT](#) είναι μια ειδική διάλεκτος για την περιγραφή διανυσματικών αντικειμένων. Εισάγουμε τις απαραίτητες υπο-βιβλιοθήκες ([geometry](#) και [wkt](#)) από την βιβλιοθήκη shapely.

```
import shapely.geometry
import shapely.wkt
```

Πολύγωνα

Καλούμε την μέθοδο [shapely.wkt.loads\(\)](#) για να δημιουργήσουμε shapely objects από wkt

```
pol1 = shapely.wkt.loads("POLYGON ((0 0, 0 -1, 7.5 -1, 7.5 0, 0 0))")
```

```
pol1
```



Η εκτύπωση του shapely αντικειμένου επιστρέφει μία περιγραφή σε μορφή WKT.

```
print(pol1)
```

```
POLYGON ((0 0, 0 -1, 7.5 -1, 7.5 0, 0 0))
```

Εκτύπωση του τύπου του αντικειμένου pol1

```
type(pol1)
```

```
shapely.geometry.polygon.Polygon
```

```
pol2 = shapely.wkt.loads("POLYGON ((0 1, 1 0, 2 0.5, 3 0, 4 0, 5 0.5, 6 -0.5, 7
-0.5, 7 1, 0 1))")
print(pol2)
```

```
POLYGON ((0 1, 1 0, 2 0.5, 3 0, 4 0, 5 0.5, 6 -0.5, 7 -0.5, 7 1, 0 1))
```

```
pol2
```



Δημιουργία MultiPolygon shapely object από αντίστοιχη συλλογή πολυγώνων MULTIPOLYGON

```
pol3 = shapely.wkt.loads("""
MULTIPOLYGON
(((40 40, 20 45, 45 30, 40 40)),
((20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20)))
""")
```

```
type(pol3)
```

```
shapely.geometry.multipolygon.MultiPolygon
```

```
pol3
```



```
type(pol3)
```

```
shapely.geometry.multipolygon.MultiPolygon
```

Σημεία

```
pnt1 = shapely.wkt.loads(
 "POINT (30 10)
 ")
```

```
pnt1
```



Συλλογή σημείων

```
pnt2 = shapely.wkt.loads(
 "MULTIPOINT(0 0,1 1)
 ")
```

```
pnt2
```



Γραμμές

```
line1 = shapely.wkt.loads(
 "LINESTRING(1.5 2.45,3.21 4)
 ")
```

```
line1
```



Συλλογή γραμμών

```
line2 = shapely.wkt.loads(
 "MULTILINESTRING((0 0,-1 -2,-3 -4, -3 -8),(2 3,3 4,6 7))
 ")
```

```
line2
```



Shapely μέσω συναρτήσεων

```
from shapely.geometry import Point, LineString, Polygon, MultiPoint,
MultiLineString, MultiPolygon
```

Κάθε συνάρτηση λαμβάνει διαφορετικά ορίσματα ανάλογα με την συνάρτηση.

Στην συνέχεια ακολουθεί η δημιουργία ενός αντικειμένου shapely γεωμετρίας σημείου.

```
pnt1 = shapely.geometry.Point((2, 0.5))
pnt1
```



Για την δημιουργία συλλογής σημείων `multipoint` δίνουμε σαν όρισμα στην συνάρτηση `shapely.geometry.MultiPoint` μία λίστα από πλειάδες (tuples). Η κάθε πλειάδα (`x, y`) αντιστοιχεί στις συντεταγμένες ενός σημείου.

```
coords = [(2, 0.5), (1, 1), (-1, 0), (1, 0)]
pnt2 = shapely.geometry.MultiPoint(coords)
pnt2
```



Για την δημιουργία γραμμών χρησιμοποιείται πάλι μία λίστα από tuples που περιγράφουν τις κορυφές της γραμμής. Στο παρακάτω παράδειγμα χρησιμοποιούμε την προηγούμενη πλειάδα αλλά πλέον χρησιμοποιούμε την μέθοδο `shapely.geometry.LineString` για την δημιουργία γραμμής και όχι την μέθοδο `shapely.geometry.MultiPoint` που δημιουργεί συλλογές σημείων.

```
line1 = shapely.geometry.LineString(coords)
line1
```



Κατασκεύη γραμμής από Point Objects:

```
p1 = shapely.wkt.loads("POINT (0 0)")
p2 = shapely.wkt.loads("POINT (1 1)")
p3 = shapely.wkt.loads("POINT (2 -1)")
p4 = shapely.wkt.loads("POINT (2.5 2)")
p5 = shapely.wkt.loads("POINT (1 -1)")

shapely.geometry.LineString([p1, p2, p3, p4, p5])
```



Αντίστοιχα μπορούμε να δημιουργήσουμε συλλογές γραμμών.

Δημιουργούμε ξεχωριστά αντικείμενα γραμμών shapely και στην συνέχεια καλούμε την συνάρτηση `shapely.geometry.MultiLineString` όπου ορίζουμε σαν όρισμα μια λίστα με τις μεμονωμένες γραμμές.

```
l1 = shapely.geometry.LineString([(2, 0.5), (1, 1), (-1, 0), (1, -1)])
l2 = shapely.geometry.LineString([(-2, 1), (2, -0.5), (3, 1)])
line2 = shapely.geometry.MultiLineString([l1, l2])
line2
```



Αντίστοιχα δημιουργούμε πολυγωνικά αντικείμενα μέσω μια λίστας με πλειάδες σημείων που χρησιμοποιείται σαν όρισμα στην συνάρτηση `shapely.geometry.Polygon`

```
coords = [(0, 0), (0, -1), (7.5, -1), [7.5, 0], (0, 0)]
shapely.geometry.Polygon(coords)
```



Αν θέλουμε μπορούμε να περάσουμε μία δεύτερη λίστα η οποία περιλαμβανει επιμέρους λίστες με πλειάδες σημείων τα οποία περιγράφουν τρύπες μέσα στο πολύγωνο.

```
coords_exterior = [(0, 0), (0, -1), (7.5, -1), (7.5, 0), (0, 0)]
coords_interiors = [[(0.4, -0.3), (5, -0.3), (5, -0.7), (0.4, -0.7), (0.4, -0.7)]]
shapely.geometry.Polygon(coords_exterior, coords_interiors)
```



Με την συνάρτηση `shapely.geometry.MultiPolygon` δημιουργούμε αντίστοιχα συλλογές πολυγώνων από μεμονωμένα αντικείμενα `shapely.geometry.polygon.Polygon`

```
pol2
```



```
multipolygon1 = shapely.geometry.MultiPolygon([pol1, pol2])
multipolygon1
```



```
print(multipolygon1)
```

```
MULTIPOLYGON (((0 0, 0 -1, 7.5 -1, 7.5 0, 0 0)), ((0 1, 1 0, 2 0.5, 3 0, 4 0, 5
0.5, 6 -0.5, 7 -0.5, 7 1, 0 1)))
```

Σύμφωνα με τις προδιαγραφές [simple features](#) το παραπάνω object δεν είναι έγκυρο γιατί ένα πολύγωνο τέμνει ένα άλλο σε άπειρο αριθμό σημείων. Μπορούμε να ελέγξουμε την εγκυρότητα ενός αντικειμένου με την κλήση τις ιδιότητας `is_valid`. Επίσης κατά την οπτικοποίηση στο προηγούμενο βήμα του αντικειμένου `multipolygon1` αυτό εμφανίζεται με κόκκινο (αντί για πράσινο). Ένδειξη ότι δεν είναι `valid`.

```
multipolygon1.is_valid
```

```
False
```

Ταυτόχρονα μπορούμε να φτιάξουμε σύνθετες συλλογές από επιμέρους αντικείμενα `shapely`.

```
geo_collection = shapely.geometry.GeometryCollection([multipolygon1, line2, pnt1])
```

```
geo_collection
```



## Shapely μέσω shape

Μπορούμε να δημιουργήσουμε αντικείμενα shapely μέσω της συνάρτησης `shapely.geometry.shape` η οποία δέχεται σαν όρισμα ένα λεξικό μορφής [GEOJSON](#) το οποίο πρέπει να έχει τις εξής δύο ιδιότητες.

- Την ιδιότητα `"type"` που περιγράφει τον τύπο γεωμετρίας
- Την ιδιότητα `"coordinates"` που περιγράφει τις γεωμετρίες και τις συντεταγμένες τους σαν λίστες ή πλειάδες.

```
d = {"type": "Point", "coordinates": (0, 1)}
shapely.geometry.shape(d)
```



Δημιουργία αντικειμένου MultiPolygon μέσω GEOJSON λεξικού:

```
d = {
 "type": "MultiPolygon",
 "coordinates": [
 [
 [[40, 40], [20, 45], [45, 30], [40, 40]]
],
 [
 [[20, 35], [10, 30], [10, 10], [30, 5], [45, 20],
 [20, 35]],
 [[30, 20], [20, 15], [20, 25], [30, 20]]
]
]
}
pol3 = shapely.geometry.shape(d)
pol3
```



## Γεωμετρικός τύπος δεδομένων

Η ιδιότητα `.geom_type` ενός αντικειμένου shapely περιγράφει τον γεωμετρικό τύπο της:

```
pol1.geom_type
```

```
'Polygon'
```

```
line1.geom_type
```

```
'LineString'
```

```
geo_collection.geom_type
```

```
'GeometryCollection'
```

## Συντεταγμένες

Για να ανακτήσουμε τις συντεταγμένες ενός shapely αντικειμένου καλούμε με διαφορετικό τρόπο τις ανάλογες συναρτήσεις ανάλογα την πολυπλοκότητα του κάθε τύπου.

Για ένα Point object καλούμε άμεσα την ιδιότητα coords η οποία επιστρέφει ένα shapely.coords.CoordinateSequence object.

```
pnt1.coords
```

```
<shapely.coords.CoordinateSequence at 0x7fb5f444b4f0>
```

Μπορούμε να λάβουμε ως λίστα τις πλειάδες συντεταγμένων που το απαρτίζουν

```
list(pnt1.coords)
```

```
[(2.0, 0.5)]
```

Αντίστοιχα για γραμμή

```
list(line1.coords)
```

```
[(2.0, 0.5), (1.0, 1.0), (-1.0, 0.0), (1.0, 0.0)]
```

Για να ελέγξουμε σε ένα αντικείμενο συλλογών γεωμετρίας (MultiPoint, MultiPolygon κτλ) πόσες επιμέρους γεωμετρίες περιλαμβάνει:

```
len(line2.geoms)
```

```
2
```

```
line2
```



Για να λάβουμε το πρώτο αντικείμενο γεωμετρίας:

```
line2.geoms[0]
```



```
type(line2.geoms[0])
```

```
shapely.geometry.linestring.LineString
```

```
line2.geoms[0].geom_type
```

```
'LineString'
```

Και λαμβάνουμε τις συντεταγμένες της πρώτης γεωμετρίας:

```
list(line2.geoms[0].coords)
```

```
[(2.0, 0.5), (1.0, 1.0), (-1.0, 0.0), (1.0, -1.0)]
```

ή της δεύτερης

```
list(line2.geoms[1].coords)
```

```
[(-2.0, 1.0), (2.0, -0.5), (3.0, 1.0)]
```

Για τα πολύγωνα ακολουθούμε διαφορετική προσέγγιση. Ένα πολύγωνο αποτελεί από το εξωτερικό περίγραμμα (exterior) ή και ένα ή περισσότερα περιγράμματα εσωτερικών οπών (interiors). Κατά συνέπεια έχουμε συντεταγμένες που περιγράφουν το κάθε περίγραμμα.

Το εξωτερικό περίγραμμα του pol1 αντικειμένου:

```
pol1.exterior
```



Και οι συντεταγμένες του:

```
list(pol1.exterior.coords)
```

```
[(0.0, 0.0), (0.0, -1.0), (7.5, -1.0), (7.5, 0.0), (0.0, 0.0)]
```

```
pol1
```



Το pol1 όμως δεν έχει εσωτερικές τρύπες για αυτό και το παρακάτω επιστρέφει μηδέν.

```
len(pol1.interiors)
```

```
0
```

Έστω το παρακάτω MultiPolygon object που δημιουργήσαμε σε προηγούμενο στάδιο.

```
pol3
```



Περιλαμβάνει δύο ξεχωριστά γεωμετρικά αντικείμενα:

```
len(pol3.geoms)
```

```
2
```

Ας πάρουμε το πρώτο:

```
pol3.geoms[0]
```



Δεν περιλαμβάνει καμία τρύπα στο εσωτερικό του. Ας πάρουμε τις συντεταγμένες από το περίγραμμά του (exterior)

```
pol3.geoms[0].exterior.coords
```

```
<shapely.coords.CoordinateSequence at 0x7fb5f4494ac0>
```

Και ας τις επιστρέψουμε σαν λίστα πλειάδων από ζεύγη της μορφής ( $x, y$ )

```
list(pol3.geoms[0].exterior.coords)
```

```
[(40.0, 40.0), (20.0, 45.0), (45.0, 30.0), (40.0, 40.0)]
```

Ας δοκιμάσουμε το δεύτερο αντικείμενο γεωμετρίας. Ας το δούμε:

```
pol3.geoms[1]
```



Λαμβάνουμε τις συντεταγμένες του εξωτερικού περιγράμματος:

```
list(pol3.geoms[1].exterior.coords)
```

```
[(20.0, 35.0),
 (10.0, 30.0),
 (10.0, 10.0),
 (30.0, 5.0),
 (45.0, 20.0),
 (20.0, 35.0)]
```

Ας δούμε πόσες τρύπες έχει στο εσωτερικό του:

```
len(pol3.geoms[1].interiors)
```

```
1
```

Παίρνουμε το περίγραμμα της τρύπας

```
pol3.geoms[1].interiors[0]
```



Και τις συντεταγμένες της

```
list(pol3.geoms[1].interiors[0].coords)
```

```
[(30.0, 20.0), (20.0, 15.0), (20.0, 25.0), (30.0, 20.0)]
```

Ιδιότητες αντικειμένων

Υπολογισμός ορίων (bounds)

```
geo_collection
```



```
geo_collection.bounds
```

```
(-2.0, -1.0, 7.5, 1.0)
```

```
shapely.geometry.box(*geo_collection.bounds)
```



```
line1
```



```
line1.bounds
```

```
(-1.0, 0.0, 2.0, 1.0)
```

```
pnt1
```



```
pnt1.bounds
```

```
(2.0, 0.5, 2.0, 0.5)
```

```
list(pnt1.coords)
```

```
[(2.0, 0.5)]
```

Υπολογισμός μήκους γραμμής

```
line1
```



```
line1.length
```

```
5.354101966249685
```

```
line2.geoms[0]
```



```
line2.geoms[0].length
```

```
5.5901699437494745
```

Υπολογισμός εμβαδού

```
pol1.area
```

```
7.5
```

```
pol2.area
```

```
6.25
```

Νέες γεωμετρίες

Κεντροειδές πολυγώνου (centroid)

```
pol2
```



```
pol2.centroid
```



```
shapely.geometry.GeometryCollection([pol2, pol2.centroid])
```



Περιμετρική ζώνη (buffer)

```
pnt1.buffer(5)
```



```
shapely.geometry.GeometryCollection([pnt1, pnt1.buffer(5)])
```



```
pol1.buffer(5)
```



```
shapely.geometry.GeometryCollection([pol1,pol1.buffer(5)])
```

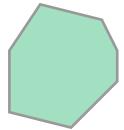


Convex hull

```
pol3
```



```
pol3.convex_hull
```



Σχέσεις μεταξύ αντικειμένων

```
shapely.geometry.GeometryCollection([pol1,pol3])
```



```
pol3
```



```
pol1.intersects(pol3)
```

```
False
```

```
shapely.geometry.GeometryCollection([pol1,pol2])
```

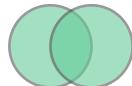


```
pol1.intersects(pol2)
```

```
True
```

## Γεωμετρικές πράξεις

```
x = shapely.geometry.Point((0, 0)).buffer(1)
y = shapely.geometry.Point((1, 0)).buffer(1)
shapely.geometry.GeometryCollection([x, y])
```



```
x.intersection(y)
```



```
x.difference(y)
```



```
x.union(y)
```



```
x.union(y)
```



Υπολογισμός απόστασης ανάμεσα σε δύο αντικείμενα

```
shapely.geometry.GeometryCollection([pol1, pol3])
```



```
pol1.distance(pol3)
```

```
10.307764064044152
```

```
pol3.distance(pol1)
```

```
10.307764064044152
```

Μετασχηματισμός προβολικού συστήματος:

```
import pyproj

from shapely.geometry import Point
from shapely.ops import transform

wgs84_pt = Point(39.35858398397631, 22.932070043920394)

wgs84 = pyproj.CRS('EPSG:4326')
greek_grid = pyproj.CRS('EPSG:2100')

project = pyproj.Transformer.from_crs(wgs84, greek_grid, always_xy=True).transform
projected_point = transform(project, wgs84_pt)
```

```
print(projected_point)
```

```
POINT (2087909.8290560723 2620022.621513317)
```

```
list(wgs84_pt.coords)[0]
```

```
(39.35858398397631, 22.932070043920394)
```

Προβολή δεδομένων σε διαδραστικό χάρτη leaflet μέσω της βιβλιοθήκης folium

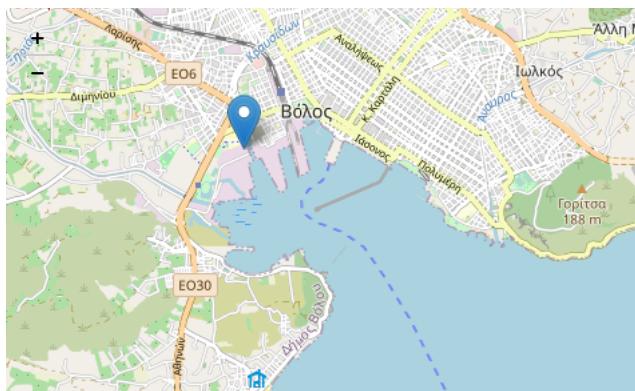
```
import folium

coords = list(wgs84_pt.coords)[0]

#Create the map
my_map = folium.Map(location = coords, zoom_start = 13)

Add marker
folium.Marker(coords, popup = 'Volos').add_to(my_map)

#Display the map
my_map
```



Leaflet (<https://leafletjs.com>) | Data by © OpenStreetMap (<http://openstreetmap.org>), under ODbL (<http://www.openstreetmap.org/copyright>).

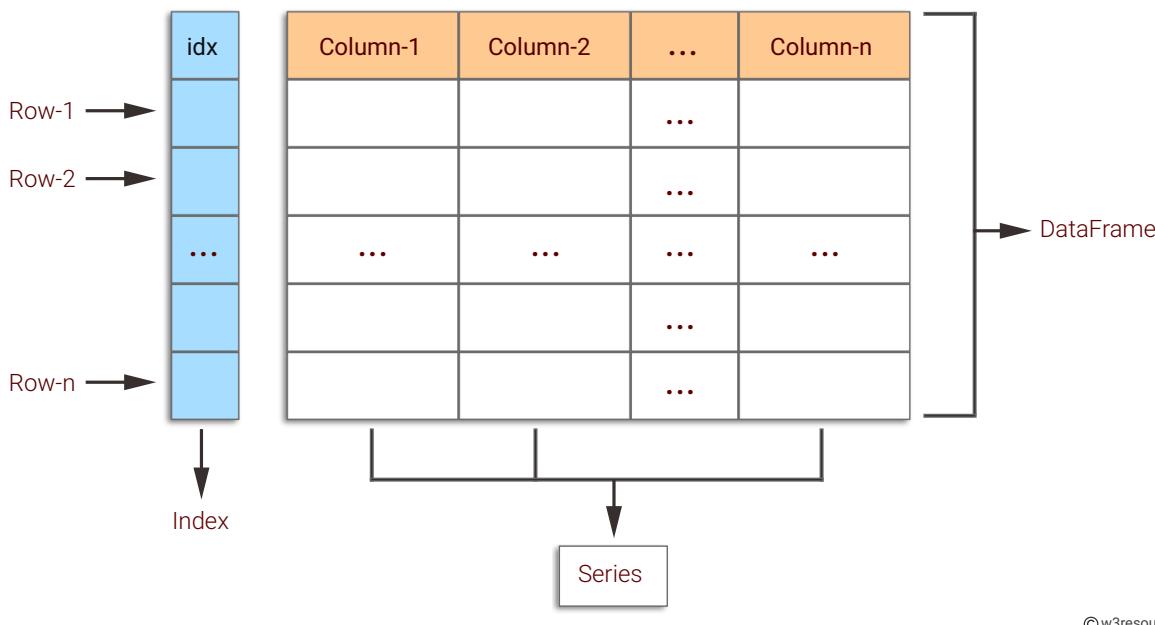
Η βιβλιοθήκη Geopandas

Μέχρι τώρα είδαμε πως μπορούμε να διαχειρίζομαστε γεωμετρικά δεδομένα με την βιβλιοθήκη shapely.

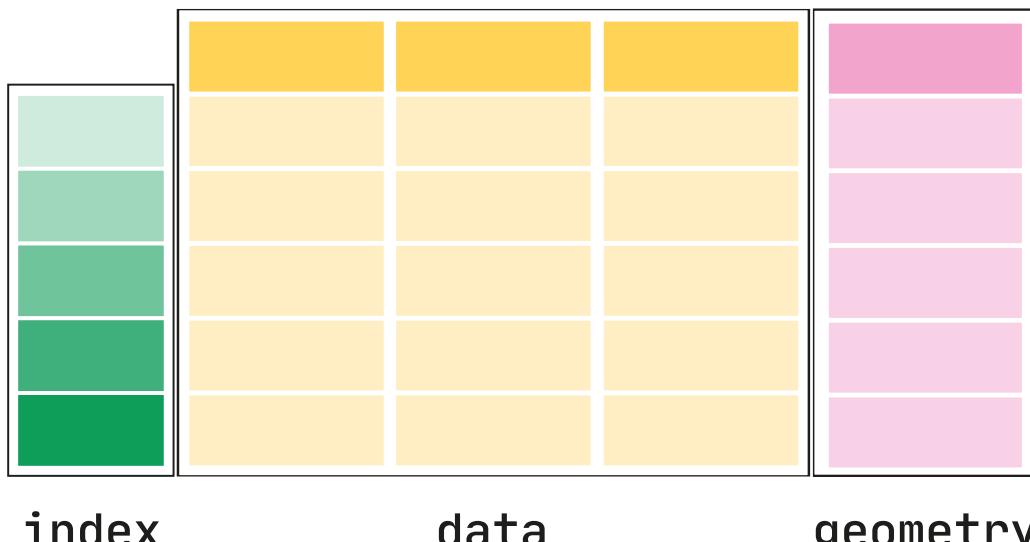
Όμως τα γεωγραφικά δεδομένα δεν περιλαμβάνουν μόνο της γεωγραφική πληροφορία και την γεωμετρική τους δομή αλλά συνοδεύονται από μια σειρά περιγραφικών δεδομένων. Αποτελεί επέκταση της βιβλιοθήκης [pandas](#) και «κληρονομεί» τα χαρακτηριστικά και τις δυνατότητες της.

Στην υφιστάμενη δομή της pandas, η geopandas υποστηρίζει την γεωμετρία με την προσθήκη μιας νέας στήλης και το γεωγραφικό σύστημα αναφοράς.

## Pandas Data structure



© w3resource.com



Εισάγουμε την βιβλιοθήκη με τον παρακάτω τρόπο

```
import geopandas as gpd
```

Για να διαβάσουμε ένα αρχείο shapefile χρησιμοποιούμε την συνάρτηση `gpd.read_file`. Το αντικείμενο που προκύπτει είναι τύπου `geopandas.geodataframe.GeoDataFrame`

```
Import shapefile using geopandas
dhmoi = gpd.read_file("../docs/dhmoi.gpkg")
type(dhmoi)
```

```
geopandas.geodataframe.GeoDataFrame
```

Η στήλη της γεωμετρίας ονομάζεται προκαθορισμένα `geometry` και είναι αντικείμενο `geopandas.geoseries.GeoSeries` που περιλαμβάνει αντικείμενα shapely.

```
type(dhmoi["geometry"])
```

```
geopandas.geoseries.GeoSeries
```

Με την μέθοδο `geom_type` μπορούμε να δούμε τον γεωμετρικό τύπο κάθε εγγραφής.

```
dhmoi.geom_type
```

```
0 MultiPolygon
1 MultiPolygon
2 MultiPolygon
3 MultiPolygon
4 MultiPolygon
 ...
320 MultiPolygon
321 MultiPolygon
322 MultiPolygon
323 MultiPolygon
324 MultiPolygon
Length: 325, dtype: object
```

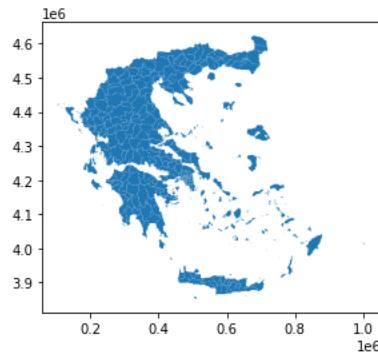
Ας δούμε τις αρχικές εγγραφές του αρχείου

```
dhmoi.head()
```

	OBJECTID	X	Y	Name	CodeELSTAT	PopM01	PopF
0	1	616259.007833	4.551127e+06	KOMOTINIS	0101	29967.0	3153
1	2	644783.135624	4.561364e+06	ARRIANON	0102	8952.0	930
2	3	602100.950461	4.555689e+06	IASMOU	0103	7290.0	756
3	4	633712.468442	4.539548e+06	MARONEIAS - SAPON	0104	8284.0	834
4	5	517831.195188	4.572544e+06	DRAMAS	0201	28041.0	2932

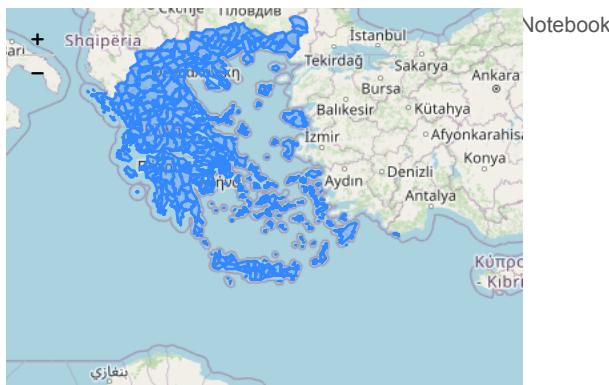
```
dhmoi.plot()
```

```
<AxesSubplot:>
```



Χαρτογραφική απόδοση σε διαδραστικό χάρτη

```
dhmoi.explore(legend=False)
```



Notebook

300 km

200 mi

Leaflet (<https://leafletjs.com>) | Data by © OpenStreetMap (<http://openstreetmap.org>), under ODbL (<http://www.openstreetmap.org/copyright>).

Εκτύπωση λεπτομερειών για το γεωγραφικό σύστημα αναφοράς το οποίο όπως φαίνεται παρακάτω είναι το ΕΓΣΑ ‘87 (EPSG:2100)

```
dhmoi.crs
```

```
<Derived Projected CRS: PROJCS["unknown",GEOGCS["unknown",DATUM["Greek_Geo ...>
Name: unknown
Axis Info [cartesian]:
- [east]: Easting (metre)
- [north]: Northing (metre)
Area of Use:
- undefined
Coordinate Operation:
- name: unnamed
- method: Transverse Mercator
Datum: Greek Geodetic Reference System 1987
- Ellipsoid: GRS 1980
- Prime Meridian: Greenwich
```

Μπορούμε να μετασχηματίσουμε τα δεδομένα σε ένα διαφορετικό προβολικό σύστημα με κλήση της μεθόδου `to_crs`

```
dhmoi_wgs84 = dhmoi.to_crs(4326)
dhmoi_wgs84.crs
```

```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

```
dhmoi.geom_type
```

```
0 MultiPolygon
1 MultiPolygon
2 MultiPolygon
3 MultiPolygon
4 MultiPolygon
...
320 MultiPolygon
321 MultiPolygon
322 MultiPolygon
323 MultiPolygon
324 MultiPolygon
Length: 325, dtype: object
```

Εκτύπωση των γεωγραφικών ορίων του αρχείου oikismoi που είναι στο σύστημα αναφορά ΕΓΣΑ ‘87.

```
dhmoi.total_bounds
```

```
array([104040.7266, 3850938. , 1007943. , 4624010.2508])
```

Εκτύπωση των γεωγραφικών ορίων του αρχείου oikismoi που είναι στο σύστημα αναφορά WGS’84.

```
dhmoi_wgs84.total_bounds
```

```
array([19.37377196, 34.80060523, 29.64123782, 41.74911332])
```

Η μέθοδος `shape` μας επιστρέφει τον πλήθος των γραμμών (325) και των στηλών (15).

```
dhmoi.shape
```

```
(325, 16)
```

Μπορούμε να εγγράψουμε ένα Geopandas Dataframe

```
dhmoi.to_file("dhmoi.geojson", driver="GeoJSON")
```

```
/home/leonidas/anaconda3/envs/book/lib/python3.10/site-
packages/geopandas/io/file.py:362: FutureWarning: pandas.Int64Index is deprecated
and will be removed from pandas in a future version. Use pandas.Index with the
appropriate dtype instead.
pd.Int64Index,
```

```
dhmoi.sort_values(by='PopTot01', ascending=False)
```

OBJECTID		X	Y	Name	CodeELSTAT	PopM01
192	193	476676.870086	4.204338e+06	ATHINAION	4501	374900.0
22	23	411331.939920	4.497210e+06	THESSALONIKIS	0701	185771.0
147	148	306893.782874	4.231792e+06	PATREON	3701	104429.0
245	246	468574.034133	4.199691e+06	PEIRAIOS	5101	87362.0
301	302	599279.522964	3.900401e+06	IRAKLEIOU	7101	81067.0
...	...	...	...	...	...	...
299	300	824000.991323	4.016217e+06	CHALKIS	6905	165.0
270	271	659987.832319	4.025277e+06	ANAFIS	6002	135.0
272	273	599675.697252	4.059438e+06	SIKINOU	6004	118.0
275	276	762387.857032	4.149989e+06	AGATHONISIOU	6102	92.0
320	321	507562.964610	3.855768e+06	GAVDOU	7403	49.0

325 rows × 16 columns

Ορισμός του index στην στήλη «CodeELSTAT»

```
dhmoi = dhmoi.set_index("CodeELSTAT")
```

```
dhmoi
```

	OBJECTID	X	Y	Name	PopM01	PopF01
CodeELSTAT						
0101	1	616259.007833	4.551127e+06	KOMOTINIS	29967.0	31534.0
0102	2	644783.135624	4.561364e+06	ARRIANON	8952.0	9307.0
0103	3	602100.950461	4.555689e+06	IASMOU	7290.0	7561.0
0104	4	633712.468442	4.539548e+06	MARONEIAS - SAPON	8284.0	8342.0
0201	5	517831.195188	4.572544e+06	DRAMAS	28041.0	29326.0
...	...	...	...	...	...	...
7403	321	507562.964610	3.855768e+06	GAVDOU	49.0	32.0
7404	322	475420.582085	3.905918e+06	KANTANOU - SELINOU	3271.0	3031.0
7405	323	465887.990746	3.920087e+06	KISSAMOU	5925.0	5545.0
7406	324	484468.969583	3.924051e+06	PLATANIA	9217.0	8647.0
7407	325	508083.904768	3.901847e+06	SFAKION	1288.0	1131.0

325 rows × 15 columns

Υπολογισμός έκτασης (σε \(\text{km}^2\)) σε μία νέα στήλη με το όνομα area

```
dhmoi["area"] = dhmoi.area*10e-6
```

```
dhmoi
```

	OBJECTID	X	Y	Name	PopM01	PopF01
CodeELSTAT						
0101	1	616259.007833	4.551127e+06	KOMOTINIS	29967.0	31534.0
0102	2	644783.135624	4.561364e+06	ARRIANON	8952.0	9307.0
0103	3	602100.950461	4.555689e+06	IASMOU	7290.0	7561.0
0104	4	633712.468442	4.539548e+06	MARONEIAS - SAPON	8284.0	8342.0
0201	5	517831.195188	4.572544e+06	DRAMAS	28041.0	29326.0
...	...	...	...	...	...	...
7403	321	507562.964610	3.855768e+06	GAVDOU	49.0	32.0
7404	322	475420.582085	3.905918e+06	KANTANOU - SELINOU	3271.0	3031.0
7405	323	465887.990746	3.920087e+06	KISSAMOU	5925.0	5545.0
7406	324	484468.969583	3.924051e+06	PLATANIA	9217.0	8647.0
7407	325	508083.904768	3.901847e+06	SFAKION	1288.0	1131.0

325 rows × 16 columns

Υπολογισμός του centroid κάθε δήμου σε μια νέα στήλη (centroid)

```
dhmoi['centroid'] = dhmoi.centroid
```

```
dhmoi
```

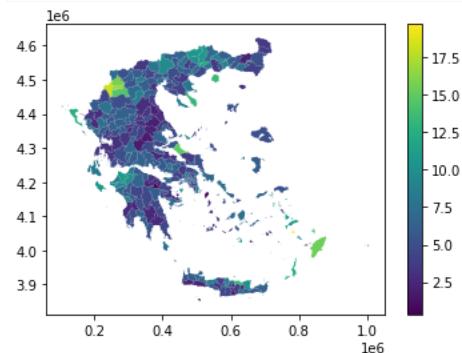
OBJECTID CodeELSTAT	X	Y	Name	PopM01	PopF01	
0101	1	616259.007833	4.551127e+06	KOMOTINIS	29967.0	31534.0
0102	2	644783.135624	4.561364e+06	ARRIANON	8952.0	9307.0
0103	3	602100.950461	4.555689e+06	IASMOU	7290.0	7561.0
0104	4	633712.468442	4.539548e+06	MARONEIAS - SAPON	8284.0	8342.0
0201	5	517831.195188	4.572544e+06	DRAMAS	28041.0	29326.0
...	...	...	...	...	...	
7403	321	507562.964610	3.855768e+06	GAVDOU	49.0	32.0
7404	322	475420.582085	3.905918e+06	KANTANOU - SELINOU	3271.0	3031.0
7405	323	465887.990746	3.920087e+06	KISSAMOU	5925.0	5545.0
7406	324	484468.969583	3.924051e+06	PLATANIA	9217.0	8647.0
7407	325	508083.904768	3.901847e+06	SFAKION	1288.0	1131.0

325 rows × 17 columns

Χαρτογραφική απόδοση του δείκτη ανεργίας (στήλη UnemrT01) ανά δήμο

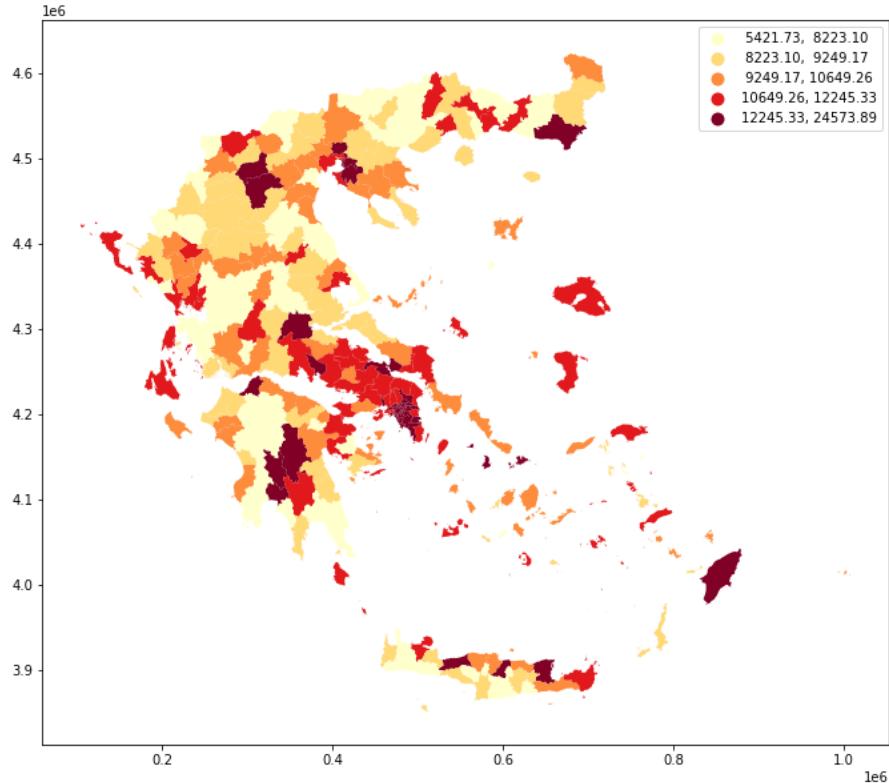
```
dhmoi.plot("UnemrT01", legend=True)
```

```
<AxesSubplot:>
```



```
dhmoi.plot("Income01", scheme='quantiles', cmap='YlOrRd', legend=True, figsize=(12, 10))
```

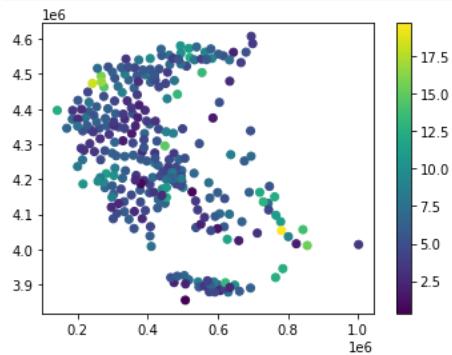
<AxesSubplot:>



Μπορούμε να οπτικοποιήσουμε διαφορετική στήλη τύπου `geopandas.GeoSeries` αφού πρώτα την ορίσουμε με την μέθοδο `set_geometry`.

```
dhmoi = dhmoi.set_geometry("centroid")
dhmoi.plot("UnemrT01", legend=True)
```

<AxesSubplot:>

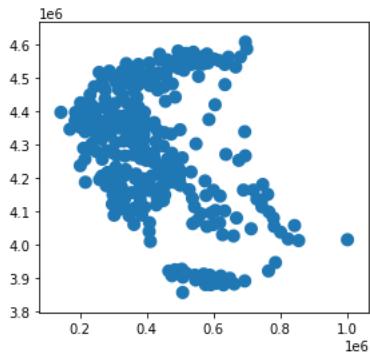


Ορισμός περιμετρικής ζώνης διαμέτρου 20χλμ γύρω από κάθε centroid.

```
dhmoi = dhmoi.set_geometry("centroid") # ορισμός default γεωμετρίας η στήλη
 centroid
dhmoi["buffered"] = dhmoi.buffer(20000) # εκτέλεση περιμετρικών ζωνών

dhmoi = dhmoi.set_geometry("buffered") # ορισμός default γεωμετρίας η στήλη
 buffered
dhmoi.plot(legend=True) # οπτικοποίηση
```

```
<AxesSubplot:>
```



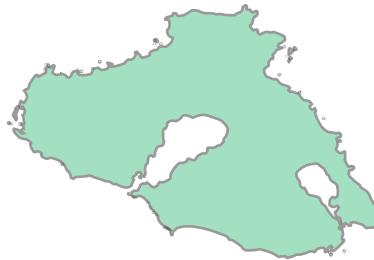
Ξανα ορίζουμε σαν προκαθορισμένη στήλη γεωμετρία την στήλη *geometry*.

```
dhmoi = dhmoi.set_geometry("geometry")
```

Μπορούμε να φιλτράρουμε γραμμές και στήλες όπως και στην pandas χρησιμοποιώντας το index και το όνομα στήλης:

```
lesvos = dhmoi.loc["5301", "geometry"]
```

```
lesvos
```



```
type(lesvos)
```

```
shapely.geometry.multipolygon.MultiPolygon
```

Η παραπάνω διαδικασία επιλογής επέστρεψε ένα shapely MultiPolygon object όπως φαίνεται.

Με την μέθοδο *dissolve* μπορούμε να συγχωνεύσουμε γεωμετρικά αντικείμενα βάση κάποιας στήλης. Αντικείμενα με ίδια τιμή θα συγχωνευθούν σε ενιαίο γεωμετρικό αντικείμενο.

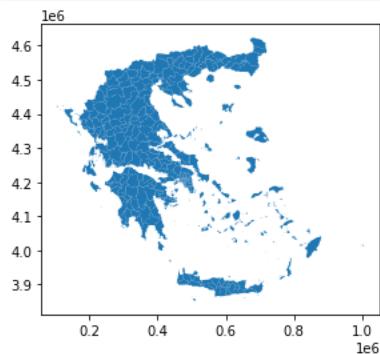
```
dhmoi.head()
```

OBJECTID CodeELSTAT	X	Y	Name	PopM01	PopF01
0101	1	616259.007833	4.551127e+06	KOMOTINIS	29967.0 31534.0
0102	2	644783.135624	4.561364e+06	ARRIANON	8952.0 9307.0
0103	3	602100.950461	4.555689e+06	IASMOU	7290.0 7561.0
0104	4	633712.468442	4.539548e+06	MARONEIAS - SAPON	8284.0 8342.0
0201	5	517831.195188	4.572544e+06	DRAMAS	28041.0 29326.0

```
nomoi = dhmoi.dissolve(by='Perif')
```

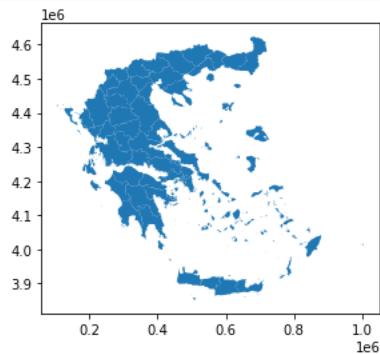
```
dhmoi.plot()
```

```
<AxesSubplot:>
```



```
nomoi.plot()
```

```
<AxesSubplot:>
```



Επιπλέον κατά την συγχώνευση μπορούμε να εφαρμόσουμε μια συνάρτηση στις αριθμητικές στήλες π.χ. *mean* ή *sum*.

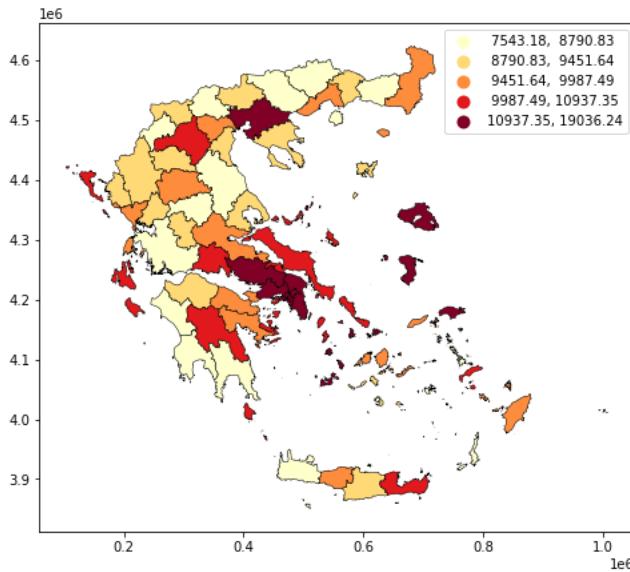
```

nomoi = dhmoi.dissolve(by='Perif', aggfunc='mean')

nomoi.plot(column = 'Income01', scheme='quantiles', edgecolor='black',
linewidth=0.5, cmap='YlOrRd', legend=True, figsize=(8, 8))

```

<AxesSubplot:>



## Βιβλιογραφία:

- Kalogirou S., 2020, Local Correlation, Spatial Inequalities, Geographically Weighted Regression and Other Tools (R package)
- GeoPandas, <https://geopandas.org/>, Πρόσβαση: 29/05/2022
- Prapas I., Analyze Geospatial Data in Python: GeoPandas and Shapely, <https://www.learndatasci.com/tutorials/geospatial-data-python-geopandas-shapely/>, Πρόσβαση: 29/05/2022
- Dorman M., 2022, VECTOR LAYERS, Geometries (shapely), <https://geobgu.xyz/py/shapely.html>, Πρόσβαση: 29/05/2022
- Wasser, Leah, Korinek, Nathan, & Palomino, Jenny. (2021). earthlab/earth-analytics-intermediate-earth-data-science-textbook: one more license fix (1.0.4) [Computer software]. Zenodo. <https://doi.org/10.5281/zenodo.5571001>

## 10. Διαγράμματα

Στόχος της τρέχουσας ενότητας είναι να παρουσιαστούν οι δυνατότητες που προσφέρει η Python για την οπτικοποίηση δεδομένων σε διαγράμματα. Για την δημιουργία των διαγραμμάτων θα χρησιμοποιηθεί η βιβλιοθήκη [seaborn] (<https://seaborn.pydata.org/>).

### Ειδική ενότητα για εκτέλεση στο Google Colab

```

έλεγχος αν το notebook τρέχει στο google colab
try:
 import google.colab
 IN_COLAB = True
except:
 IN_COLAB = False

αν το notebook τρέχει στο colab, mount το Google Drive και αλλαγή στο directory
που έχει γίνει clone το github repository.
εγκατάσταση απαραίτητων βιβλιοθηκών
if IN_COLAB:
 from google.colab import drive
 drive.mount('/content/drive')
 %cd /content/drive/MyDrive/Colab\ Notebooks/programming/notebooks
 !pip install pandas seaborn

```

```

import pandas as pd # η βιβλιοθήκη pandas για την διαχείριση πινάκων

import seaborn as sns # η βιβλιοθήκη seaborn για δημιουργία διαγραμμάτων
import matplotlib.pyplot as plt # η βιβλιοθήκη seaborn για δημιουργία
 διαγραμμάτων, πιο σύνθετη
sns.set() # ορισμός προκαθορισμένων ρυθμίσεων για την βιβλιοθήκη matplotlib και
 seaborn (themes, fonts, scaling, color palette)

```

## Διαγράμματα με γραμμές (line plots), ραβδογράμματα (bar plots), γραφήματα πίτας (pie charts)

Στο παρακάτω παράδειγμα θα παρουσιαστεί σε μορφή διαγράμματος με γραμμές (line chart) η διαχρονική εξέλιξη του πληθυσμού για την περίοδο 2010-2020 για μια σειρά από χώρες.

Για την επίδειξη της δημιουργίας ενός διαγράμματος σε γραμμές θα χρησιμοποιηθούν διαδέσιμα δεδομένα από την πλατφόρμα [kaggle](#).

### Πληθυσμός ανά χώρα για την περίοδο 2010-2020

**Πηγή:** Kaggle, Countries population. 2010 - 2020 data. ISO codes, <https://www.kaggle.com/datasets/cityapiio/countries-population-2010-2020-data> (Πρόσβαση, 28/12/2022)

Το εν λόγω dataset περιλαμβάνει ανά έτος τον πληθυσμό ανά χώρα για την περίοδο 2010-2020.

Αρχικά διαβάζουμε το αρχείο csv και προσαρτούμε τα περιεχόμενά του σε μια νέα μεταβλητή με το όνομα `mydata`.

```

mydata = pd.read_csv("../docs/plots_data/countries_general_info_historical.24-10-
2021.csv") #

```

Σύντομη προεπισκόπηση:

```

mydata.head()

```

	Name	NativeName	CallingCode	Iso3166P1Alpha2Code	Iso3166P1Alpha3Code	I
0	Canada	Canada	1	CA	CAN	
1	Japan	日本	81	JP	JPN	
2	Norway	Kongeriket Norge	47	NO	NOR	
3	Ireland	Eire	353	IE	IRL	
4	Hungary	Magyarorszag	36	HU	HUN	

Τα δεδομένα είναι διαθέσιμα σε μορφή `wide`. Αυτό σημαίνει ότι ο πληθυσμός κάθε έτους δίδεται σε διαφορετική στήλη. Για την ευκολότερη αξιοποίηση των δεδομένων πρέπει να μετατρέψουμε τα δεδομένα σε μορφή `long`.

(Διαβάστε για την διαφορά μεταξύ `wide` και `long` data format [εδώ](#).)

Μετατροπή και προεπισκόπηση:

```

mydata_long = pd.wide_to_long(mydata, stubnames=["Population"], i="Name",
j="year", sep="_",)
mydata_long

```

Name	year	Iso3166P1NumericCode	CallingCode	Iso3166P1Alpha3Code	Iso3166P1Alpha2Code
<b>Canada</b>	<b>2010</b>	124	1	CAN	
<b>Japan</b>	<b>2010</b>	392	81	JPN	
<b>Norway</b>	<b>2010</b>	578	47	NOR	
<b>Ireland</b>	<b>2010</b>	372	353	IRL	
<b>Hungary</b>	<b>2010</b>	348	36	HUN	
...	...	...	...	...	...
<b>Somalia</b>	<b>2019</b>	706	252	SOM	
<b>Sudan</b>	<b>2019</b>	729	249	SDN	
<b>Eswatini</b>	<b>2019</b>	748	268	SWZ	
<b>Kingdom of the Netherlands</b>	<b>2019</b>	528	31	NLD	
<b>Danish Realm</b>	<b>2019</b>	208	45	DNK	

1920 rows × 7 columns

Μετά την μετατροπή οι στήλες με τον πληθυσμό ανά έτος συγχωνεύτηκαν σε δύο στήλες μόνο. Η στήλη *Population* περιλαμβάνει τον πληθυσμό και η στήλη *year* περιλαμβάνει το έτος.

Σαν πρώτο παράδειγμα θα χρησιμοποιήσουμε μόνο τα δεδομένα της Ελλάδας, φιλτράροντας με βάση το πεδίο *Iso3166P1Alpha2Code*.

```
mydata_long=mydata_long[mydata_long.Iso3166P1Alpha2Code.isin(["GR"])]
mydata_long.head(5)
```

Name	year			
Greece	2010	300	30	GRC
	2011	300	30	GRC
	2012	300	30	GRC
	2013	300	30	GRC
	2014	300	30	GRC

Διάγραμμα γραμμής

Στην συνέχεια φτιάχνουμε ένα διάγραμμα με γραμμές με βάση τα δεδομένα της Ελλάδας.

```
mydata_long2 = mydata_long.reset_index()

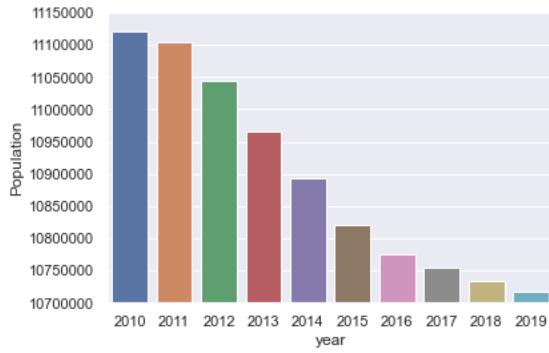
create plot
myplot = sns.lineplot(x = "year", y = "Population", data=mydata_long,
hue="Name", marker="o", palette=["green"])
myplot.set(title='Πληθυσμιακή μεταβολή 2010-2019') # Ορισμός τίτλου
myplot.ticklabel_format(style='plain', axis='y') # ορισμός y axis labels σε plain
format αντί scientific
plt.show()
```



Ραβδόγραμμα

Μπορούμε να αποδώσουμε το παραπάνω διάγραμμα και σαν ραβδόγραμμα όπου το μέγεθος του πληθυσμού κάθε έτους είναι μια στήλη το ύψος της οποίας εξαρτάται από το μέγεθος του πληθυσμού.

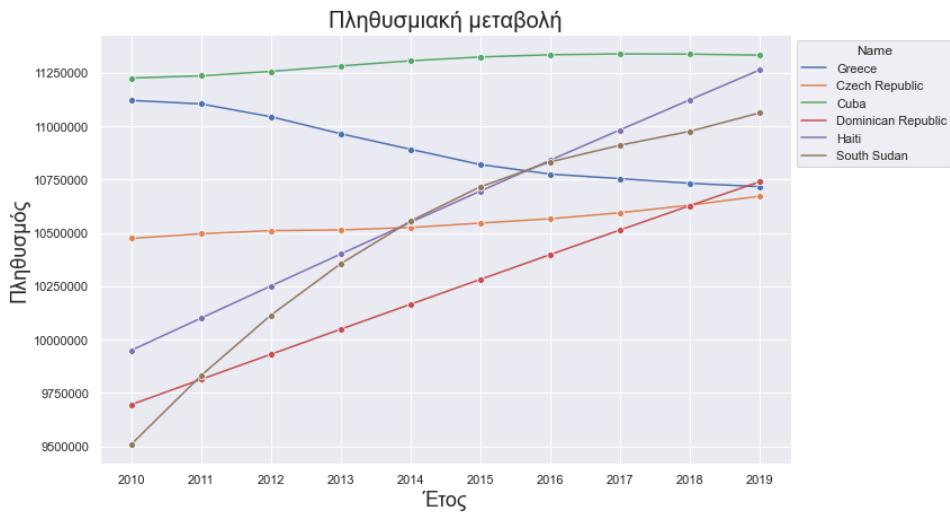
```
bar plot
myplot = sns.barplot(x = "year", y = "Population", data = mydata_long2)
myplot.ticklabel_format(style='plain', axis='y')
plt.ylim(10700000, 11150000)
plt.show()
```



Είναι δυνατή η δημιουργία διαγράμματος γραμμών με πολλαπλές γραμμές όπου κάθε γραμμή αντιστοιχεί στην διαχρονική μεταβολή του πληθυσμού διαφορετικής χώρας. Φιλτράρουμε τον αρχικό πίνακα (dataframe) και κρατάμε όσες χώρες είχαν πληθυσμό το 2019 > 10500000 και < 11500000. Αντίστοιχα μετατρέπουμε το φορμάτ του πίνακα από *wide* σε *long*.

```
mydata_2 = mydata[(mydata['Population_2019'] > 10500000) & (mydata['Population_2019'] < 11500000)]
mydata_long2 = pd.wide_to_long(mydata_2, stubnames=["Population"], i="Name",
j="year", sep=", ")
#mydata_long2
```

```
fig, myplot = plt.subplots(figsize = (11,7))
myplot = sns.lineplot(x = "year", y = "Population", data=mydata_long2,
hue="Name", marker="o")
myplot.axes.set_title("Πληθυσμιακή μεταβολή", fontsize=20)
myplot.set_xticks(mydata_long2.index.get_level_values(level="year").unique().tolist())
myplot.set_xlabel('Έτος', fontsize = 18)
myplot.set_ylabel('Πληθυσμός', fontsize = 18)
myplot.ticklabel_format(style='plain', axis='y')
#myplot.legend(loc='upper right', fontsize = 15);
sns.move_legend(myplot, "upper left", bbox_to_anchor=(1, 1))
```



```
mydata_long2[mydata_long2.index.isin([2019], level='year')]
```

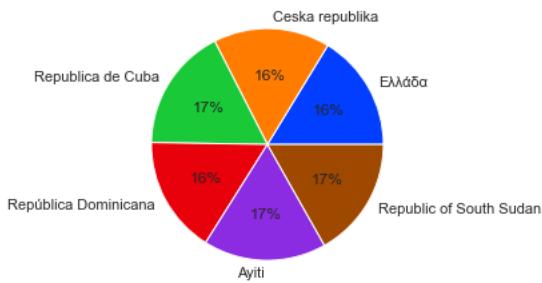
Name	year			
Greece	2019	300	30	GRC
Czech Republic	2019	203	420	CZE
Cuba	2019	192	53	CUB
Dominican Republic	2019	214	1809	DOM
Haiti	2019	332	509	HTI
South Sudan	2019	728	211	SSD

### Γράφημα πίτας

Αν θέλουμε να εκφράσουμε γραφικά την αναλογία του πληθυσμού του 2019 κάθε χώρας σε σχέση με το σύνολο τους τότε μπορούμε να χρησιμοποιήσουμε ένα διάγραμμα «πίτας» (pie chart).

```
pie chart
df = mydata_long2[mydata_long2.index.isin([2019], level='year')]
palette_color = sns.color_palette('bright')

pie = plt.pie(df['Population'], labels = df['NativeName'], colors = palette_color,
autopct='%.0f%%')
plt.show()
```



### Ιστόγραμμα συχνοτήτων, Θηκόγραμμα

Στην ενότητα αυτή θα χρησιμοποιηθούν δεδομένα πληθυσμού ανά έτος και περιφέρεια NUTS2 για την περίοδο 2010 - 2021.

(Πηγή: Eurostat, Population and demography, Demography, population stock and balance, Population on 1 January by NUTS 2 region, <https://ec.europa.eu/eurostat/web/population-demography/demography-population-stock-balance/database> (Πρόσβαση 28/12/2022) \_

Ανάγνωση του σχετικού αρχείο δεδομένων:

```
nuts2 = pd.read_csv("../docs/plots_data/eurostat/tgs00096_page_linear.csv") #
```

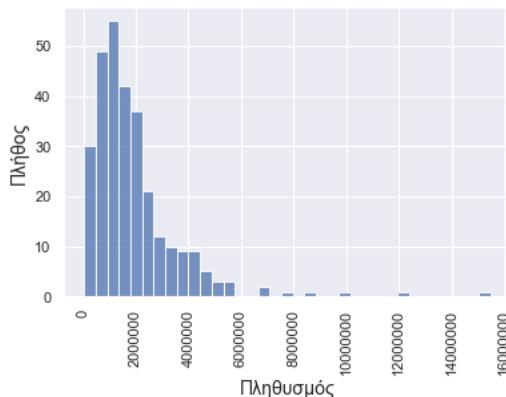
Φιλτράρουμε μόνο τον πληθυσμό για το 2021

```
nuts2_2021 = nuts2[nuts2['TIME_PERIOD'] == 2021]
```

## Ιστόγραμμα συχνοτήτων

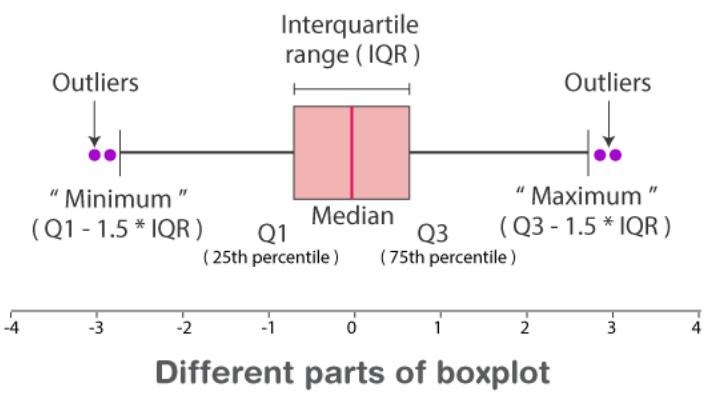
Το ιστόγραμμα συχνοτήτων περιγράφει την κατανομή μίας μεταβλητής. Μετράει τον αριθμό των παρατηρήσεων που περιέχονται ανά κλάση τιμών.

```
myplot = sns.histplot(data=nuts2_2021, x="OBS_VALUE", bins=35)
myplot.ticklabel_format(style='plain', axis='x')
myplot.set_xlabel('Πληθυσμός', fontsize = 14)
myplot.set_ylabel ('Πλήθος', fontsize = 14)
plt.xticks(rotation=90)
sns.set(rc={"figure.figsize": (10, 8)})
plt.show()
```



## Θηκόγραμμα (box-plot)

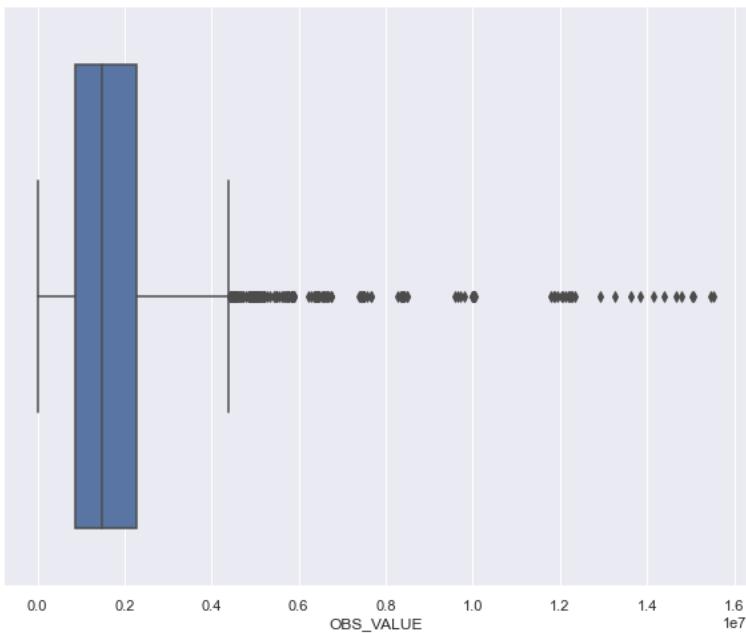
Το θηκόγραμμα περιγράφει την κατανομή μιας μεταβλητής μέσω της οπτικοποίησης του διαμέσου, του ενδοτεταρτημοριακού εύρους, του upper και lower whisker.



© Byjus.com

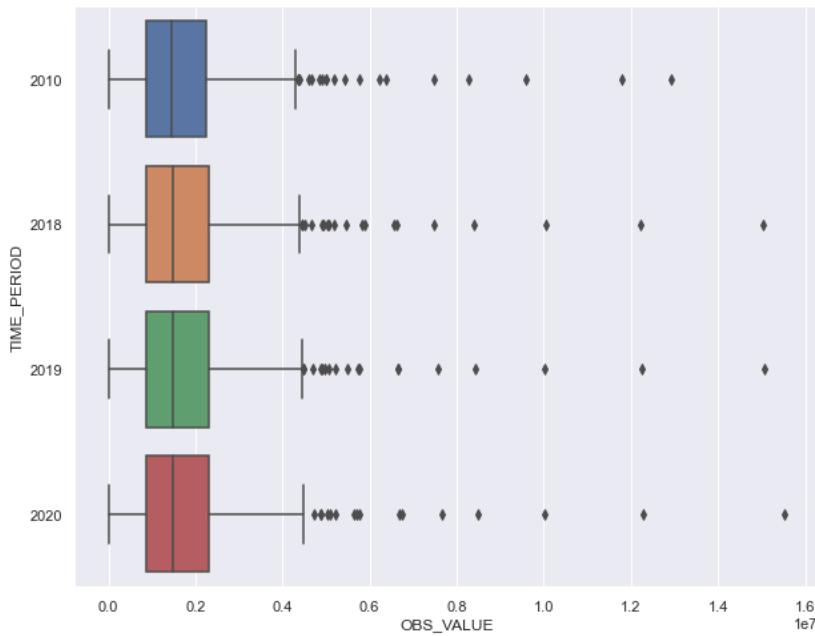
## Δημιουργία θηκογράμματος

```
sns.boxplot(data=nuts2_2021, x=nuts2["OBS_VALUE"])
plt.show()
```



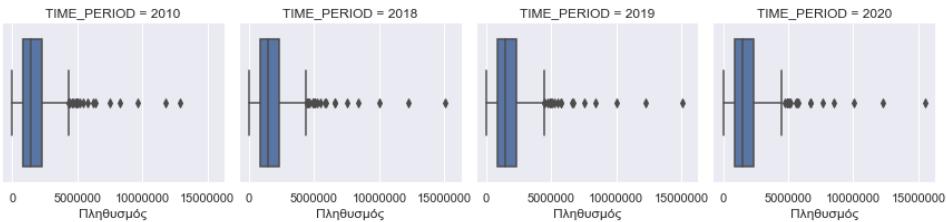
Για λόγους σύγκρισης μπορούμε να δημιουργήσουμε πολλαπλά θηκογράμματα με κριτήριο κατηγοριοποίησης των δεδομένων μια κατηγορική μεταβλητή. Στο συγκεκριμένο παράδειγμα χρησιμοποιούμε σαν κριτηρίο κατηγοριοποίησης την μεταβλητή `TIME_PERIOD` αφού πρώτα την μετατρέψουμε σε κατηγορική. Το αποτέλεσμα του παρακάτω κώδικα θα είναι να πάρουμε ένα θηκόγραμμα για τα δεδομένα κάθε έτους.

```
nuts2 = nuts2[nuts2['TIME_PERIOD'].isin([2010, 2018, 2019, 2020])]
nuts2['TIME_PERIOD'] = nuts2.TIME_PERIOD.astype('category') # SOS
sns.boxplot(data=nuts2, x="OBS_VALUE", y="TIME_PERIOD")
plt.show()
```



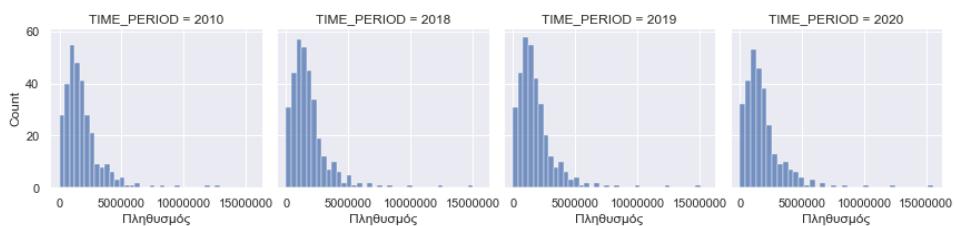
Επιπλέον είναι εφικτό να δημιουργήσουμε ξεχωριστά θηκογράμματα ανά έτος μέσω της συνάρτησης `FacetGrid`:

```
g = sns.FacetGrid(nuts2, col="TIME_PERIOD", height=3, col_wrap=4)
g.map(sns.boxplot, "OBS_VALUE", order=['TIME_PERIOD'])
g.set_xlabels('Πληθυσμός', fontsize = 12)
for ax in g.axes.flatten():
 ax.ticklabel_format(style='plain', axis='x')
plt.show()
```



Αντίστοιχα μπορούμε να κάνουμε το ίδιο με πολλαπλά ιστογράμματα. Το σημαντικό είναι να αλλάξουμε την τιμή της παραμέτρου που δέχεται η συνάρτηση `g.map` από `sns.boxplot` σε `sns.histplot`.

```
g = sns.FacetGrid(nuts2, col="TIME_PERIOD", height=3, col_wrap=4)
g.map(sns.histplot, "OBS_VALUE")
g.set_xlabels("Πληθυσμός", fontsize = 12)
for ax in g.axes.flatten():
 ax.ticklabel_format(style='plain', axis='x')
plt.show()
```



Διάγραμμα διασποράς (scatterplot), φυσαλίδων (bubble chart), μήτρας συσχετίσεων (correlation matrix), συντελεστής συσχέτισης, γραμμή παλινδρόμησης

```
Deaths by age, sex and NUTS 2 region
#https://ec.europa.eu/eurostat/databrowser/view/demo_r_magec/default/table?lang=en
```

Στην τρέχουσα ενότητα παρουσιάζεται η δημιουργία ενός διαγράμματος διασποράς με την χρήση οικονομικών δεδομένων από την Ελληνική στατιστική υπηρεσία (ΕΛ.ΣΤΑΤ.). Θα χρησιμοποιηθεί ο Οικονομικά ενεργός και μη ενεργός πληθυσμός, οι απασχολούμενοι κατά τομέα οικονομικής δραστηριότητας, και οι άνεργοι για τους δήμους της Ελλάδας.

(Πήγη: ΕΛΣΤΑΤ, Οικονομικά χαρακτηριστικά 2011, 23. Οικονομικά ενεργός και μη ενεργός πληθυσμός, απασχολούμενοι κατά τομέα οικονομικής δραστηριότητας, άνεργοι. Δήμοι, <https://www.statistics.gr/el/statistics/-/publication/SAM04/2011> (Πρόσβαση 28/12/2022))

Ανάγνωση του σχετικού αρχείου δεδομένων:

```
econ_data =
pd.read_csv("../docs/plots_data/A1602_SAM04_TB_DC_00_2011_B23_F_GR.csv", sep='; ')
#econ_data.head()
```

Το αρχείο περιλαμβάνει δεδομένα για διαφορετικά διοικητικά επίπεδα. Κρατάμε μόνο τους δήμους (έχουν Επίπεδο διοικητικής διαίρεσης με τιμή 3) και αφαιρούμε το Άγιο Όρος.

```
econ_data = econ_data[econ_data['Επίπεδο διοικητικής διαίρεσης']=='3']
econ_data = econ_data[econ_data['Περιγραφή']!='ΔΗΜΟΙ ΑΡΙΣΤΟΤΕΛΗ ΚΑΙ ΑΓΙΟ ΟΡΟΣ
(ΑΥΤΟΔΙΟΙΚΗΤΟ)']
```

Υπολογίζουμε το % απασχολουμένων ανά τομέα οικονομικής δραστηριότητας επί του οικονομικά ενεργού πληθυσμού, και το ποσοστό ανεργίας (ως νέα στήλη `unempl_pct`):

```
econ_data[['A_pct', 'B_pct', 'C_pct']] = econ_data[['Πρωτογενής Τομέας',
'Δευτερογενής Τομέας', 'Τριτογενής Τομέας']].apply(lambda x: 100*x/x.sum(),
axis=1)
econ_data['unempl_pct'] = 100*econ_data['Ανεργοί']/econ_data['Σύνολο οικονομικών
ενεργών']
```

Μετονομασία στήλης “Γεωγραφικός κωδικός” σε “DHMOSID”. Θα χρειαστεί στην συνέχεια σαν κλειδί για να συνδέσουμε πίνακες. Επιπλέον θα ορίσουμε τον τύπο δεδομένων της στήλης σε συμβολοσειρά.

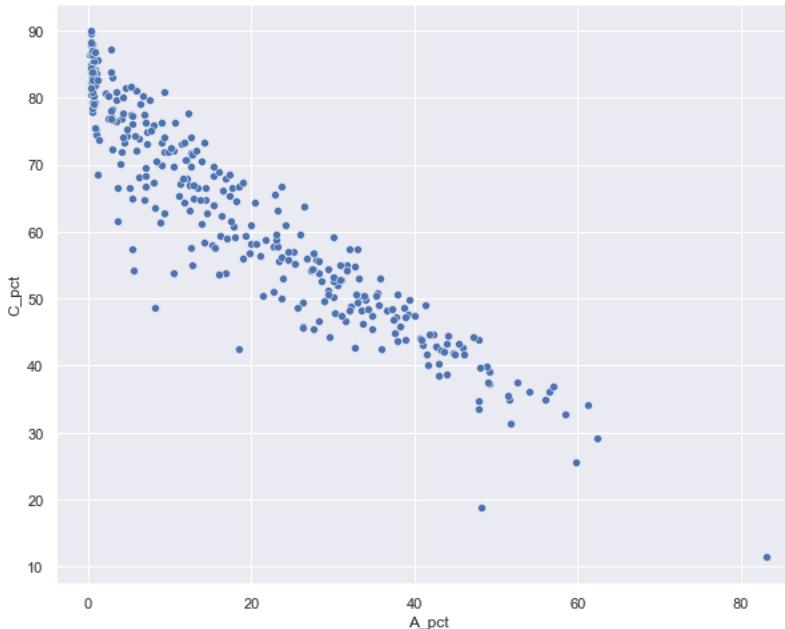
```
econ_data = econ_data.rename(columns={'Γεωγραφικός κωδικός': 'DHMOSID'})
econ_data['DHMOSID'] = econ_data['DHMOSID'].astype(str) # SOS
```

Είναι εφικτή η αποθήκευση ενός pandas dataframe σαν αρχείο (πχ CSV):

```
econ_data.to_csv('../docs/plots_data/output/econ_data.csv', index=False)
```

## Διάγραμμα διασποράς

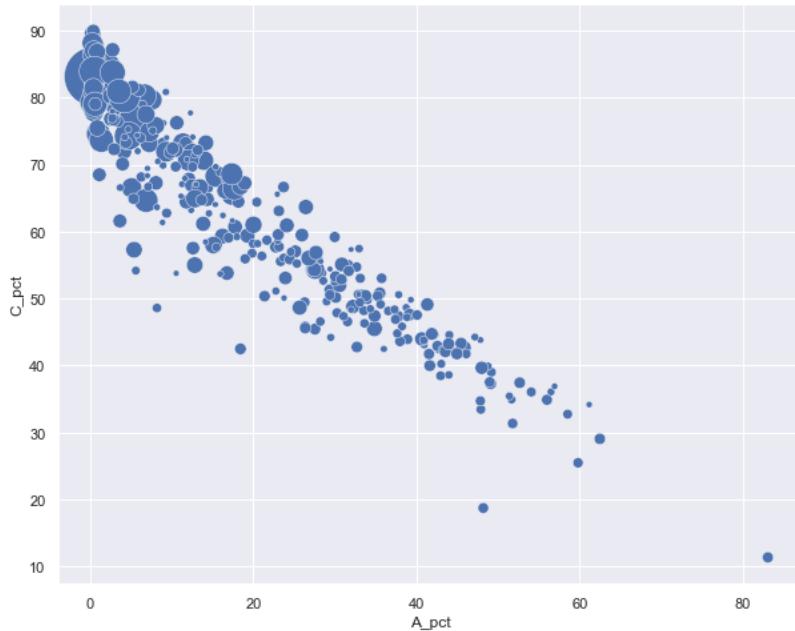
```
sns.scatterplot(data=econ_data, x="A_pct", y="C_pct")
plt.show()
```



## Bubble Chart

Μέσω της συνάρτησης scatterplot() της βιβλιοθήκης seaborn είναι δυνατή η δημιουργία *bubble charts*. Στην συνάρτηση αυτή υπάρχει η παράμετρος `size` η οποία καθορίζει το μέγεθος του κύκλου με βάση μια στήλη με αριθμητικά δεδομένα.

```
sns.scatterplot(data=econ_data, x="A_pct", y="C_pct", size="Σύνολο", legend=False,
sizes=(20, 2000))
plt.show()
```



## Ο συντελεστής συσχέτισης, γραμμή παλινδρόμησης, γράφημα μήτρας συσχετίσεων

Για την επίδειξη του κώδικα της τρέχουσας ενότητας θα αξιοποιηθούν δεδομένα απασχόλησης κατά επίπεδο εκπαίδευσης από την ΕΛ.ΣΤΑΤ. Στόχος είναι να υπολογίσουμε τον συντελεστής συσχέτισης ανάμεσα στο ποσοστό πτυχιούχων πρωτοβάθμιας εκπαίδευσης και το ποσοστό απασχολουμένων στον τριτογενή τομέα.

**Πηγή:** ΕΛΣΤΑΤ, Οικονομικά χαρακτηριστικά 2011, Β.10 Απασχολούμενοι κατά επίπεδο εκπαίδευσης. Δήμοι,

<https://www.statistics.gr/el/statistics/-/publication/SAM04/2011> (Πρόσβαση, 28/12/2022)

Ως συνήθως ξεκινάμε με την ανάγνωση του σχετικού αρχείου.

```
edu_data =
pd.read_csv("../docs/plots_data/A1602_SAM04_TB_DC_00_2011_B10_F_GR.csv", sep=';')
edu_data = edu_data[edu_data['Γεωγραφικό επίπεδο']==5]

edu_data['Γεωγραφικό επίπεδο'] = edu_data['Γεωγραφικό επίπεδο'].astype('int') # SOS μετατροπή στήλης σε ακέραιο

edu_data = edu_data.rename(columns={'Γεωγραφικός κωδικός': 'DHMOSID'})
edu_data['DHMOSID'] = edu_data['DHMOSID'].astype(str) # μετατροπή στήλης σε συμβολοσειρά
edu_data['DHMOSID'] = edu_data['DHMOSID'].str[1:] # αφαίρεση πρώτου χαρακτήρα για να ταιριάζει με το DHMOSID του econ_data
```

Υπολογισμός του ποσοστού πτυχιούχων ανά βαθμίδα εκπαίδευσης

```
edu_data[['clevel_pct', 'metadeyt_pct', 'blevel_pct', 'alevel_pct']] =
edu_data[['Κάτοχοι διδακτορικού ή μεταπτυχιακού τίτλου / Πτυχιούχοι Παν/μίου - Πολυτεχνείου, ΑΤΕΙ, ΑΣΠΑΙΤΕ ανώτερων επαγγελματικών και τσοτιμών σχολών', 'Πτυχιούχοι μεταδευτεροβάθμιας εκπαίδευσης (ΙΕΚ, Κολλέγια κλπ.) / Απόφοιτοι Λυκείου (Γενικού, Εκκλησιαστικού, Επαγγελματικού κλπ.)', 'Απόφοιτοι τριτάξιου Γυμνασίου και πτυχιούχοι Επαγγελματικών σχολών', 'Απόφοιτοι Δημοτικού / Άλλη περίπτωση(1)']].apply(lambda x: 100*x/x.sum(), axis=1)
```

## Σύνδεση πινάκων

```
df_cd = pd.merge(econ_data, edu_data, how='inner', on = 'DHMOSID')
```

```
df_cd.columns
```

```

Index(['Επίπεδο διαιρητικής διαίρεσης', 'DHMOSID', 'Περιγραφή', 'Σύνολο_X',
 'Σύνολο οικονομικών ενεργών', 'Σύνολο απασχολούμενων',
 'Πρωτογενής Τομέας', 'Δευτερογενής Τομέας', 'Τριτογενής Τομέας',
 'Ανεργοί', 'Οικονομικά μη ενεργοί', 'A_pct', 'B_pct', 'C_pct',
 'unempL_pct', 'Γεωγραφικό επίπεδο', 'Περιγραφή τόπου μόνιμης διαμονής ',
 'Σύνολο_y',
 'Κάτοχοι διδακτορικού ή μεταπτυχιακού τίτλου / Πτυχιούχοι Παν/μίου -',
 'Πολυτεχνείου, ΑΤΕΙ, ΑΣΠΑΙΤΕ ανώτερων επαγγελματικών και υπότιμων σχολών',
 'Πτυχιούχοι μεταδευτεροβάθμιας εκπαίδευσης (ΙΕΚ, Κολλέγια κλπ.) / Απόφοιτοι
Λυκείου (Γενικού, Εκκλησιαστικού, Επαγγελματικού κλπ.)',
 'Απόφοιτοι τριτάξιου υψηλασίου και πτυχιούχοι Επαγγελματικών σχολών',
 'Απόφοιτοι Δημοτικού / Άλλη περίπτωση(1)', 'clevel_pct', 'metadeyt_pct',
 'blevel_pct', 'alevel_pct'],
 dtype='object')

```

## Ο συντελεστής συσχέτισης Pearson

Ο συντελεστής συσχέτισης περιγράφει την στατιστική εξάρτηση μεταξύ δύο μεταβλητών.

```

corr = df_cd['alevel_pct'].corr(df_cd['C_pct'], method='pearson') # οι δυνατές
τιμές για την παράμετρο method είναι 'pearson', 'kendall' ή 'spearman'
corr

```

```
-0.867723183833861
```

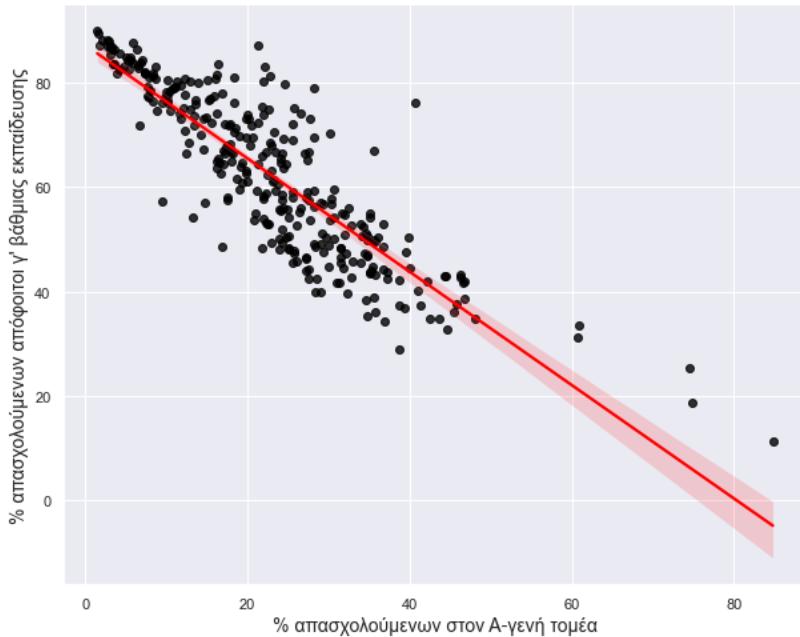
## Γραμμή παλινδρόμησης

```

myplot = sns.regplot(data=df_cd,
 x="alevel_pct",
 y="C_pct",
 scatter_kws={"color": "black"}, line_kws={"color": "red"})

myplot.set_xlabel('% απασχολούμενων στον Α-γενή τομέα', fontsize = 14)
myplot.set_ylabel ('% απασχολούμενων απόφοιτοι για βάθμιας εκπαίδευσης', fontsize
= 14)
sns.set(rc={"figure.figsize": (10, 8)})
plt.show()

```



Είναι δυνατή η αποθήκευση ενός διαγράμματος σε μορφή εικόνας:

```

fig = myplot.get_figure()
fig.savefig("../docs/plots_data/output/out_linear.png")

```

## Γράφημα μήτρας συσχετίσεων (Correlation matrix)

Για το συγκεκριμένο παράδειγμα θα χρησιμοποιηθεί ένα διαδεδομένο αρχείο δεδομένων, το αρχείο Iris. Το αρχείο Iris χρησιμοποιείται σαν δοκιμαστικό αρχείο για μεθόδους στατιστικής ανάλυσης, μηχανικής εκμάθησης και οπτικοποίησης και περιλαμβάνει 50 εγγραφές για κάθε ένα από τα είδη λουλουδιών Iris setosa, Iris virginica, και Iris versicolor για το μήκος και το πλάτος για τα σέπαλα και τα πέταλα τους. Περισσότερες λεπτομέρειες για το συγκεκριμένο αρχείο δεδομένων μπορείτε να βρείτε στην [Wikipedia](#).

Ως συνήθως αρχικά γίνεται η ανάγνωση του σχετικού αρχείου:

```
iris = pd.read_csv('../docs/plots_data/Iris.csv')
```

Η μήτρα συσχετίσεων καλείται μέσω της μεθόδου corr() ενός pandas dataframe και υπολογίζει την συσχέτιση μεταξύ των στηλών του αντίστοιχου dataframe. Το αποτέλεσμα είναι ένα dataframe.

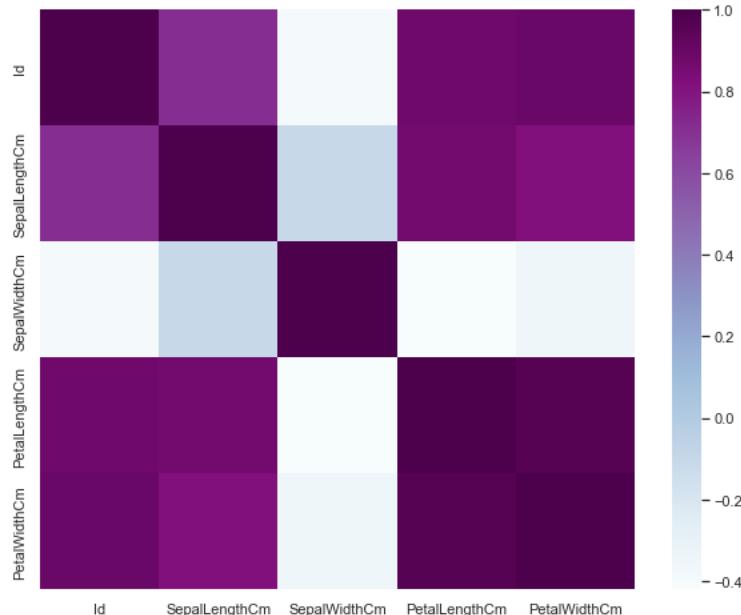
Στο τρέχον παράδειγμα η μήτρα συσχετίσεων υπολογίζεται με:

```
corr = iris.corr()
corr
```

	<b>Id</b>	<b>SepalLengthCm</b>	<b>SepalWidthCm</b>	<b>PetalLengthCm</b>	<b>PetalWidthCm</b>
<b>Id</b>	1.000000	0.716676	-0.397729	0.882747	0.899110
<b>SepalLengthCm</b>	0.716676	1.000000	-0.109369	0.871754	0.817131
<b>SepalWidthCm</b>	-0.397729	-0.109369	1.000000	-0.420516	-0.356100
<b>PetalLengthCm</b>	0.882747	0.871754	-0.420516	1.000000	0.962293
<b>PetalWidthCm</b>	0.899110	0.817954	-0.356544	0.962757	1.000000

Στην συνέχεια μπορεί να γίνει οπτικοποίηση σε διάγραμμα με την συνάρτηση heatmap της βιβλιοθήκης seaborn.

```
sns.heatmap(corr,
 xticklabels=corr.columns.values,
 yticklabels=corr.columns.values,
 cmap="BuPu")
```



## 11. Η βιβλιοθήκη numpy

Η ενότητα αποτελεί μια σύντομη εισαγωγή στην βιβλιοθήκη [numpy](#) η οποία χρησιμοποιείται για αριθμητική ανάλυση, αριθμητική λύση γραμμικών συστημάτων και επεξεργασία πολυδιάστατων πινάκων. Περισσότερες λεπτομέρειες για την βιβλιοθήκη παρέχονται [εδώ](#). Η θεμέλια δομή της βιβλιοθήκης είναι το ndarray δηλαδή ο πολυδιάστατος πίνακας. Είναι πολύ χρήσιμη στην επεξεργασία δεδομένων γιατί η πλειοψηφία των βιβλιοθηκών που διαχειρίζονται δορυφορικές εικόνες επιστρέφουν αντικείμενα numpy ndarrays.

## Ειδική ενότητα για εκτέλεση στο Google Colab

```
έλεγχος αν το notebook τρέχει στο google colab
try:
 import google.colab
 IN_COLAB = True
except:
 IN_COLAB = False

αν το notebook τρέχει στο colab, mount το Google Drive και αλλαγή στο directory
που έχει γίνει clone το github repository.
εγκατάσταση απαραίτητων βιβλιοθηκών
if IN_COLAB:
 from google.colab import drive
 drive.mount('/content/drive')
 %cd /content/drive/MyDrive/Colab\ Notebooks/programming/notebooks
 !pip install rasterio matplotlib

import numpy as np
import matplotlib.pyplot as plt
import rasterio as rio
from pathlib import Path
```

Ας φτιάξουμε έναν πρώτο πίνακα. Ακόμα και τα διανύσματα θα τα θεωρήσουμε σαν έναν πίνακα με μια στήλη. Τα στοιχεία ενός πίνακα πρέπει να είναι του ίδιου τύπου (είτε `integer` είτε `float` είτε `boolean`).

```
x=np.array([1,2,3,4])
print(x)
```

```
[1 2 3 4]
```

Για τον πίνακα αυτό μπορούμε να ανακτήσουμε μια σειρά ιδιοτήτων

Καταρχήν ας δούμε τι τύπος δεδομένων είναι ο πίνακας αυτός:

```
type(x)
```

```
numpy.ndarray
```

Ποιές είναι οι διαστάσεις του πίνακα;

```
print(x.shape)
```

```
(4,)
```

Και πόσες είναι οι διαστάσεις:

```
print(x.ndim)
```

```
1
```

Με την παρακάτω εντολή μπορούμε να ανακτήσουμε το μέγεθός του:

```
print(x.size)
```

```
4
```

Από τι τύπο δεδομένων αποτελούνται τα στοιχεία του:

```
print(x.dtype)
```

```
int64
```

Και πόση μνήμη καταλαμβάνει σε bytes:

```
print(x nbytes)
```

```
32
```

Ο επόμενος πίνακας έχει 4 γραμμές και 1 στήλη.

```
y=np.array([[1],[2],[3],[4]])
print(y.shape)
```

```
(4, 1)
```

Αν ελέγξουμε το πλήθος των διαστάσεων του, θα διαπιστώσουμε ότι έχει 2:

```
print(y.ndim)
```

```
2
```

Άρα ο πίνακας  $\langle x \rangle$  είναι μονοδιάστατος και ο πίνακας  $\langle y \rangle$  διάστατος. Ας επαναλάβουμε τον τρόπο σύνταξης. Δώστε προσοχή στην σύνταξη σε σχέση με τον αρχικό ορισμό που δώσαμε για τον πίνακα  $\langle x \rangle$  σε σχέση με τον τρέχοντα πίνακα  $\langle y \rangle$ .

```
x=np.array([1,2,3,4])
y=np.array([[1],[2],[3],[4]])

print(f'Διαστάσεις για τον πίνακα x: {x.ndim}\nΔιαστάσεις για τον πίνακα y:
{y.ndim}')
```

```
Διαστάσεις για τον πίνακα x: 1
Διαστάσεις για τον πίνακα y: 2
```

Μπορούμε να δημιουργήσουμε ένα numpy array μέσω μιας λίστα python:

```
l = [2, 25, 8, 1]
arr = np.asarray(l)
print(type(arr))
```

```
<class 'numpy.ndarray'>
```

Όπως είδαμε μέσω της εντολής `x.dtype` τα στοιχεία του πίνακα είναι ακέραιοι αριθμού (int64). Αν κατά την δημιουργία έστω και ένας αριθμός ήταν δεκαδικός (float), τότε όλα τα στοιχεία του πίνακα μετατρέπονται σε float.

```
x=np.array([1,2,3,4.5])
print(x.dtype)
```

```
float64
```

Μπορούμε ρητά να μετατρέψουμε τον τύπο δεδομένων των στοιχείων ένος πίνακα π.χ. από ακέραιο σε δεκαδικό

```
x=np.array([1,2,3,4])
print(x.dtype)
```

```
int64
```

```
x=x.astype(float)
print(x.dtype)
```

```
float64
```

Προσοχή στην απώλεια δεδομένων κατά την μετατροπή από ακέραιο σε δεκαδικό. Η παρακάτω μετατροπή οδηγεί σε στρογγυλοποιήσεις

```
x=np.array([1,2,3,4.5])
x=x.astype(int)
print(x)
```

```
[1 2 3 4]
```

Επίσης κατά την δημιουργία μπορούμε να ορίσουμε τον τύπο δεδομένων των στοιχείων:

```
x = np.array([[1,2,3],[4,5,6]], dtype = float)
print(x.dtype)
```

```
float64
```

Ανάλογα τον τύπο των στοιχείων αλλάζει και το μέγεθος. Αυτό μπορεί να δημιουργήσει προβλήματα μνήμης σε πολύ μεγάλα ndarrays π.χ σε μια πολυφασματική δορυφορική υψηλής ανάλυσης και μεγάλης έκτασης.

```
x = np.array([[1,2,3],[4,5,6]], dtype = np.uint32)
print(x nbytes)
```

```
24
```

```
print(x.astype(float). nbytes)
```

```
48
```

Μπορούμε να δημιουργήσουμε ένα ndarray το οποίο θα περιλαμβάνει μόνο την τιμή 1 στα στοιχεία του μέσω της συνάρτησης `np.ones`. Η παραπάνω εκτέλεση επιστρέφει float data type.

```
np.ones(5)
```

```
array([1., 1., 1., 1., 1.])
```

Μπορούμε ρητά να ορίσουμε τον τύπο δεδομένων στην συνάρτηση `np.ones`.

```
np.ones(5, dtype=int)
```

```
array([1, 1, 1, 1, 1])
```

Ή να είναι πολυδιάστατος πίνακας με στοιχεία με τιμές 1:

```
np.ones((5,2))
```

```
array([[1., 1.],
 [1., 1.],
 [1., 1.],
 [1., 1.],
 [1., 1.]])
```

Αντίστοιχα μπορούμε να δημιουργήσουμε ένα array που να περιλαμβάνει στοιχεία μόνο με 0.

```
np.zeros((3,3),dtype=int)
```

```
array([[0, 0, 0],
 [0, 0, 0],
 [0, 0, 0]])
```

## Πράξεις μεταξύ array

Πρόσθεση

```
x=np.array([[1,2],[3,4], [5,6]])
y=np.array([[1,1],[1,1], [2,2]])
v=x+y
print(v)
```

```
[[2 3]
 [4 5]
 [7 8]]
```

Αντιστοιχα μπορούμε να προσθέσουμε το array με έναν μόνο ακέραιο:

```
x=x+1
print(x)
```

```
[[2 3]
 [4 5]
 [6 7]]
```

Αφαίρεση

```
x=np.array([[1,2],[3,4], [5,6]])
y=np.array([[1,1],[1,1], [2,2]])
v=x-y
print(v)
```

```
[[0 1]
 [2 3]
 [3 4]]
```

Πολλαπλασιασμός. Σε αυτήν την περίπτωση χρησιμοποιείται η συνάρτηση `dot` και όχι το σύμβολο `*`: Μπορούμε να πολλαπλασιάσουμε ένα πίνακα με έναν αριθμό:

```
x=np.array([[1,2],[3,4],[5,6]])
v=np.dot(x, 2.5, out=None)
print(v)
```

```
[[2.5 5.]
 [7.5 10.]
 [12.5 15.]]
```

ή με έναν άλλο πίνακα. Σε αυτήν την περίπτωση το γινόμενο  $(x \cdot y)$  δύο πινάκων  $(x, y)$  ορίζεται μόνο όταν αριθμός των γραμμών του ενός πίνακα ισούται με τον αριθμό των στηλών του άλλου. Όπως παρατηρείτε στις επόμενες γραμμές κώδικα καλούμε την ιδιότητα `T` του πίνακα  $(y)$  για να κάνουμε αντιμετάθεση τις γραμμές με τις στήλες του (*ανάστροφος πίνακας*) και να πληρείται αυτή η συνθήκη.

```
y=np.array([[1,1],[1,1],[2,2]])
v=np.dot(x,y.T, out=None)
print(v)
```

```
[[3 3 6]
 [7 7 14]
 [11 11 22]]
```

Διαίρεση πινάκων:

```
v = x/y
print(v)
```

```
[[1. 2.]
 [3. 4.]
 [2.5 3.]]
```

Μπορούμε να φτιάξουμε μια ακολουθία τιμών σε έναν πίνακα `numpy.arange(start=1, stop=10, step=3)`

```
x = np.arange(start=1, stop=10, step=2) # η πιο απλά np.arange(1, 10, 2)
print(x)
```

```
[1 3 5 7 9]
```

Με την συνάρτηση `reshape` μπορούμε να αλλάξουμε τις διαστάσεις ενός πίνακα

```
x = np.arange(6)
print(x.shape)
```

```
(6,)
```

```
x = x.reshape(2,3)
print(x.shape)
```

```
(2, 3)
```

Επιπλέον μπορούμε με εύκολο τρόπο να υπολογίσουμε στατιστικά μέτρα θέσης και μεταβλητών της ενός πίνακα `ndarray`.

Ας δημιουργήσουμε ένα πίνακα με ακέραιους διαστάσεων:

```
np.random.rand(2023)
myarray= np.random.randint(1,11, size=(4,3))
print(myarray)
```

```
[[4 10 7]
 [9 5 4]
 [10 1 1]
 [5 7 6]]
```

Αν καλέσουμε τη παραπάνω εντολή θα πάρουμε το άθροισμα για τα στοιχεία που περιέχονται στο array.

```
np.sum(myarray)
```

```
69
```

Αν ορίσουμε την παράμετρο `axis=0` θα πάρουμε το άθροισμα για τις στήλες αν `axis=1` το άθροισμα για τις γραμμές.

```
np.sum(myarray, axis=0) # άθροισμα για τις στήλες, άξονας x
```

```
array([28, 23, 18])
```

```
np.sum(myarray, axis=1) # άθροισμα για τις γραμμές, άξονας y
```

```
array([21, 18, 12, 18])
```

Αντίστοιχα μπορούμε να χρησιμοποιήσουμε και άλλες συναρτήσεις:

```
np.amin(myarray)
```

```
1
```

```
np.amax(myarray, axis=0) # max κατά στήλη
```

```
array([10, 10, 7])
```

Αντί για συναρτήσεις μπορούμε να χρησιμοποιήσουμε μεθόδους από τα `numpy` objects που εκτελούν την αντίστοιχη λειτουργία, π.χ. για το `max`

```
myarray.max(axis=0)
```

```
array([10, 10, 7])
```

Και αντίστοιχα να πάρουμε και άλλα μέτρα όπως:

```
μέσος όρος
np.mean(myarray)
```

5.75

```
διάμεσος
np.median(myarray)
```

5.5

```
τυπική απόκλιση
np.std(myarray)
```

2.9190466480228325

```
np.percentile(myarray, 25) # 25th percentile
```

4.0

```
np.percentile(myarray, 75) # 75th percentile
```

7.5

```
np.percentile(myarray, 50) # 50th percentile or median
```

5.5

Σε αρκετές περιπτώσεις στα στοιχεία ενός πίνακας μπορεί να εχουν την τιμή *NaN* που σημαίνει «*Not a number*». Η τιμή του *NaN* χρησιμοποιείται όταν δεν υπάρχουν δεδομένα (missing values). Χαρακτηριστικό παράδειγμα είναι όταν διαβάζουμε μια δορυφορική εικόνα και κάνουμε flag ως *NaN* τα pixels που εχουν νεφοκάλυψη για να τα αποκλείσουμε από την ανάλυση. Και γενικότερα όταν θέλουμε να εξαιρέσουμε τιμές ή στοιχεία από μαθηματικές πράξεις μπορούμε να τα θέσουμε ως *NaN*. Προσοχή, το *NaN* δεν ισούται με 0. Να σημειωθεί ότι σε ένα array που περιέχει *NaN* στοιχεία πολλές από τις συναρτήσεις που προαναφέρθηκαν για τον υπολογισμό στατιστικών μέτρων επιστρέφουν *NaN*. Στην περίπτωση αυτή χρησιμοποιούνται παραλλαγές των συναρτήσεων (π.χ. *nansum* αντί *sum*). Για παράδειγμα έστω το παραπάνω array που περιέχει *NaN* στοιχεία μεταξύ των άλλων,

```
myarray=myarray.astype(float) #μετατροπή σε float (ακέραιο τύπο δεδομένων),
#αναγκαίο για να θέσουμε κάποια στοιχέια σε NaN
myarray[myarray<=4] = np.NaN # ορισμός σε NaN για όσες τιμές είναι <=4
myarray
```

```
array([[nan, 10., 7.],
 [9., 5., nan],
 [10., nan, nan],
 [5., 7., 6.]])
```

```
np.sum(myarray)
```

nan

```
np.nansum(myarray)
```

59.0

```
np.mean(myarray)
```

```
nan
```

```
np.nanmean(myarray)
```

```
7.375
```

Ανάλογη λειτουργία έχει και το module `numpy.ma` που χρησιμοποιείται για να κάνουμε mask τα στοιχεία ενός array. Δηλαδή όταν ορίσουμε κάποια στοιχεία σαν masked αυτά εξαιρούνται από τον υπολογισμό και τις διάφορες πράξεις που εκτελούνται στο array. Διαβάστε περισσότερα για τα mask arrays [εδώ](#).

```
import numpy.ma as ma #εισαγωγή της απαραίτητης βιβλιοθήκης
```

Έστω ο παρακάρω πίνακας

```
x=np.arange(6)
```

Μπορούμε να δημιουργήσουμε ένα masked array μέσω της συνάρτησης `masked_array`. Στην παράμετρο mask μπορούμε να επισημάνουμε ποιά στοιχεία θα είναι masked ορίζοντας την τιμή 1 1 (ή True)

```
x_masked = ma.masked_array(x, mask=[1,0,0,0,0,0])
```

```
print(x)
```

```
[0 1 2 3 4 5]
```

```
print(x_masked)
```

```
[-- 1 2 3 4 5]
```

```
x_masked
```

```
masked_array(data=[--, 1, 2, 3, 4, 5],
 mask=[True, False, False, False, False, False],
 fill_value=999999)
```

To `masked_array` έχει μια ιδιότητα που ονομάζεται `fill_value` και πρόκειται για μια τιμή που θα αντικαταστήσει τα masked στοιχεία όταν καλέσουμε την μέθοδο `filled()`.

```
x_masked.filled()
```

```
array([999999, 1, 2, 3, 4, 5])
```

Κατά την κλήση της μπορεί να οριστεί αυτή η τιμή πέρα από την προκαθορισμένη.

```
x_masked.filled(15)
```

```
array([15, 1, 2, 3, 4, 5])
```

Αν και το πρώτο στοιχείο το έχουμε κάνει masked, τα πραγματικά δεδομένα εξακολουθούν να υφίστανται:

```
x_masked.data
```

```
array([0, 1, 2, 3, 4, 5])
```

Όμως κατά τους διάφορους υπολογισμούς τα masked στοιχεία αγνοούνται π.χ. στον μέσο όρο:

```
np.mean(x_masked)
```

```
3.0
```

```
print(x_masked)
```

```
[-- 1 2 3 4 5]
```

Επιπλέον μπορούμε να χρησιμοποιήσουμε και άλλες χρήσιμες συναρτήσεις και μεθόδους στα ndarray objects.

Με τον παρακάτω τρόπο μπορούμε να εντοπίσουμε τις μοναδικές τιμές από τα στοιχεία ενός πίνακα. Έστω ο παρακάτω πίνακας

```
np.random.rand(2)
myarray= np.random.randint(1,11, size=(10,10))
myarray
```

```
array([[6, 7, 3, 3, 1, 6, 3, 4, 4, 8],
 [8, 6, 4, 3, 7, 1, 10, 6, 8, 4],
 [2, 5, 8, 2, 10, 4, 5, 9, 3, 3],
 [9, 4, 1, 1, 7, 8, 2, 9, 8, 1],
 [6, 1, 6, 10, 10, 8, 9, 8, 7, 5],
 [4, 2, 10, 5, 8, 6, 2, 4, 1, 2],
 [1, 5, 7, 7, 2, 6, 3, 3, 1, 1],
 [4, 9, 3, 2, 1, 10, 10, 6, 1, 1],
 [7, 3, 9, 9, 9, 10, 5, 5, 4, 7],
 [10, 8, 3, 9, 1, 5, 3, 4, 2, 10]])
```

```
unique, counts = np.unique(myarray, return_counts=True)
unique, counts
```

```
(array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]),
 array([14, 9, 12, 11, 8, 9, 8, 10, 9, 10]))
```

Μπορούμε να μετατρέψουμε ένα πίνακα διαστάσεων  $(i:j)$  σε διάνυσμα δηλ. σε πίνακα μιας μόνο στήλης με την χρήση της μεθόδου `flatten`. Ο πίνακας myarray είναι διαστάσεων  $10 \times 10$  και έχει 100 στοιχεία:

```
myarray.shape
```

```
(10, 10)
```

```
myarray.size
```

```
100
```

Καλώ την μέθοδο `flatten` και επιβεβαιώ τις διαστάσεις. Πλέον ολα τα στοιχεία είναι σε μια στήλη.

```
myarray_f = myarray.flatten()
myarray_f
```

```
array([6, 7, 3, 3, 1, 6, 3, 4, 4, 8, 8, 6, 4, 3, 7, 1, 10,
 6, 8, 4, 2, 5, 8, 2, 10, 4, 5, 9, 3, 3, 9, 4, 1, 1,
 7, 8, 2, 9, 8, 1, 6, 1, 6, 10, 10, 8, 9, 8, 7, 5, 4,
 2, 10, 5, 8, 6, 2, 4, 1, 2, 1, 5, 7, 7, 2, 6, 3, 3,
 1, 1, 4, 9, 3, 2, 1, 10, 10, 6, 1, 1, 7, 3, 9, 9, 9,
 10, 5, 5, 4, 7, 10, 8, 3, 9, 1, 5, 3, 4, 2, 10])
```

```
myarray_f.shape
```

```
(100,)
```

Σε ένα array μπορούμε να δοκιμάσουμε αν ισχύει μια συνθήκη στις τιμές της. Στην παρακάτω γραμμή κώδικα τεστάρουμε αν έστω και ένα στοιχείο περιέχει τιμές  $>8$ . Θα επιστρέψει `True` γιατί βλέπουμε ότι αρκετά στοιχεία έχουν την τιμή  $>8$ .

```
print(myarray)
```

```
[[6 7 3 3 1 6 3 4 4 8]
 [8 6 4 3 7 1 10 6 8 4]
 [2 5 8 2 10 4 5 9 3 3]
 [9 4 1 1 7 8 2 9 8 1]
 [6 1 6 10 10 8 9 8 7 5]
 [4 2 10 5 8 6 2 4 1 2]
 [1 5 7 7 2 6 3 3 1 1]
 [4 9 3 2 1 10 10 6 1 1]
 [7 3 9 9 9 10 5 5 4 7]
 [10 8 3 9 1 5 3 4 2 10]]
```

```
np.any(myarray > 8)
```

```
True
```

Αντίστοιχα μπορούμε να δοκιμάσουμε αν όλες οι τιμές ενός πίνακας πληρούν μια συνθήκη. Εδώ δοκιμάζουμε αν όλες οι τιμές του πίνακας είναι  $>2$ . Φυσικά η απάντηση είναι False γιατί υπάρχουν και τιμές  $<2$ .

```
np.all(myarray > 2)
```

```
False
```

Μπορούμε να κάνουμε τον αντίστοιχο έλεγχο κατά συγκεκριμένο άξονα (xaxis) δηλαδή κατά στήλη ή γραμμή. Στην επόμενη γραμμή δοκιμάζουμε σε κάθε στήλη (axis=0) αν περιλαμβάνεται έστω και μια τιμή  $>8$ . Όπως φαίνεται στην 4η και 9η στήλη δεν υπάρχει ούτε μια τιμή  $>8$ .

```
np.any(myarray > 8, axis=0)
```

```
array([True, True, True, True, True, True, True, True, False,
 True])
```

Έχουμε την δυνατότητα να ενώσουμε δύο πίνακες με την συνάρτηση **concatenate**:

```
arr = np.array([4, 7, 12])
arr1 = np.array([5, 9, 15])

Use concatenate() to join two arrays
con = np.concatenate((arr, arr1))
print(con)
```

```
[4 7 12 5 9 15]
```

Σε πολὺ διάστατους πίνακες μπορούμε να κάνουμε την ένωση με βάση συγκεκριμένο άξονα (παράμετρος axis), δηλ. κατά γραμμή ή στήλη.

```
arr = np.arange(20).reshape(4,5)
arr1 = np.arange(30,50).reshape(4,5)
con = np.concatenate((arr, arr1), axis=1) # κατά στήλη
print(con)
```

```
[[0 1 2 3 4 30 31 32 33 34]
 [5 6 7 8 9 35 36 37 38 39]
 [10 11 12 13 14 40 41 42 43 44]
 [15 16 17 18 19 45 46 47 48 49]]
```

Επίσης μπορούμε να ενώσουμε δύο arrays με την χρήση των συναρτήσεων **hstack** (οριζόντια, κατά στήλη) και **vstack** (κάθετα, κατά γραμμή). Ας δούμε το παρακάτω παράδειγμα:

```
arr1 = np.arange(15).reshape(3,5)
arr2=np.arange(10).reshape(2,5)
np.vstack((arr1, arr2))
```

```
array([[0, 1, 2, 3, 4],
 [5, 6, 7, 8, 9],
 [10, 11, 12, 13, 14],
 [0, 1, 2, 3, 4],
 [5, 6, 7, 8, 9]])
```

```
arr1 = np.arange(15).reshape(5,3)
arr2=np.arange(10).reshape(5,2)
np.hstack((arr1, arr2))
```

```
array([[0, 1, 2, 0, 1],
 [3, 4, 5, 2, 3],
 [6, 7, 8, 4, 5],
 [9, 10, 11, 6, 7],
 [12, 13, 14, 8, 9]])
```

## Indexing and Slicing

Σε μονοδιάστατα arrays η επιλογή στοιχείων γίνεται με την ίδια λογική όπως της λίστες στην Python όπου το πρώτο στοιχείο έχει το ευρετήριο `0` και το τελευταίο το ευρετήριο `n-1`.

```
np.random.rand(2)
arr1= np.random.randint(1,11, size=(10,))
fifth = arr1[5] # στοιχείο με ευρετήριο 4
last = arr1[arr1.size-1] # τελευταίο στοιχείο
print(fifth, last)
```

```
8 4
```

Σε πολυδιάστατους πίνακες η προσπέλαση στοιχείων γίνεται μέσω ευρετηρίων στον αντίστοιχο άξονα.

```
np.random.rand(2)
x = np.random.randint(1,11, size=(4,4))
x
```

```
array([[1, 4, 7, 2],
 [2, 2, 10, 8],
 [9, 9, 8, 7],
 [7, 7, 10, 6]])
```

Επιλογή των γραμμών με ευρετήριο 1 και 3 και όλες τις στήλες

```
x[1:3,:]
```

```
array([[2, 2, 10, 8],
 [9, 9, 8, 7]])
```

Επιλογή όλες τις γραμμες και τις στήλες με ευρετήριο 1 και 3

```
x[:,1:3]
```

```
array([[4, 7],
 [2, 10],
 [9, 8],
 [7, 10]])
```

Επιλογή των γραμμών και στηλών με ευρετήριο 1 και 3

```
x[1:3,1:3]
```

```
array([[2, 10],
 [9, 8]])
```

Επιλογή επιλεγμένων στοιχείων μέσω συγκεκριμένων ευρετηρίων. Από τις γραμμές 0,1,3 επιλέγω αντίστοιχα τα στοιχεία από τις στήλες 0,1,2

```
x[[0,1,3],[0,1,2]]
```

```
array([1, 2, 10])
```

Επιπλέον έχουμε την δυνατότητα να αντικαταστήσουμε τιμές σε ένα array, πχ στο array `x` μπορούμε να αντικάστησουμε την τιμή 10 με 99

```
x[x == 10] = 99
x
```

```
array([[1, 4, 7, 2],
 [2, 2, 99, 8],
 [9, 9, 8, 7],
 [7, 7, 99, 6]])
```

Ή να ορίσουμε περισσότερες συνθήκες

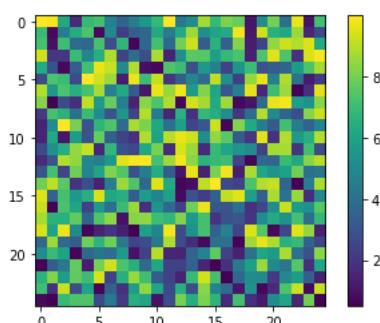
```
x[(x<3) | (x>8)] = 99
x
```

```
array([[99, 4, 7, 99],
 [99, 99, 99, 8],
 [99, 99, 8, 7],
 [7, 7, 99, 6]])
```

Αυτή η δυνατότητα είναι ιδιαίτερα χρήσιμη στην τηλεπισκόπηση. Έστω ότι έχουμε μια δορυφορική εικόνα τα εικονοστοιχεία της οποίας έχουν τις παρακάτω τιμές:

```
image = np.random.uniform(low=0.5, high=10, size=(25,25))
```

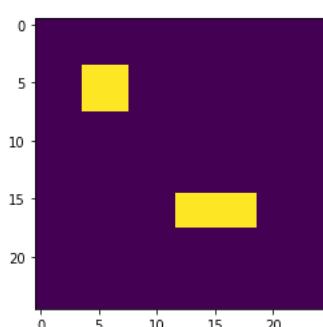
```
plt.imshow(image)
plt.colorbar()
plt.show()
```



Και έστω ότι έχουμε ένα άλλο αρχείο raster σε μορφή numpy ndarray το οποίο αποτελεί mask (πχ αρχείο νεφοκάλυψης). Τα στοιχεία που έχουν την τιμή 1 θα αποτελέσουν την μάσκα και τα 0 θα είναι τα έγκυρα εικονοστοιχεία. Με βάση αυτό το ndarray μπορούμε να θέσουμε ως μη έγκυρα όσα στοιχεία της image ταυτίζονται ευρετηριακά με τα στοιχεία της mask με τιμή 1. Ας φτιάξουμε ένα υποθετικό αρχείο mask.

```
mask = np.zeros((25, 25))
mask[4:8, 4:8] = 1
mask[15:18, 12:19] = 1
```

```
plt.imshow(mask)
plt.show()
```



```

import numpy.ma as ma
x_masked = ma.masked_array(image, mask=mask)

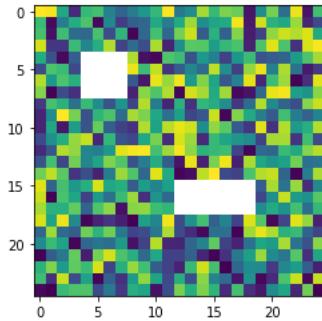
ή ενναλλακτικά ορισμός των elements σε NaN τιμές
#image[mask==1]=np.nan

```

```

plt.imshow(x_masked)
plt.show()

```



```

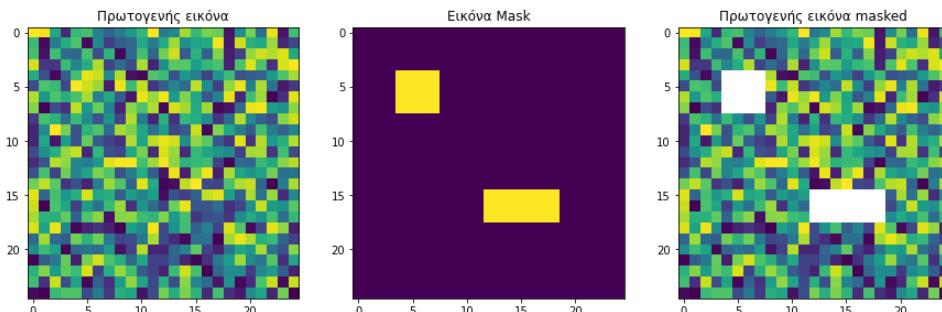
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 15))

ax1.imshow(image)
ax2.imshow(mask)
ax3.imshow(x_masked)

ax1.title.set_text('Πρωτογενής εικόνα')
ax2.title.set_text('Εικόνα Mask')
ax3.title.set_text('Πρωτογενής εικόνα masked')

plt.show()

```



Αρκετές από τις βιβλιοθήκες για ψηφιδωτά δεδομένα στην Python επιστρέφουν τα δεδομένα των αρχείων ως numpy ndarrays. Στο παράδειγμα που ακολουθεί θα χρησιμοποιηθεί η βιβλιοθήκη [rasterio](#) για την ανάγνωση μια πολυφασματικής δορυφορικής εικόνας.

```

INPUTDIR = Path('data')
with rio.open(INPUTDIR / 'pre_fire.tif') as src:
 image = src.read()
 print(type(image))

```

```

<class 'numpy.ndarray'>

```

όπως βλέπουμε η παραπάνω βιβλιοθήκη επιστρέφει ένα numpy.ndarray object όπου έχει, 3 διαστάσεις, 5 bands διαστάσεων 572x1040 η κάθε μία.

```

image.ndim, image.shape

```

```

(3, (5, 572, 1040))

```

## 12. Άσκηση επεξεργασίας και ανάλυσης γεωχωρικών δεδομένων

Ειδική ενότητα για εκτέλεση στο Google Colab

```

έλεγχος αν το notebook τρέχει στο google colab
try:
 import google.colab
 IN_COLAB = True
except:
 IN_COLAB = False

αν το notebook τρέχει στο colab, mount το Google Drive και αλλαγή στο directory
που έχει γίνει clone το github repository.
εγκατάσταση απαραίτητων βιβλιοθηκών
if IN_COLAB:
 from google.colab import drive
 drive.mount('/content/drive')
 %cd /content/drive/MyDrive/Colab\ Notebooks/programming/notebooks
 !pip install geopandas pandas rasterio numpy matplotlib folium scikit-learn
 scikit-image

```

Υπολογισμός καμένων εκτάσεων με δορυφορικά δεδομένα Sentinel-2, τον δείκτη Relativized Burn Ratio (RBR) και τον αλγόριθμο ομαδοποίησης k-means.

Στόχος του τρέχοντος jupyter notebook είναι να επιδειχθούν συνοπτικά οι δυνατότητες της Python στην γεωεπεξεργασία και χωρικής ανάλυσης.

Σαν μελέτη περίπτωσης επιλέχθηκε η πυρκαγιά στο Μάτι Αττικής στις 23 Ιουλίου 2018. Σκοπός είναι να αποτυπώσουμε χωρικά την έκταση της πυρκαγιάς, να υπολογίσουμε την έκταση της δομημένης περιοχής που έχει κάει αλλά και να διαπιστώσουμε ποιοί ΟΤΑ έχουν πληγεί περισσότερο. Κατά την εκτέλεση των διαδικασιών προγραμματισμού θα γίνει χρήση μεθόδων τηλεπισκόπησης, δορυφορικών δεδομένων Sentinel2 και διανυσματικών γεωχωρικών δεδομένων.

## Ανάγνωση δεδομένων

Αρχικά γίνεται εισαγωγή των απαραίτητων βιβλιοθηκών καθώς και ο ορισμός των καταλόγων που περιέχουν τα δεδομένα εισόδου (input data) και τα αποτελέσματα της ανάλυσης (output data).

```

import rasterio # βιβλιοθήκη για την ανάγνωση, επεξεργασία και εγγραφή διανυσματικών δεδομένων
from rasterio import features # υπορουτίνα της rasterio για την διαχείριση γεωγραφικών αντικειμένων (features)
import rasterio.mask # υπορουτίνα της rasterio για την δημιουργία Mask σε ψηφιδωτά δεδομένα
from rasterio.crs import CRS # υπορουτίνα της rasterio για την διαχείριση προβολικών συστημάτων
from rasterio.plot import show,show_hist # υπορουτίνες την rasterio για την οπτικοποίηση ψηφιδωτών (raster) δεδομένων και ιστογραμάτων
from rasterio.io import MemoryFile # υπορουτίνα της rasterio για την δημιουργία προσωρινών αρχείων στην μνήμη
from rasterio.warp import reproject, Resampling, calculate_default_transform # υπορουτίνες της rasterio για αλλαγή προβολικού και μετασχηματικό ψηφιδωτών δεδομένων

import pandas as pd # η βιβλιοθήκη pandas για την διαχείριση και ανάλυση δεδομένων
import geopandas as gpd # επέκταση της βιβλιοθήκης pandas για την διαχείριση και ανάλυση γεωγραφικών δεδομένων
import numpy as np # βιβλιοθήκη για την μαθηματική επεξεργασία πινάκων (arrays)

#from shapely.geometry import shape
import fiona # βιβλιοθήκη για την διαχείριση γεωγραφικών αντικειμένων (features)

from matplotlib import pyplot as plt # βιβλιοθήκη για την οπτικοποίηση δεδομένων σε διαγράμματα
import matplotlib.patches as mpatches
from matplotlib.colors import ListedColormap

import folium # οπτικοποίηση γεωγραφικών δεδομένων στην Python μέσω της βιβλιοθήκης leaflet
from sklearn import cluster # υπορουτίνα της scikit-learn για μη επιβλεπόμενη ομαδοποίηση (unsupervised clustering algorithms)

from pathlib import Path # βιβλιοθήκη για την διαχείριση διαδρομών (paths)

from skimage import data, img_as_float # skimage, βιβλιοθήκη για την επεξεργασία εικόνας
from skimage import exposure

Input directory
INPUTDIR = Path('data')

Output directory
OUTDIR = Path('output')

```

Στην συνέχεια ανοίγουμε τα raster από τον Sentinel-2 για ημερομηνία πρίν την φωτιά. Επίσης εκτυπώνουμε μερικές χρήσιμες ιδιότητες των δεδομένων όπως διαστάσεις

```

with rasterio.open(INPUTDIR / 'pre_fire.tif') as src:
 image = src.read() # θα επιστρέψει ένα numpy array της μορφής [band, row, col].
 #show((src,1), cmap='pink') #
 https://matplotlib.org/stable/tutorials/colors/colormaps.html
 print(src.crs) # πληροφορίες για το προβολικό σύστημα
 print("Shape: ", src.shape) # πληροφορίες για τις διαστάσεις του numpy array
 print("Bounds: ", src.bounds) # πληροφορίες για την γεωγραφική έκταση των δεδομένων
 print("Layers: ", src.count) # πληροφορίες για τον αριθμό των layers
 print("Profile: ", src.profile) # τα ανωτέρω περιγράφονται αναλυτικά μέσω του profile property
 print("Mean: ", image[0,:,:].mean()) # ενδεικτικά μπορούμε να πάρουμε μέσο όρο τιμών για το πρώτο layer το οποίο λαμβάνουμε με image[0,:,:]

 # Βάζουμε τα δεδομένα κάθε καναλιού (band) σε ξεχωριστή μεταβλητή. Θα χρειαστεί στην συνέχεια για τον υπολογισμό δεικτών
 B2_pre,B3_pre,B4_pre,B8_pre,B12_pre = (image[i,:,:] for i in range(0,src.count))

 # λαμβάνουμε το profile σαν ξεχωριστό object. Αυτό το object μπορούμε να το τροποποιήσουμε και να το χρησιμοποιήσουμε στην συνέχεια
 # για τις ανάγκες δημιουργίας νέων αρχείων.
 dNBR_meta = src.profile
 dNBR_meta.update({'driver':'GTiff',
 'width':src.shape[1],
 'height':src.shape[0],
 'count':1,
 'dtype':'float64',
 'crs':src.crs,
 'transform':src.transform,
 'nodata':-9999})

```

```

EPSG:32634
Shape: (572, 1040)
Bounds: BoundingBox(left=757670.0, bottom=4211870.0, right=768070.0, top=4217590.0)
Layers: 5
Profile: {'driver': 'GTiff', 'dtype': 'uint16', 'nodata': None, 'width': 1040, 'height': 572, 'count': 5, 'crs': CRS.from_epsg(32634), 'transform': Affine(10.0, 0.0, 757670.0, 0.0, -10.0, 4217590.0), 'blockxsize': 528, 'blockysize': 576, 'tiled': True, 'compress': 'lzw', 'interleave': 'pixel'}
Mean: 684.4699451990317

```

Οπτικοποίηση του B2 (B2\_pre) καναλιού (πρόκειται για το πρώτο κανάλι= `image[0,:,:]`). Στην πρώτη προβολή της εικόνας αυτή έχει χαμηλό contrast. Για καλύτερη οπτικοποίηση θα επέμβουμε με μια διαδικασία που λέγεται histogram stretching κατά την οποία η εικόνα γίνεται rescaled στο εύρος τιμών που περιλαμβάνονται ανάμεσα στο 2ο και 98ο τεταρτημόριο.

```

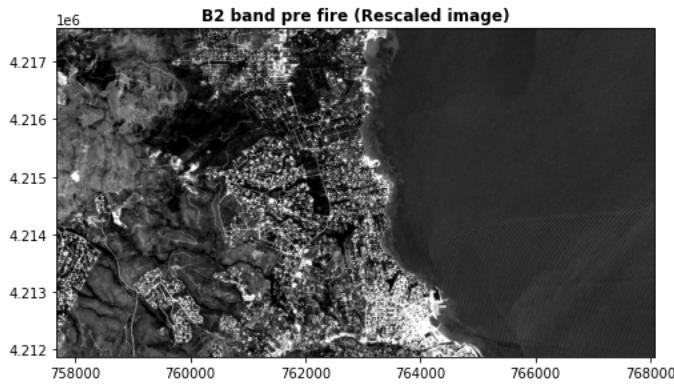
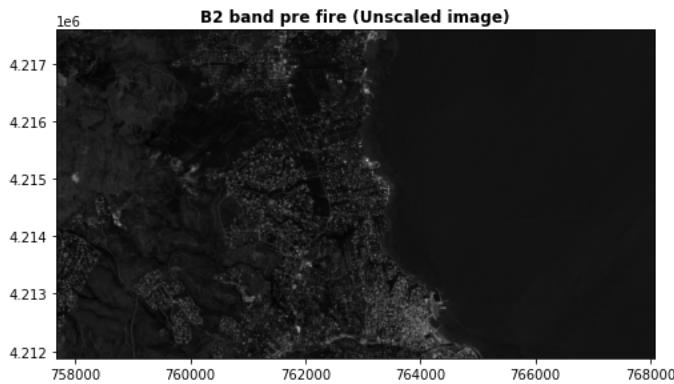
Contrast stretching

def contrast_stretching(image):
 p2, p98 = np.percentile(image, (2, 98))
 image_rescale = exposure.rescale_intensity(image, in_range=(p2, p98))
 return(image_rescale)

B2_pre_rescaled, B3_pre_rescaled, B4_pre_rescaled =[contrast_stretching(image) for image in [B2_pre, B3_pre, B4_pre]]

fig, (ax1, ax2) = plt.subplots(2,1, figsize=(12,10))
#show(img_rescale, ax=axr, cmap=plt.cm.gray, transform=dNBR_meta['transform'],)
map1 = show(B2_pre, ax=ax1, cmap=plt.cm.gray, title='B2 band pre fire (Unscaled image)', transform=dNBR_meta['transform'])
map2 = show(B2_pre_rescaled, ax=ax2, cmap=plt.cm.gray, title='B2 band pre fire (Rescaled image)', transform=dNBR_meta['transform'])

```



Μπορούμε να οπτικοποιήσουμε μια εικόνα *True Color* που είναι συνδυασμός των καναλιών *B4,B3,B2* (RGB Composite).

```
RGB image
Stack bands

def normalize(band):
 band_min, band_max = (band.min(), band.max())
 return ((band-band_min)/((band_max - band_min)))

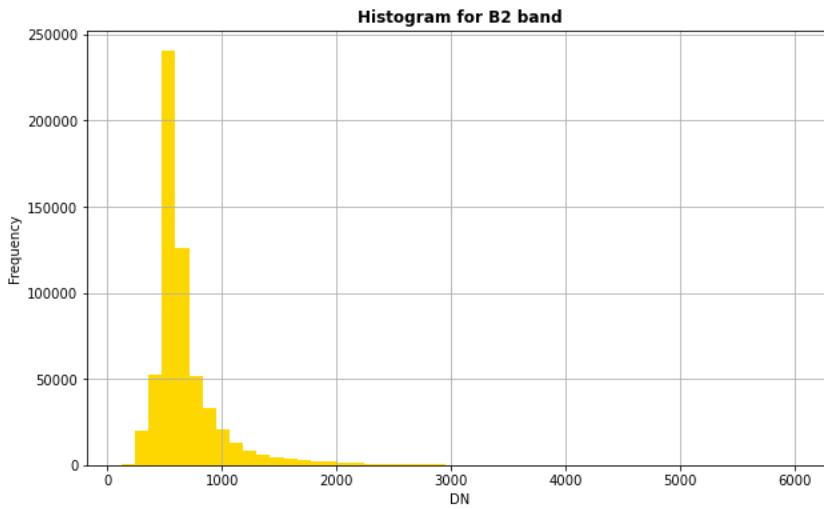
rgb = np.dstack((normalize(B4_pre), normalize(B3_pre), normalize(B2_pre)))

rgb = np.rollaxis(rgb, axis=2) # αναταξινόμηση των διαστάσεων ως [band, row, col],
απαιτούμενο για show function της rasterio
map1 =show(rgb ,title='B2 band pre fire (Rescaled image)',
transform=dNBR_meta['transform'])
```



Μέσω της συνάρτησης `show_hist` μπορούμε να δούμε το ιστόγραμμα συχνοτήτων για ένα κανάλι (ή και για όλα τα κανάλια ταυτόχρονα). Στην παρακάτω γραφμή κώδικα θα εκτυπώσουμε το σχετικό ιστόγραμμα συχνοτήτων για το κανάλι B2 (κανάλι με index 0).

```
fig, axhist = plt.subplots(1, 1, figsize=(10, 6))
show_hist(B2_pre, ax=axhist, bins=50, lw=0.0, histtype='stepfilled',
 title="Histogram for B2 band")
axhist.get_legend().remove()
plt.show()
```



Προσπέλαση των δεδομένων Sentinel-2 για μία ημερομηνία μετά την πυρκαγιά. Βάζουμε επίσης τα επιμέρους κανάλια σε ξεχωριστές μεταβλητές.

```
with rasterio.open(INPUTDIR / 'post_fire.tif') as src:
 image = src.read()
 B2_post,B3_post,B4_post,B8_post,B12_post = (image[i,:,:] for i in
range(0,src.count))
```

## Υπολογισμός δεικτών

Φτιάχνουμε μια συνάρτηση που υπολογίζει τον δείκτη **Normalized Burn Ratio (NBR)**. Περισσότερα για τον δείκτη εδώ. <https://www.usgs.gov/landsat-missions/landsat-normalized-burn-ratio>

```
def NBR(B8, B12):
 NBR = (B8.astype(float)-B12.astype(float)) /
(B8.astype(float)+B12.astype(float))
 return(NBR)
```

Υπολογίζουμε τον δείκτη **Normalized Burn Ratio** πριν και μετά την πυρκαγιά (**NBR\_pre**, **NBR\_post**) και την διαφορά αυτών (**dNBR**).

Χρησιμοποιούνται τα απαραίτητα κανάλια μέσω των σχετικών μεταβλητών που δημιουργήθηκαν πριν.

Στην συνέχεια χρησιμοποιείται ο δείκτης αυτός για να υπολογιστεί ένας πιο αξιόπιστος δείκτης, ο δείκτης **Relativized Burn Ratio (RBR)**.

```
Normalized Burn Ratio
NBR_pre = NBR(B8_pre, B12_pre)
NBR_post = NBR(B8_post, B12_post)

difference of NBR
dNBR = NBR_pre - NBR_post

#RBR
RBR = dNBR / (NBR_pre + 1.001)
```

Αποθήκευση του δείκτη RBR ως geotif

```
with rasterio.open(OUTDIR / 'RBR.tif', 'w', **dNBR_meta) as dst:
 dst.write(RBR.astype(rasterio.float64), 1)
```

Οι παραπάνω υπολογισμοί αφορούσαν μαθηματικές πράξεις πινάκων (**numpy arrays**). Επίσης τα αποτελέσματα τους είναι απλοί πίνακες. Όπως προαναφέρθηκε ο κώδικας **image[i,:,:]** που φτιάζαμε τις επιμέρους μεταβλητές (B8, B12 κτλ.) επιστρέφει **numpy arrays**. Ο δείκτης **RBR** είναι πίνακας **numpy array** ο οποίος πρέπει να επανασυνδεθεί με την συνοδευτική του χωρική του πληροφορία. Στην πράξη γράφουμε ένα αρχείο στην μνήμη με χωρικές παραμέτρους που ορίζονται στο αρχικό profile που έχουμε αποθηκεύσει στην μεταβλητή **dNBR\_meta**. Στην συνέχεια διαβάζουμε αυτό το αρχείο από την μνήμη και το βάζουμε σε ένα νέο **dataset object** με το όνομα **RBR\_rast**. Πλέον ο δείκτης **RBR** έχει χωρική πληροφορία.

```

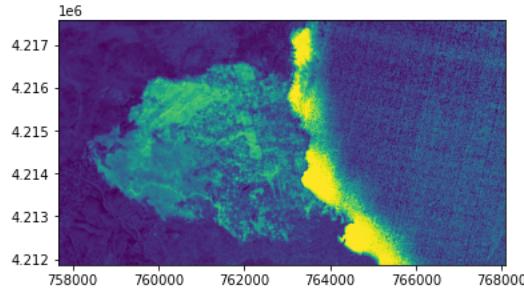
memfile = MemoryFile()

https://rasterio.readthedocs.io/en/stable/topics/writing.html
Register GDAL format drivers and configuration options with a
context manager.
with rasterio.Env():
 with MemoryFile() as memfile:
 with memfile.open(**dNBR_meta) as dst:
 dst.write(RBR.astype(rasterio.float64), 1)

 with memfile.open() as dataset: # Reopen as DatasetReader
 RBR_rast = dataset.read()
At the end of the ``with rasterio.Env()`` block, context
manager exits and all drivers are de-registered.

plot = show(contrast_stretching(RBR_rast[0,:,:]), cmap='viridis',
transform=dNBR_meta['transform'])

```



### Αλλαγή προβολικού συστήματος με την βιβλιοθήκη rasterio

Επειδή θα χρειαστεί στην συνέχεια να συνδυάσουμε τα δεδομένα raster (δείκτης `RBR`) με διανυσματικά δεδομένα που είναι στο προβολικό σύστημα `EΓΣΑ'87` πρέπει να μετασχήματίσουμε το αρχείο `RBR` σε `EΓΣΑ'87`. Τον δείκτη `RBR` τον έχουμε αποθηκεύσει παραπάνω ήδη ως `geotif` με τον πρωτογενές προβολικό σύστημα των δεδομένων Sentinel-2 (`WGS 84 / UTM zone 34N - EPSG:32634`). Μπορούμε να τον ανοίξουμε πάλι, να άλλάξουμε προβολικό σύστημα σε `EΓΣΑ'87` και να τον αποθηκεύσουμε σαν `geotif` επίσης.

```

dst_crs = "EPSG:2100" # web mercator(ie google maps)

with rasterio.open(OUTDIR / "RBR.tif") as src:

 # transform for input raster
 src_transform = src.transform

 # calculate the transform matrix for the output
 dst_transform, width, height = calculate_default_transform(
 src.crs, # source CRS
 dst_crs, # destination CRS
 src.width, # column count
 src.height, # row count
 *src.bounds, # unpacks outer boundaries (left, bottom, right, top)
)

 #print("Source Transform:\n", src_transform, '\n')
 #print("Destination Transform:\n", dst_transform)

 # set properties for output
 dst_kw_args = src.meta.copy()
 dst_kw_args.update(
 {
 "crs": dst_crs,
 "transform": dst_transform,
 "width": width,
 "height": height,
 "nodata": 0, # replace 0 with np.nan
 }
)

 with rasterio.open(OUTDIR / "RBR_2100.tif", "w", **dst_kw_args) as dst:
 # iterate through bands
 for i in range(1, src.count + 1):
 reproject(
 source=rasterio.band(src, i),
 destination=rasterio.band(dst, i),
 src_transform=src.transform,
 src_crs=src.crs,
 dst_transform=dst_transform,
 dst_crs=dst_crs,
 resampling=Resampling.nearest,
)

```

## Masking με vector δεδομένα

Το αρχείο του δείκτη RBR που έχουμε αποθηκεύσει σε ΕΓΣΑ87 πρέπει να το κάνουμε masking με τον διανυσματικό αρχείο των ΟΤΑ. Με αυτόν τον τρόπο διατηρούνται μόνο τα δεδομένα στην ξηρά και αποκλείεται η θάλασσα.

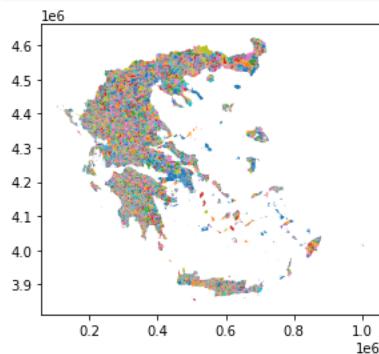
```

https://rasterio.readthedocs.io/en/stable/topics/masking-by-shapefile.html

Ανάγνωση δεδομένων vector (OTA)
OTA = gpd.read_file("./data/OTA/0adb0521-2223-43cd-96d3-d816ad7a193c.shp").to_crs(2100)
OTA.plot(column='NAME_LATIN')

```

<AxesSubplot:>



```

Ανάγνωση δείκτη RBR
with rasterio.open(OUTDIR / "RBR_2100.tif") as src:
 out_image, out_transform = rasterio.mask.mask(src, OTA.geometry, crop=False)
εφαρμόζουμε το mask με βάση τα πολύγωνα των OTA
 out_meta = src.meta # profile για εγγραφή δεδομένων

#
out_meta.update({"driver": "GTiff",
"height": out_image.shape[1],
"width": out_image.shape[2],
"nodata": 0, # ορίζουμε σαν nodata value = 0
"transform": out_transform
})

εγγραφή του masked αρχείου σαν geotif
with rasterio.open(OUTDIR / "RBR_2100.masked.tif", "w", **out_meta) as dest:
 dest.write(out_image)

```

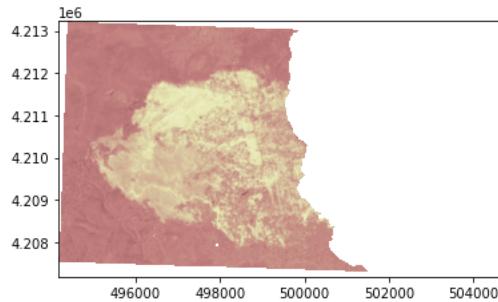
Επιβεβαίωση του masking μέσω οπτικοποίησης.

Κατά το masking που κάναμε με τους OTA όσο pixel δεν τέμνονται με αυτούς πήραν την τιμή 0 (όπως πχ τα pixels της θάλασσας). Για λόγους οπτικοποίησης θέτουμε τις τιμές αυτές σε κενές τιμές (`np.nan`) σε ένα αντιγράφο του αρχείου.

```

mask_image = np.copy(out_image)
mask_image[mask_image==0] = np.nan
plot = show(mask_image[0,:,:],cmap='pink', transform=out_meta['transform'])

```



## Επίπεδο σοβαρότητας (burn severity) καμένης περιοχής

Στην συνέχεια θα δημιουργηθεί ένας χάρτης σοβαρότητας (burn severity) καμένης περιοχής. Οι τιμές του δείκτη Relativized Burn Ratio θα κατηγοριοποιηθούν με βάση τον παρακάτω πίνακα:

Severity Level	dNBR Range (scaled by $10^3$ )
Enhanced Regrowth, high (post-fire)	-500 to -251
Enhanced Regrowth, low (post-fire)	-250 to -101
Unburned	-100 to +99
Low Severity	+100 to +269
Moderate-low Severity	+270 to +439
Moderate-high Severity	+440 to +659
High Severity	+660 to +1300

Burn severity classes and thresholds proposed by USGS. Color coding established by UN-SPIDER. Πήγα: <https://un-spider.org/advisory-support/recommended-practices/recommended-practice-burn-severity/burn-severity-earth-engine>.

```

bins = np.array([-np.Inf, -250, -100, 100, 270, 440, 660, np.Inf]) / 10**3 #
κατηγορίες
burn_severity = np.digitize(out_image, bins, right=False) # με αυτόν τον κώδικα
yίνεται η κατηγοριοποίηση

masking για λόγους οπτικοποίησης
burn_severity_masked = np.ma.masked_where(out_image == 0, burn_severity)
np.unique(burn_severity_masked)

```

```

masked_array(data=[1, 2, 3, 4, 5, 6, 7, --],
 mask=[False, False, False, False, False, False, True],
 fill_value=999999)

```

```

colors = [
 '#778835',
 '#a7c050',
 '#0be344',
 '#f8fc11',
 '#f8b140',
 '#f8671a',
 '#a600d4'
]

values = np.unique(np.ma.getdata(burn_severity_masked)) # επιστρέφει τις μοναδικές
τιμές κατηγοριοποίησης

χρώματα για κάθε κατηγορία
label=['Enhanced Regrowth, high (postfire)',
 'Enhanced Regrowth, low (postfire)',
 'Unburned',
 'Low Severity',
 'Moderate-low Severity',
 'Moderate-high Severity',
 'High Severity']

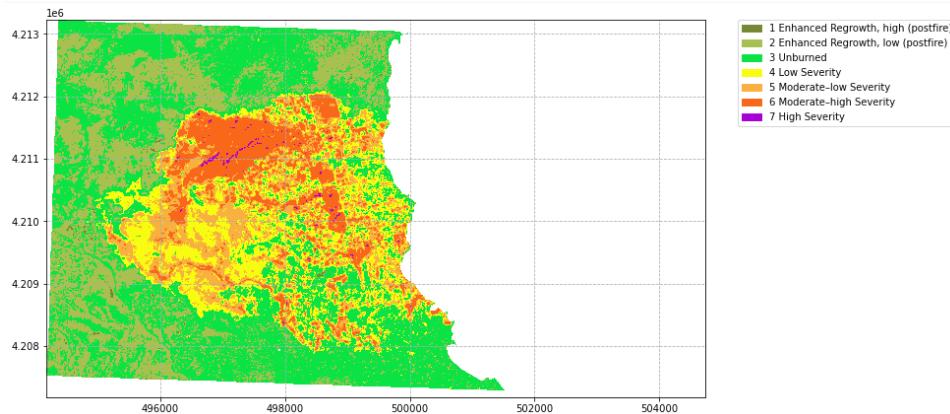
matplotlib patches,
https://matplotlib.org/stable/api/_as_gen/matplotlib.patches.Patch.html
τα πλαίσια χρωμάτων στο υπόμνημα
patches = [mpatches.Patch(color=colors[i], label="{l} {f}".format(l=values[i], f=label[i])) for i in range(len(values))]

plt.figure(figsize=(10,10))
fig, ax= plt.subplots(ncols=1, figsize=(12, 12))
put those patched as legend-handles into the legend
plt.legend(handles=patches, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
ax.grid(True, linestyle='--')
show(burn_severity_masked[0,:,:], ax=ax, transform=out_meta['transform'], cmap=
ListedColormap(colors))

```

<AxesSubplot:>

<Figure size 720x720 with 0 Axes>



Αντίστοιχα μπορούμε να φτιάξουμε έναν πίνακα με την κατανομή ανά κατηγορία σοβαρότητας.

```

unique, counts = np.unique(burn_severity_masked, return_counts=True) # πίνακας
κατανομής συχνοτήτων

μετατροπή σε pandas dataframe
φιλτράρουμε εκτός τα masked data με ~unique.mask
df = pd.DataFrame(data={'Severity category':unique.data[~unique.mask] ,
'Label':label, 'Number of Pixels': counts[~unique.mask]})

υπολογισμός ποσοστιαίας κατανομής
df['% per category'] = round((df['Number of Pixels'] / df['Number of
Pixels'].sum()) * 100,2)

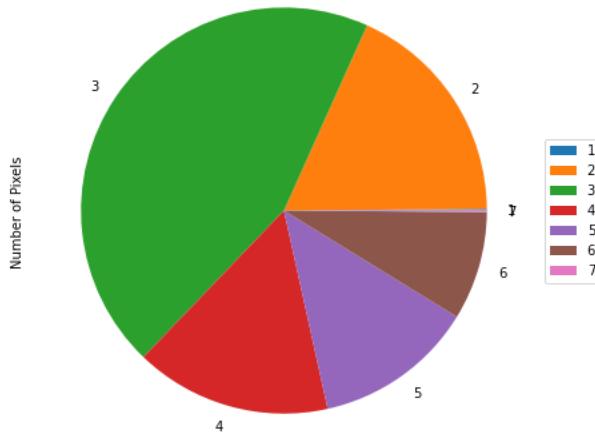
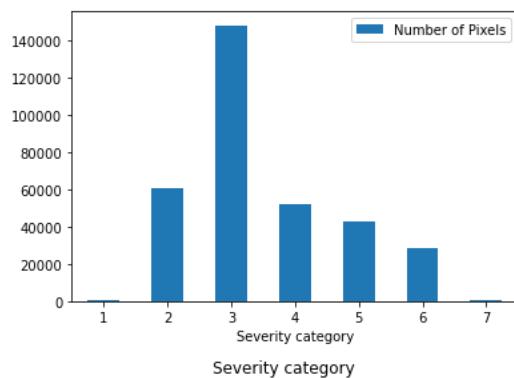
ορισμός στήλης 'Severity category' ως index, χρήσιμο στο επόμενο στάδιο της
οπτικοποίησης
df.set_index('Severity category', inplace=True)
df

```

Severity category	Label	Number of Pixels	% per category
1	Enhanced Regrowth, high (postfire)	418	0.13
2	Enhanced Regrowth, low (postfire)	60314	18.15
3	Unburned	148009	44.55
4	Low Severity	51941	15.63
5	Moderate-low Severity	42423	12.77
6	Moderate-high Severity	28763	8.66
7	High Severity	372	0.11

Δημιουργούμε ιστόγραμμα συχνοτήτων ή εναλλακτικά διάγραμμα πίτα για την ποσοστιαία κατανομή ανά κατηγορία σοβαρότητας.

```
ax = df.plot.bar(y='Number of Pixels', rot=0)
plot = df.plot.pie(y='Number of Pixels', figsize=(7, 7), title='Severity category').legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
```



## Ομαδοποίηση της έκταση σε καμένη έκταση και σε μη καμένη με τον αλγόριθμο K-means

Αρχικά πρέπει να μετασχηματίσουμε τα δεδομένα μας στην μορφή με μία μόνο στήλη δηλ οντανούν shape (rows, 1):

```
X = out_image.reshape((-1, 1))
X.shape
```

```
(640090, 1)
```

Στην συνέχεια παραμετροποιούμε και τρέχουμε τον αλγόριθμο KMeans από την υπορουτίνα cluster της βιβλιοθήκης scikit-learn. Οι παράμετροι έχουν ως εξής:

- n\_clusters, είναι ο αριθμός των ομάδων που θα ομαδοποιήσει τα δεδομένα ο αλγόριθμος. Δύο στην περίπτωσή μας καμένη/μη καμένη γη.

- **random\_state**, ένας ακέραιος που ελέγχει την τυχαιότητα του αριθμού κατά την αρχικοποίηση του κέντρου καθώς και την αναπαραγωγησιμότητα του αλγορίθμου.
- **max\_iter**, μέγιστος αριθμός επαναλήψεων του αλγορίθμου.

```
βλ. παράδειγμα: # https://www.acgeospatial.co.uk/k-means-sentinel-2-python/
k_means = cluster.KMeans(n_clusters=2, random_state=31415, max_iter=30)
k_means.fit(X) # υπολογισμός k-means clustering στα δεδομένα RBR.

X_cluster = k_means.labels_ # τα labels_ είναι η κατηγοριοποίηση σε κατηγορίες, 1
και 0, καμένη και μη καμένη γη αντίστοιχα.
#X_cluster = X_cluster.reshape(img.shape)

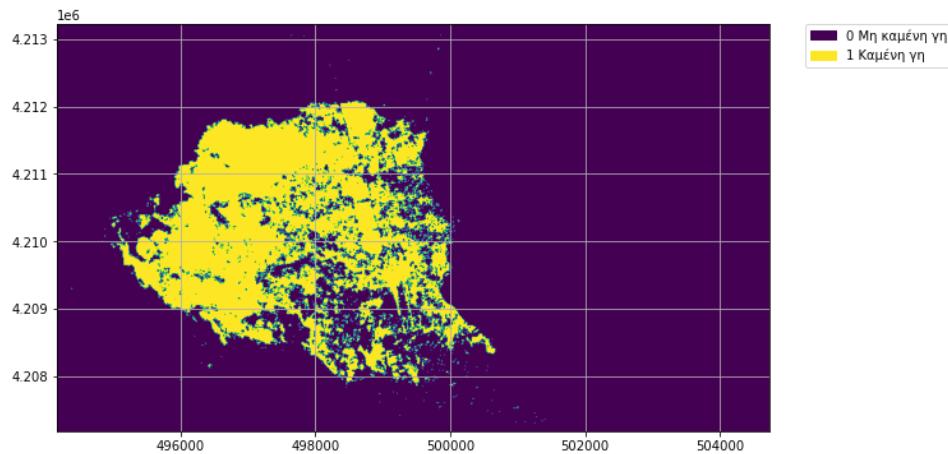
για να εγγράψουμε το αποτέλεσμα σε geotif κάνουμε reshape του πίνακα στις
αναγκαίες διαστάσεις.
X_cluster = X_cluster.reshape(out_image[0,:,:].shape)

εγγραφή σε geotif
out_meta.update({'dtype': rasterio.uint8})
with rasterio.open(OUTDIR / 'RBR_kmeans_scikit.tif', 'w', **out_meta) as dst:
 dst.write(X_cluster.astype(rasterio.uint8), 1)

values = np.unique(X_cluster.ravel())
plt.figure(figsize=(10,10))
im = plt.imshow(X_cluster, interpolation='none')

get the colors of the values, according to the
colormap used by imshow

colors = [im.cmap(im.norm(value)) for value in values]
create a patch (proxy artist) for every color
label=[["Μη καμένη γη", "Καμένη γη"]]
patches = [mpatches.Patch(color=colors[i], label="{l} {f}".format(l=values[i], f=
label[i])) for i in range(len(values))]
put those patched as legend-handles into the legend
plt.legend(handles=patches, bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.grid()
#plt.show() # αν καλέσουμε την μέθοδο show του matplotlib δεν θα εχει ορθή
γεωγραφική αναφορά ο χάρτης
καλούμε την συνάρτηση show από την βιβλιοθήκη του rasterio.plot
show(X_cluster, transform=out_meta['transform'])
```



<AxesSubplot:>

## Μετατροπή δεδομένων raster σε vector

```

mask = None
with rasterio.Env():
 with rasterio.open(OUTDIR / 'RBR_kmeans_sciki.tif') as src:
 image = src.read(1) # first band
 crs = src.profile['crs']
 t = src.profile['transform']
 results = (
 {'properties': {'raster_val': v}, 'geometry': s} #
 for i, (s, v) in enumerate(features.shapes(image, mask=mask,
 transform=src.transform)))
 geoms = list(results)

```

Δημιουργία geopandas dataframe (`kmeans_vector`) με την ομαδοποίηση σε καμένη και μη καμένη γη. Η στήλη `raster_val` λαμβάνει τιμές `1=καμένη` και `0=μη καμένη γη`. Ακολουθεί μετατροπή της στήλης `raster_val` σε ακέραιο τύπο δεδομένων οπτικοποίηση των 10 πρώτων γραμμών του πίνακα.

```
kmeans_vector = gpd.GeoDataFrame.from_features(geoms, crs=CRS.to_epsg(crs))
```

```
kmeans_vector['raster_val'] = kmeans_vector['raster_val'].astype('int')
kmeans_vector.head(10)
```

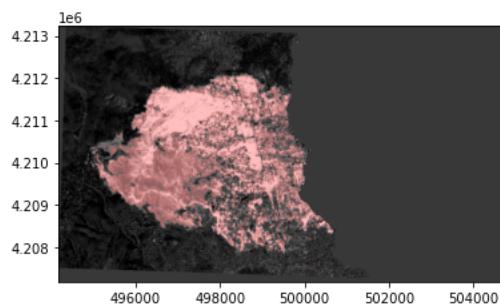
	geometry	raster_val
0	POLYGON ((497642.124 4213049.519, 497642.124 4...	1
1	POLYGON ((499850.255 4213049.519, 499850.255 4...	1
2	POLYGON ((497821.972 4213039.527, 497821.972 4...	1
3	POLYGON ((499820.280 4212859.679, 499830.272 4...	1
4	POLYGON ((498491.405 4212679.832, 498491.405 4...	1
5	POLYGON ((498211.642 4212609.891, 498211.642 4...	1
6	POLYGON ((499700.382 4212509.975, 499700.382 4...	1
7	POLYGON ((499710.373 4212489.992, 499710.373 4...	1
8	POLYGON ((498291.574 4212410.060, 498291.574 4...	1
9	POLYGON ((498301.566 4212240.204, 498301.566 4...	1

Φιλτράρουμε τον πίνακα `kmeans_vector` και κρατάμε μόνο την καμένη γη (`'raster_val == 1'`)

```
burnt_area = kmeans_vector.query('raster_val == 1')
```

οπτικοποίηση σε χάρτη

```
_, ax = plt.subplots()
burnt_area.plot(ax=ax, color='red', alpha=.25) ## alpha είναι ο βαθμός διαφάνειας
plot.show(contrast_stretching(out_image), transform=t, ax=ax, cmap='gray')
```



Αποθήκευση του αρχείου σε shapefile.

```
kmeans_vector.to_file(OUTDIR / 'kmeans.shp')
```

```
/home/leonidas/anaconda3/envs/book/lib/python3.10/site-
packages/geopandas/io/file.py:362: FutureWarning: pandas.Int64Index is deprecated
and will be removed from pandas in a future version. Use pandas.Index with the
appropriate dtype instead.
pd.Int64Index,
```

Στην συνέχεια θα ακολουθήσει ο υπολογισμός της καμένης δομημένης επιφάνειας. Αρχικά διαβάζουμε το αρχικό shapefile και ελέγχουμε το προβολικό του σύστημα πρέπει να είναι στο [ΕΓΣΑ'87](#) για να είναι συγκρίσιμο το αρχείο με τα δεδομένα της ομαδοποίησης που έχουμε κάνει.

```
builtup = gpd.read_file(INPUTDIR / "built_up.shp")
builtup.crs
```

```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

Επειδή το αρχείο της δομημένης επιφάνειας είναι στο [WGS'84](#) κάνουμε μετασχηματισμό σε [ΕΓΣΑ'87](#). Και συνοπτική διερεύνηση των περιγραφικών δεδομένων του αρχείο.

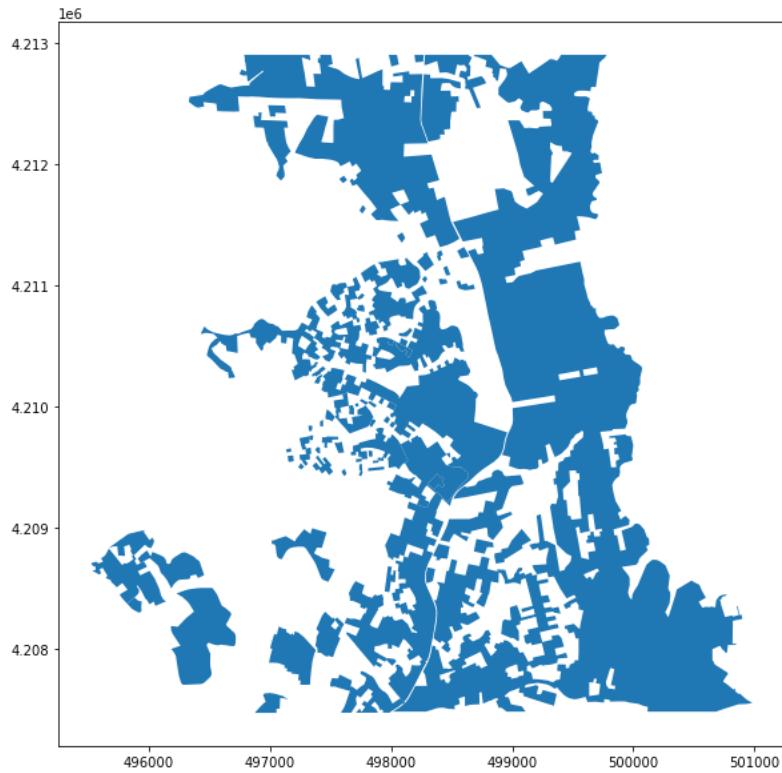
```
builtup = builtup.to_crs(2100) # μετασχηματισμός σε ΕΓΣΑ87.
builtup.head()
```

	obj_type	name	info	det_method	notation	or_src_id	dmg_src_id	cd_\
0	11-Residential Buildings	Unknown	997-Not Applicable	Not Applicable	Built up area	997	997	Applic
1	11-Residential Buildings	Unknown	997-Not Applicable	Not Applicable	Built up area	997	997	Applic
2	11-Residential Buildings	Unknown	997-Not Applicable	Not Applicable	Built up area	997	997	Applic
3	11-Residential Buildings	Unknown	997-Not Applicable	Not Applicable	Built up area	997	997	Applic
4	11-Residential Buildings	Unknown	997-Not Applicable	Not Applicable	Built up area	997	997	Applic

Και ακολουθεί οπτικοποίηση της γεωγραφικής πληροφορίας της δομημένης επιφάνειας.

```
builtup.plot(figsize= (10,10))
```

```
<AxesSubplot:>
```



Για να εντοπίσουμε την καμένη δομημένη επιφάνεια πρέπει να πάρουμε την τομή (intersection) των δύο αρχείων (του αρχείου με τα πολύγωνα της δομημένης επιφάνειας και του αρχείου με τα πολύγωνα καμένης/μη καμένης γης που προέκυψαν από τον αλγόριθμο k-means).

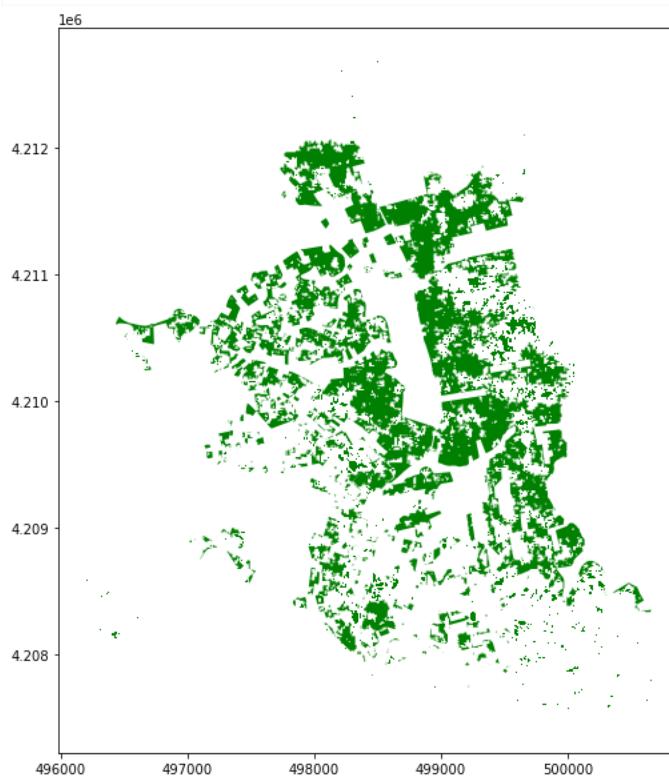
(Δείτε περισσότερα στο [Set-Operations with Overlay](#) της βιβλιοθήκης geopandas.)

```
burn_buildup_area = gpd.overlay(builtup, burnt_area, how='intersection') #
υπολογισμός τομής των δύο αρχείων
```

Οπτικοποίηση του αποτελεσματος και αποθήκευση σε geopackage.

```
burn_buildup_area.plot(figsize= (10,10), color="green")
burn_buildup_area.to_file(OUTDIR / 'burn_buildup_area.gpkg', driver='GPKG',
layer='name')
```

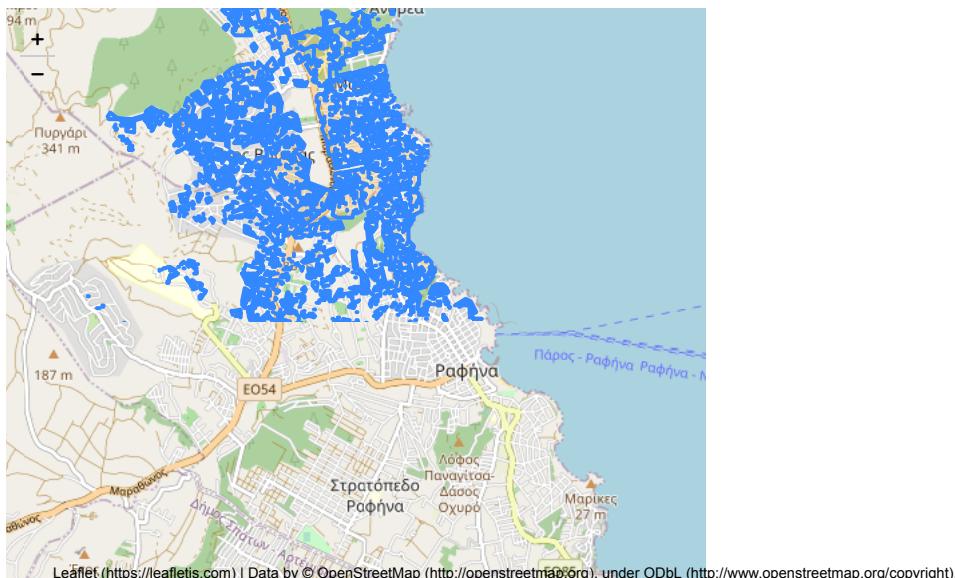
```
/home/leonidas/anaconda3/envs/book/lib/python3.10/site-
packages/geopandas/io/file.py:362: FutureWarning: pandas.Int64Index is deprecated
and will be removed from pandas in a future version. Use pandas.Index with the
appropriate dtype instead.
pd.Int64Index,
```



Η οπτικοποίηση του αποτελέσματος σε διαδραστικό χάρτη είναι εφικτή μέσω της βιβλιοθήκης folium.

```
folium_map = folium.Map(location=[38.0425, 23.9785], zoom_start=13,
tiles='OpenStreetMap') # CartoDB positron

for _, r in burn_buildup_area.to_crs(4326).iterrows():
 # Without simplifying the representation of each borough,
 # the map might not be displayed
 sim_geo = gpd.GeoSeries(r['geometry'])
 geo_j = sim_geo.to_json()
 geo_j = folium.GeoJson(data=geo_j,
 style_function=lambda x: {'fillColor': 'orange'})
 folium.Popup(r['obj_type']).add_to(geo_j)
 geo_j.add_to(folium_map)
folium_map
```



## Υπολογισμός καμένης δομημένης επιφάνειας

Στην συνέχεια υπολογίζεται η συνολική καμένη επιφάνεια σε στρέμματα. Για κάθε ένα πολύγωνο υπολογίζουμε την έκταση του. Το αποτέλεσμα του υπολογισμού είναι  $\text{m}^2$  το οποίο το μετατρέπουμε σε στρέμματα (1 στρ.= 1000  $\text{m}^2$ ). Στην συνέχεια παίρνουμε το άθροισμά τους και εκτυπώνουμε το αποτέλεσμα

```
A method
burn_buildup_area["area"] = burn_buildup_area['geometry'].area / 10**3 # τμ σε στρέμματα
b_area = burn_buildup_area["area"].sum()
print(f"Η έκταση της καμένης δομημένης επιφάνειας είναι {round(b_area,2)} στρ.")
```

Η έκταση της καμένης δομημένης επιφάνειας είναι 3054.92 στρ.

Εναλλακτικά μπορούμε να ενώσουμε όλα τα πολύγωνα και να πάρουμε την έκταση ενός ενιαίου πολυγώνου. Η ένωση των πολυγώνων γίνεται μέσω της μεθόδου `dissolve`.

```
B method
burn_buildup_area_dis = burn_buildup_area.dissolve("raster_val") #
https://geopandas.org/en/stable/docs/reference/api/geopandas.GeoDataFrame.dissolve.html#geopandas.GeoDataFrame.dissolve
b_area = float(burn_buildup_area_dis.geometry.area)/10**3
print(f"Η έκταση της καμένης δομημένης επιφάνειας είναι {round(b_area,2)} στρ.")
```

Η έκταση της καμένης δομημένης επιφάνειας είναι 3054.92 στρ.

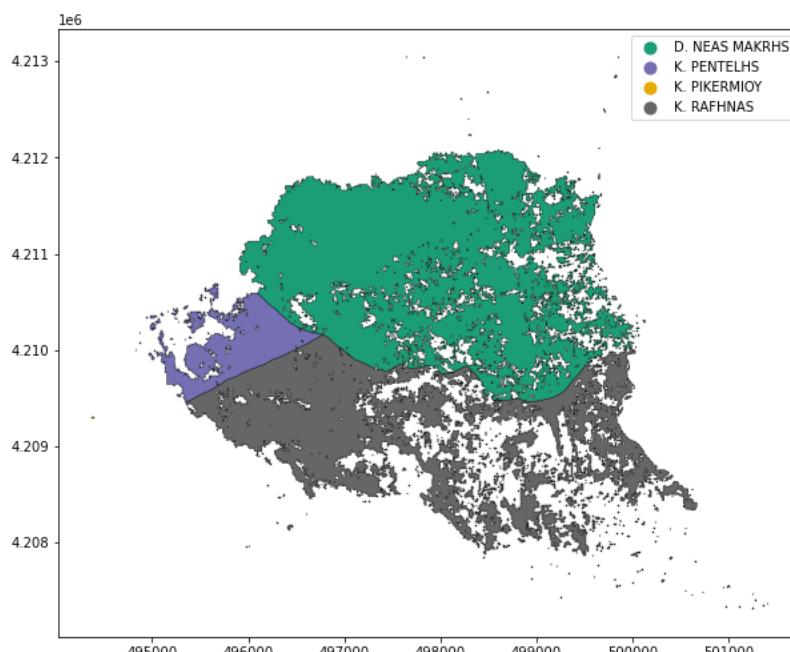
## Υπολογισμός καμένης έκτασης ανά OTA

Επιπλέον είναι εφικτό να διαπιστώσουμε πόση έκταση έχει καεί ανά OTA. Όπως είδαμε σε προηγούμενο βήμα κατά την οπτικοποίηση του αρχείου των OTAs, αυτό περιλαμβάνει το σύνολο της Ελλάδας. Βρίσκουμε την τομή των αρχείων OTAs και του ομαδοποιημένου με k-means αρχείου. Στην συνέχεια λαμβάνουμε την έκταση της κάθε γεωμετρίας σε μια νέα στήλη (`area`).

```
burn_OTA = gpd.overlay(OTA, burnt_area, how='intersection')
burn_OTA['area'] = burn_OTA.geometry.area
```

Όπως βλέπουμε το αποτέλεσμα πλέον περιορίζεται στην καμένη περιοχή. Έχει γίνει κατηγοριοποίηση των καμένων περιοχών με βάση των OTAs στον οποίο ανήκουν.

```
burn_OTA.plot(figsize= (10,10),cmap='Dark2', column='NAME_LATIN', legend=True,
edgecolor="black",antialiased=True, linewidth=0.4)
plt.savefig(OUTDIR / 'burn_OTA_area_map.png') #save as png
```



Για να υπολογίσουμε την καμένη έκταση ανά ΟΤΑ κάνουμε ομάδοποίηση με βάση την τιμή του πεδίου **NAME\_LATIN** και υπολογίζουμε το άθροισμα της στήλης **area**. Κάνουμε αναγωγή του αποτελέσματος σε στρέμματα.

```
Group by cell_id and croptype and calculate total intersection area
burn_OTA_area = burn_OTA.groupby(['NAME_LATIN'])['area'].sum().reset_index()
burn_OTA_area['area'] = burn_OTA_area['area'] / 10**3 # τμ σε στρέμματα
```

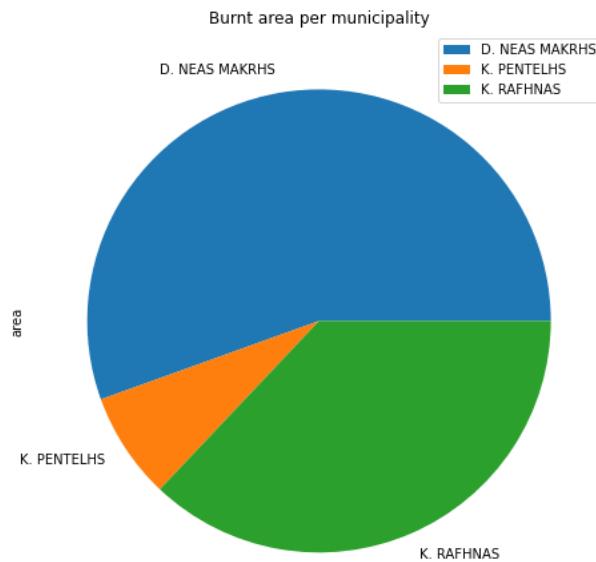
```
burn_OTA_area = burn_OTA_area.set_index('NAME_LATIN')
burn_OTA_area
```

area	
NAME_LATIN	
D. NEAS MAKRHS	5809.633000
K. PENTELHS	780.402739
K. PIKERMIOY	0.399324
K. RAFHNAS	3877.550790

Τέλος οπτικοποιούμε την αναλογία της καμένης γης σε ένα διάγραμμα/πίτα (pie chart). Επειδή η καμένη έκταση στην κοινότητα Πικερμίου είναι πολύ μικρή, φιλτράρουμε τα δεδομένα και κρατάμε μόνο έκτασεις > 500 στρ.

```
burn_OTA_area = burn_OTA_area[burn_OTA_area['area']>500]

burn_OTA_area.plot(kind='pie', y='area', x='NAME_LATIN', figsize=(8,8), title='Burnt
area per municipality')
plt.savefig(OUTDIR / 'burn_OTA_area.png')
```



## Ευχαριστίες

Ευχαριστώ τον [Βασίλη Λέτσιο](#) για την κοινοποίηση της μεθοδολογίας που ανέπτυξε στα πλαίσια του μαθήματος [«ΕΙΔΙΚΕΣ ΕΦΑΡΜΟΓΕΣ ΤΗΛΕΠΙΣΚΟΠΗΣΗΣ»](#) για τον υπολογισμό των δεικτών καμένης γης με δεδομένα Sentinel-2.

## Βιβλιογραφία - Πηγές δεδομένων

Parks, S., Dillon, G., Miller, C., 2014. A New Metric for Quantifying Burn Severity: The Relativized Burn Ratio. *Remote Sensing* 6, 1827–1844.  
<https://doi.org/10.3390/rs6031827>

