



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Αρχές Γλωσσών Προγραμματισμού και Μεταφραστές

Αναφορά Project Python

Όνομα προγράμματος: [Document_Similarity.py](#)

Παρακάτω παρατίθενται αριθμημένες οι εικόνες από τον κώδικά μου καθώς και αναλυτικός σχολιασμός κάθε μιας. Έπειτα υπάρχουν εικόνες με τον τρόπο λειτουργίας της εφαρμογής.

Εικόνες Κώδικα:

Εικόνα 1:

Imports, initializations και ερώτηση στον χρήστη που ζητά τον αριθμό των Documents.
Απαντήσεις μικρότερες ή ίσες του ένα δεν γίνονται δεκτές και η ερώτηση επαναλαμβάνεται.

```
1  from collections import Counter
2  from itertools import chain
3  import math
4  import re
5
6  N = int(input("Amount of documents: "))
7  while N <= 1:
8      print("Documents must be two or more")
9      N = int(input("Amount of documents: "))
10
11  listsOne = ["Null"] * N
12  listTwo = list()
13  dictToList = list()
14  digitList = list()
15  resultList = list()
16  finalList = list()
17  topList = list()
18  printingList = list()
19  endingList = list()
20  part1 = 0
21  part2 = 0
22  part3 = 0
23  cos = 0
24  tempMax = 0
```

Εικόνα 2:

Εισαγωγή των Documents από τον χρήστη, αντικατάσταση όλων των “-” με “ ” και αποδοχή μόνο των αλφαριθμητικών a-z, A-Z και 0-9. Ένα κενό Document δεν γίνεται αποδεκτό. Αν το Document γίνει αποδεκτό εμφανίζει μήνυμα “Document No. x added successfully”. Ακόμα, για την ομοιότητα των Documents μεταξύ τους όλα τα κεφαλαία γίνονται πεζά. Όλα αυτά αποθηκεύονται σε μια λίστα μέσα στη nested list “listsOne”. Τέλος, δημιουργώ μια ακόμα λίστα τη “listTwo” η οποία είναι ίδια με την “listsOne” αλλά όχι nested, με σκοπό μέσω της “counter” να μετρήσουμε πόσες φορές εμφανίζονται οι λέξεις σε όλες τις λίστες της “listsOne” συνολικά και να αποθηκεύσουμε το αποτέλεσμα μας στο countsAll.

```
26 for x in range(N):
27     print("Document No.", x + 1)
28     document = input("Enter your document here:\n")
29     document = re.sub(r'[-]', ' ', document)
30     document = re.sub('[^a-zA-Z0-9 ]', '', document)
31     listsOne[x] = document.split()
32     while len(listsOne[x]) == 0:
33         print("Document can't be empty")
34         print("Document No.", x + 1)
35         document = input("Enter your document here:\n")
36         listsOne[x] = document.split()
37     print("Document No.", x + 1, "added successfully\n")
38     listsOne = [[word.lower() for word in line] for line in listsOne]
39
40 listTwo = list(chain.from_iterable(listsOne))
41 countsAll = Counter(listTwo)
```

Εικόνα 3:

Σε αυτό το κομμάτι του κώδικα αποθηκεύω τον “Counter” κάθε λίστας στην μεταβλητή “countsOne”. Έπειτα αφαιρώ από το “countsAll” το “countsOne” και αν κάποια λέξη υπάρχει στο “countsAll” και όχι στο “countsOne” την προσθέτω στο “countsOne” με value ίσο με μηδέν. Στην Python το “counter” λειτουργεί βασισμένο στα Dictionaries, οπότε το “countsAll” και το “countsOne” είναι σε μορφή Dictionary. Για παράδειγμα, αν προσθέσω τα Documents:

“I am Nikos Bakalis” και “I am Nikolas Bakalis”

τα αποτελέσματα των “countsAll” και “countsOne” θα είναι τα αντίστοιχα:

countsAll:

```
Counter({'i': 2, 'am': 2, 'bakalis': 2, 'nikos': 1, 'nikolas': 1})
```

countsOne:

```
Counter({'i': 1, 'am': 1, 'nikos': 1, 'bakalis': 1, 'nikolas': 0})
```

```
Counter({'i': 1, 'am': 1, 'nikolas': 1, 'bakalis': 1, 'nikos': 0})
```

Με αυτό δεδομένο χρησιμοποιώ τη μεταβλητή “temp” για να φτιάξω τη λίστα “dictToList” την οποία έπειτα κάνω και sort. Αφού την κάνω sort, περνάω όλες τις τιμές της μεταβλητής “dictToList” στην ήδη υπάρχουσα nested λίστα “listsOne”. Τέλος προσθέτω στο “countsAll” το “countsOne” που αφαίρεσα στην αρχή.

```
43 for y in range(len(listsOne)):
44     countsOne = Counter(listsOne[y])
45     countsAll.subtract(countsOne)
46     for word in list(countsAll):
47         if word in countsAll:
48             if word not in countsOne:
49                 countsOne.update({word: 0})
50     for key, value in countsOne.items():
51         temp = [key, value]
52         dictToList.append(temp)
53     dictToList.sort()
54     tempForLists = dictToList
55     dictToList = listsOne[y]
56     listsOne[y] = tempForLists
57     dictToList.clear()
58     countsAll = countsAll + countsOne
```

Εικόνα 4:

Εδώ χρησιμοποιώ διπλή for με σκοπό να πάρω μόνο τις τιμές value της λίστας “listsOne” και τις αποθηκεύω στην μεταβλητή digitList. Τέλος, χωρίζω τη “digitList” σε Chunks με όνομα “digitChunks”, ο διαχωρισμός της λίστας ισοδυναμεί με το μέγεθος της “digitList” διά τον αριθμό “N” των Documents που εισάγει ο χρήστης.

```
60 for lists in range(N):
61     for micro in range(len(listsOne[lists])):
62         for value in range(len(listsOne[lists][micro])):
63             digitList.append(listsOne[lists][micro][value])
64         break
65
66 digitChunks = [digitList[x:x+(int(len(digitList)/N))] for x in range(0, len(digitList), int(len(digitList)/N))]
```

Εικόνα 5:

Έπειτα, παίρνω τους αποθηκευμένους αριθμούς στην “digitChunks” και τους εφαρμόζω στον τύπο της “cos” που μας δίνεται. Η διαδικασία αυτή γίνεται με τρία βήματα, και τελικά καταλήγω σε δύο λίστες, την “resultList” και την “finalList”. Η “resultList” περιέχει τους αριθμούς των Documents που εξετάζω και το ποσοστό που βγαίνει από τον τύπο, ενώ η “finalList” περιέχει μόνο το ποσοστό του τύπου. Τέλος, ορίζω ξανά τις τιμές που άλλαξαν λόγω του τύπου ίσες με μηδέν. Πρέπει να σημειωθεί ότι οι δύο πρώτες “for” γίνονται βασισμένες στη θεωρία των συνδυασμών, καθώς ποτέ δεν εξετάζουν το ίδιο Document και ποτέ δεν θα εξεταστούν τα ίδια δύο Documents παραπάνω από μια φορά.

π.χ. Δεν θα εξεταστεί ποτέ η ομοιότητα του Document 1 με τον εαυτό του, ούτε θα εξεταστεί πρώτα η ομοιότητα του Document 1 με το Document 2 και μετά του Document 2 με το Document 1.

```
68 for i in range(len(digitChunks)):
69     for j in range(i + 1, len(digitChunks)):
70         for allDigits in range(len(digitChunks[i])):
71             part1 = float(part1 + (digitChunks[i][allDigits] * digitChunks[j][allDigits]))
72             part2 = float(part2 + (digitChunks[i][allDigits] * digitChunks[i][allDigits]))
73             part3 = float(part3 + (digitChunks[j][allDigits] * digitChunks[j][allDigits]))
74         cos = float(part1 / (math.sqrt(part2) * math.sqrt(part3)))
75         resultList.append(i + 1)
76         resultList.append(j + 1)
77         resultList.append(cos)
78         finalList.append(cos)
79         part1 = 0
80         part2 = 0
81         part3 = 0
82         cos = 0
```

Εικόνα 6:

Αν ο αριθμός των Documents που εισήγαγε ο χρήστης είναι ίσος με δύο τυπώνει το μήνυμα “The similarity between Document No. 1 and Document No. 2 is: x%”

```
86     if N == 2:
87         print("The similarity between Document No: 1 and Document No: 2 is:", round((finalList[0] * 100), 2), "%")
```

Εικόνα 7:

Αλλιώς αν τα Documents είναι περισσότερα από δυο, σπάει την λίστα “resultList” σε chunks ανά τρία, καθώς τρεις είναι οι αριθμοί που αναφέρονται στο ποσοστό ομοιότητας κάθε δύο Documents. Ο αριθμός του πρώτου Document, ο αριθμός του δεύτερου και το ποσοστό ομοιότητάς τους. Έπειτα, για να τυπωθεί το κάθε ποσοστό μεταξύ όλων των Documents χρησιμοποιεί της δυο “for” ώστε να διαβάσει την “resultChunks” και να αντιστοιχίσει κάθε στοιχείο της εκεί που πρέπει.

```
88     else:
89         resultChunks = [resultList[x:x+3] for x in range(0, len(resultList), 3)]
90         for allResults in range(len(resultChunks)):
91             for allNumbers in range(2, len(resultChunks[allResults]), 3):
92                 print("The similarity between Document No:", resultChunks[allResults][0], "and Document No:",
93                       resultChunks[allResults][1], "is:", round((resultChunks[allResults][2] * 100), 2), "%")
```

Εικόνα 8:

Έπειτα, μέσα από την πράξη των συνδυασμών υπολογίζει πόσους συνδυασμούς μπορούν να δώσουν τα Documents που εισήγαγε ο χρήστης. Και τυπώνει την ανάλογη ερώτηση. Αν βάλεις διαφορετικό αριθμό η ερώτηση επαναλαμβάνεται. Το αποδεκτό αποτέλεσμα που θα εισάγει ο χρήστης αποθηκεύεται στην μεταβλητή K.

```
94     print("\n\nEnter a Number between 1 and", int(math.factorial(N) / (math.factorial(2) * math.factorial(N - 2))))
95     K = int(input("Find the top similar documents: "))
96     while K > math.factorial(N) / (math.factorial(2) * math.factorial(N - 2)):
97         print("The number must be between 1 and", int(math.factorial(N) / (math.factorial(2) * math.factorial(N - 2))))
98         K = int(input("Find the top similar documents: "))
```

Εικόνα 9:

Στη συνέχεια, το πρόγραμμα διαβάζει την “finalList” και την κάνει sort από το μεγαλύτερο στοιχείο στο μικρότερο σε μια νέα λίστα με όνομα “topList”.

```
99     for i in range(K):
100         for allResults in range(len(finalList)):
101             if tempMax < finalList[allResults]:
102                 tempMax = finalList[allResults]
103         finalList.remove(tempMax)
104         topList.append(tempMax)
105         tempMax = 0
```

Εικόνα 10:

Έπειτα θέλοντας να ελέγξουμε την ομοιότητα των ποσοστών που δίνει η “topList” και η “resultChunks” χρησιμοποιούμε τρεις “for” για να τις διαβάσουμε και περνάμε τα όμοια ποσοστά στην “printingList” μαζί με τον αριθμό των αντίστοιχων Documents.

```
104     for i in range(len(topList)):
105         for j in range(len(resultChunks)):
106             for allNumbers in range(len(resultChunks[j])):
107                 if topList[i] == resultChunks[j][2]:
108                     printingList.append(round((topList[i] * 100), 2))
109                     printingList.append(resultChunks[j][0])
110                     printingList.append(resultChunks[j][1])
111                     break
```

Εικόνα 11:

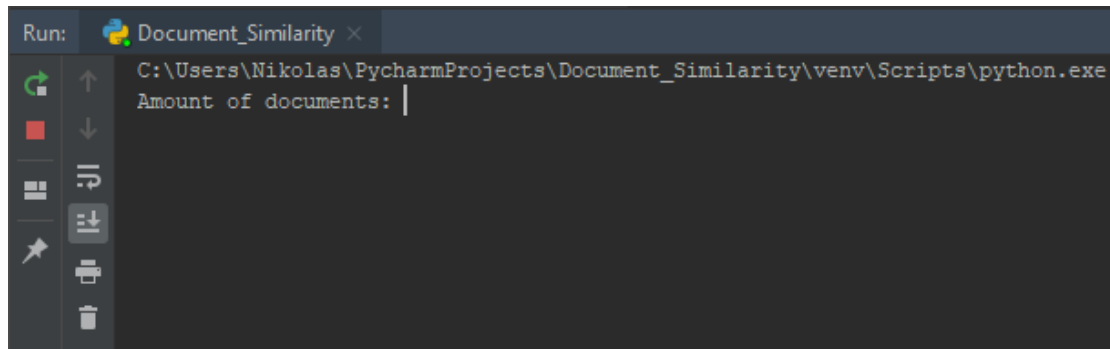
Τέλος, χωρίζουμε την “printingList” σε “printingChunks” βασισμένοι στην ίδια λογική όπως και στην “resultList” και στα “resultChunks” με τα τρία δεδομένα, δύο για τους αριθμούς των Documents και ένα για το ποσοστό ομοιότητάς τους. Διαβάζουμε την “printingChunks” και κάθε στοιχείο της που δεν υπάρχει στην “endingList” (ώστε να μην περνάει μόνο το πρώτο στοιχείο) το προσθέτουμε στην “endingList”. Τελικά, το πρόγραμμα τυπώνει τα Top K Similar Documents που παίρνει από την “endingList”.

```
112     printingChunks = [printingList[x:x + 3] for x in range(0, len(printingList), 3)]
113     for allNumbers in range(len(printingChunks)):
114         if printingChunks[allNumbers] not in endingList:
115             endingList.append(printingChunks[allNumbers])
116     print("\n\n\n")
117     for i in range(K):
118         print(i + 1, " The", endingList[i][0], "% similarity, come from document No:",
119               endingList[i][1], "and Document No:", endingList[i][2])
```

Λειτουργία Κώδικα:

Εικόνα 1:

Με το που ξεκινάει το πρόγραμμα ζητάει από τον χρήστη να εισάγει τον αριθμό των Documents που θα εισάγει αργότερα.



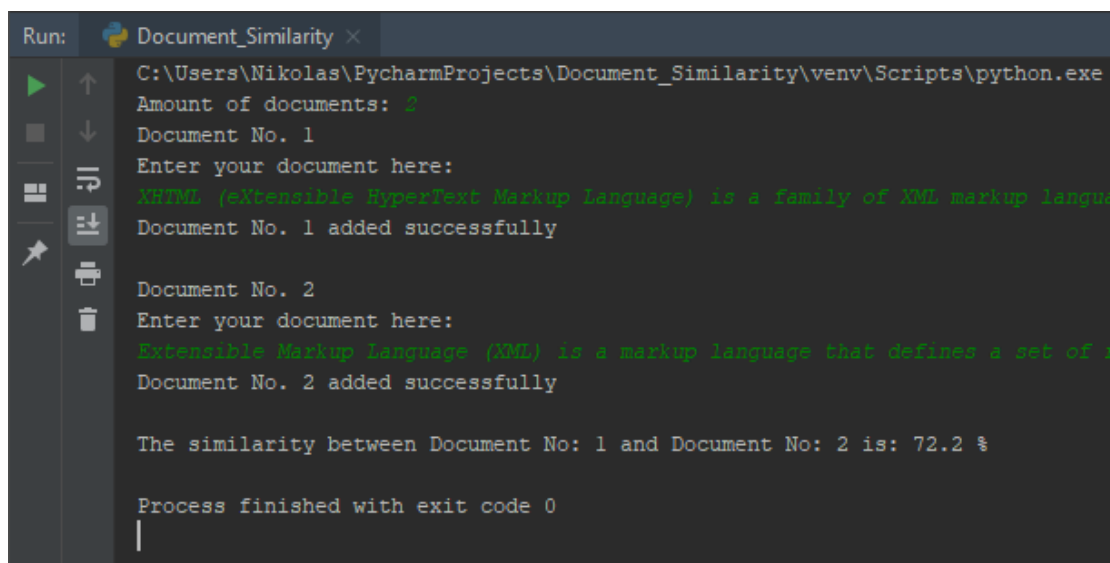
```
Run: Document_Similarity x
C:\Users\Nikolas\PycharmProjects\Document_Similarity\venv\Scripts\python.exe
Amount of documents: |
```

Εικόνα 2:

Στην περίπτωση που πατήσουμε ότι θέλουμε να εισάγουμε δύο Documents έχουμε τα εξής αποτελέσματα:

Το πρόγραμμα μας ζητάει να εισάγουμε το Document No. 1 και αφού το εισάγουμε σωστά το αποθηκεύει και μας ζητάει να εισάγουμε και το δεύτερο Document. Αφού εισάγουμε και αυτό, το πρόγραμμα κατευθείαν μας βγάζει το ποσοστό ομοιότητάς τους και τερματίζει.

Τα κείμενα που χρησιμοποιήθηκαν είναι και αυτά που δόθηκαν ως παράδειγμα στην άσκηση.



```
Run: Document_Similarity x
C:\Users\Nikolas\PycharmProjects\Document_Similarity\venv\Scripts\python.exe
Amount of documents: 2
Document No. 1
Enter your document here:
XHTML (eXtensible HyperText Markup Language) is a family of XML markup languages
Document No. 1 added successfully

Document No. 2
Enter your document here:
Extensible Markup Language (XML) is a markup language that defines a set of rules
Document No. 2 added successfully

The similarity between Document No: 1 and Document No: 2 is: 72.2 %

Process finished with exit code 0
|
```


Εικόνα 3:

Στην περίπτωση που πατήσουμε ότι θέλουμε να εισάγουμε παραπάνω από δύο Documents, έστω τέσσερα, έχουμε τα εξής αποτελέσματα:

Αφού εισάγουμε και τα τέσσερα Documents, μέσα από τους συνδυασμούς παίρνουμε τα ποσοστά ομοιότητας για όλα τα Documents. Έπειτα, το πρόγραμμα ζητάει να εισάγουμε έναν αριθμό μεταξύ του ένα και του έξι, καθώς τόσοι είναι όλοι οι πιθανοί συνδυασμοί που μπορούμε να πάρουμε. Εισάγουμε το τρία και βλέπουμε ότι όντως εμφανίζει τα ποσοστά των τριών “Top Similar Documents” καθώς και τους αριθμούς που χαρακτηρίζουν καθένα από αυτά και το πρόγραμμα τερματίζει.

Τα κείμενα που χρησιμοποιήθηκαν είναι και αυτά που δόθηκαν ως παράδειγμα στην άσκηση και δύο ακόμα από το Google Scholar με λέξη κλειδί “Software”.

```
C:\Users\Nikolas\PycharmProjects\Document_Similarity\venv\Scripts\python.exe
Amount of documents: 4
Document No. 1
Enter your document here:
XML (eXtensible HyperText Markup Language) is a family of XML markup languages
Document No. 1 added successfully

Document No. 2
Enter your document here:
Extensible Markup Language (XML) is a markup language that defines a set of rules
Document No. 2 added successfully

Document No. 3
Enter your document here:
Ideally in the deployment phase, components should be composable, and their s
Document No. 3 added successfully

Document No. 4
Enter your document here:
The most important contribution to the success or failure of a software proje
Document No. 4 added successfully

The similarity between Document No: 1 and Document No: 2 is: 72.2 %
The similarity between Document No: 1 and Document No: 3 is: 16.18 %
The similarity between Document No: 1 and Document No: 4 is: 45.74 %
The similarity between Document No: 2 and Document No: 3 is: 15.59 %
The similarity between Document No: 2 and Document No: 4 is: 48.54 %
The similarity between Document No: 3 and Document No: 4 is: 29.3 %

Enter a Number between 1 and 6
Find the top similar documents: 3

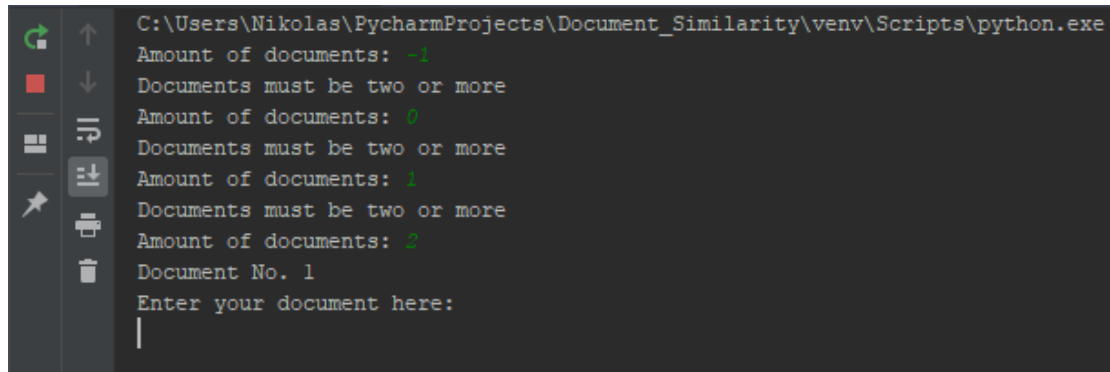
1 The 72.2 % similarity, come from document No: 1 and Document No: 2
2 The 48.54 % similarity, come from document No: 2 and Document No: 4
3 The 45.74 % similarity, come from document No: 1 and Document No: 4

Process finished with exit code 0
```

Ας εξετάσουμε τώρα τι γίνεται στις περιπτώσεις που ο χρήστης κάνει κάποιο “λάθος”.

Εικόνα 4:

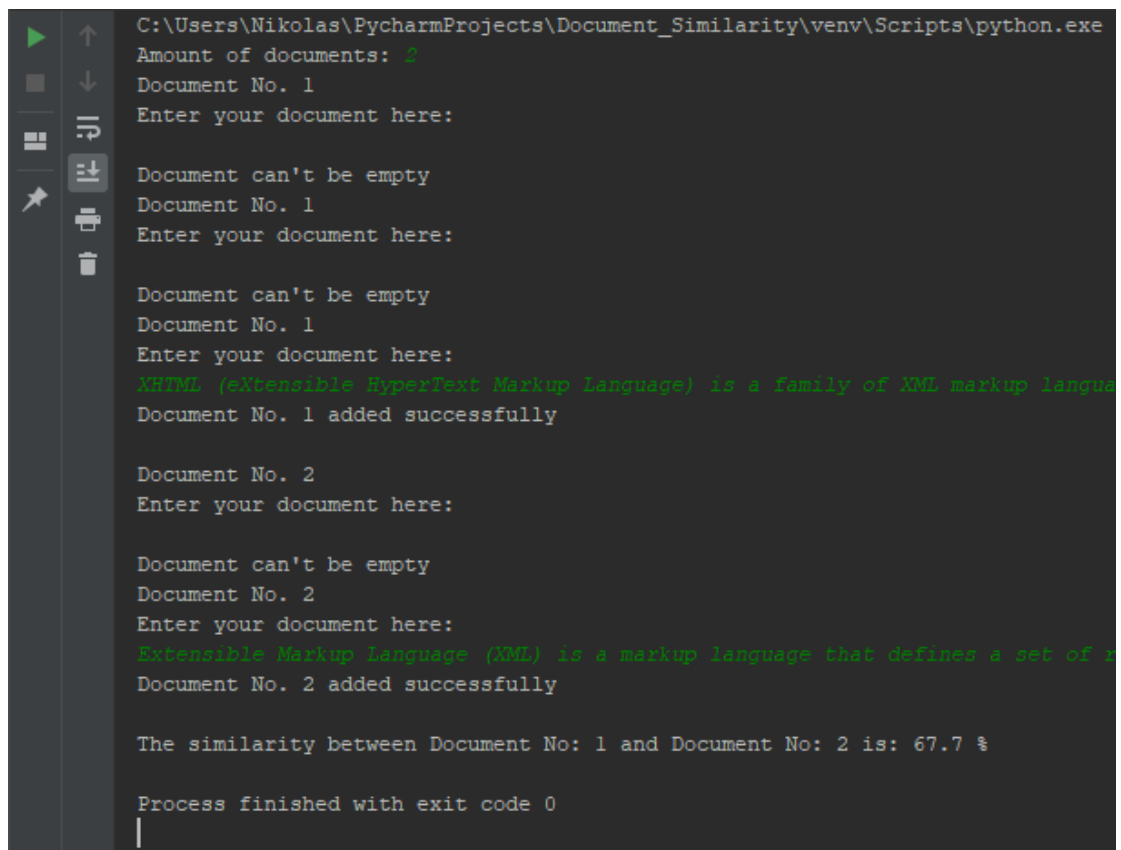
Βλέπουμε πως τις δοκιμές -1, 0, 1 ως αριθμούς Documents το πρόγραμμα δεν τις δέχεται ενώ με την 2 προχωράει κανονικά.



```
C:\Users\Nikolas\PycharmProjects\Document_Similarity\venv\Scripts\python.exe
Amount of documents: -1
Documents must be two or more
Amount of documents: 0
Documents must be two or more
Amount of documents: 1
Documents must be two or more
Amount of documents: 2
Document No. 1
Enter your document here:
|
```

Εικόνα 5:

Σε περίπτωση που ο χρήστης εισάγει κενό Document το πρόγραμμα δεν το δέχεται και ζητάει να γίνει ξανά εισαγωγή του Document. Μόνο με εισαγωγή Document το πρόγραμμα προχωράει



```
C:\Users\Nikolas\PycharmProjects\Document_Similarity\venv\Scripts\python.exe
Amount of documents: 2
Document No. 1
Enter your document here:
Document can't be empty
Document No. 1
Enter your document here:
Document can't be empty
Document No. 1
Enter your document here:
<HTML (eXtensible HyperText Markup Language) is a family of XML markup languages
Document No. 1 added successfully

Document No. 2
Enter your document here:
Document can't be empty
Document No. 2
Enter your document here:
Extensible Markup Language (XML) is a markup language that defines a set of rules
Document No. 2 added successfully

The similarity between Document No: 1 and Document No: 2 is: 67.7 %

Process finished with exit code 0
|
```

Εικόνα 6:

Σε περίπτωση λάθους στον αριθμό των συνδυασμών το πρόγραμμα ζητάει να ξανά πληκτρολογήσει ο χρήστης τον αριθμό των συνδυασμών που θέλει.

```
C:\Users\Nikolas\PycharmProjects\Document_Similarity\venv\Scripts\python.exe
Amount of documents: 4
Document No. 1
Enter your document here:
XHTML (eXtensible HyperText Markup Language) is a family of XML markup languages
Document No. 1 added successfully

Document No. 2
Enter your document here:
Extensible Markup Language (XML) is a markup language that defines a set of rules
Document No. 2 added successfully

Document No. 3
Enter your document here:
Ideally in the deployment phase, components should be composable, and their
Document No. 3 added successfully

Document No. 4
Enter your document here:
The most important contribution to the success or failure of a software project
Document No. 4 added successfully

The similarity between Document No: 1 and Document No: 2 is: 72.2 %
The similarity between Document No: 1 and Document No: 3 is: 16.18 %
The similarity between Document No: 1 and Document No: 4 is: 45.74 %
The similarity between Document No: 2 and Document No: 3 is: 15.59 %
The similarity between Document No: 2 and Document No: 4 is: 48.54 %
The similarity between Document No: 3 and Document No: 4 is: 29.3 %

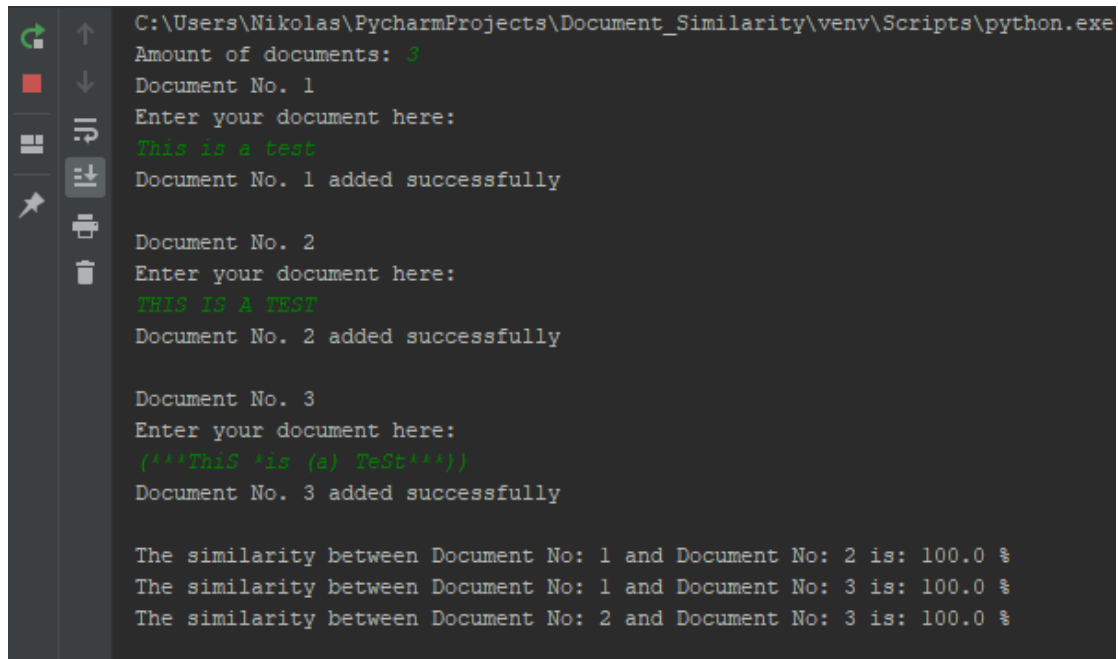
Enter a Number between 1 and 6
Find the top similar documents: 7
The number must be between 1 and 6
Find the top similar documents: 3
The number must be between 1 and 6
Find the top similar documents: 3

1 The 72.2 % similarity, come from document No: 1 and Document No: 2
2 The 48.54 % similarity, come from document No: 2 and Document No: 4
3 The 45.74 % similarity, come from document No: 1 and Document No: 4

Process finished with exit code 0
|
```

Εικόνα 7:

Τέλος βλέπουμε ότι το πρόγραμμα είναι φτιαγμένο να αγνοεί όλους τους χαρακτήρες που δεν είναι αλφαριθμητικά όπως: a-z, A-Z, 0-9 και παρατηρούμε ότι μετατρέπει όλα τα κεφαλαία γράμματα σε πεζά. Αυτά τα τρία “διαφορετικά” Documents έχουν 100% ομοιότητα μεταξύ τους.



```
C:\Users\Nikolas\PycharmProjects\Document_Similarity\venv\Scripts\python.exe
Amount of documents: 3
Document No. 1
Enter your document here:
This is a test
Document No. 1 added successfully

Document No. 2
Enter your document here:
THIS IS A TEST
Document No. 2 added successfully

Document No. 3
Enter your document here:
(*(*ThiS 'is (a) TeSt'*)*)
Document No. 3 added successfully

The similarity between Document No: 1 and Document No: 2 is: 100.0 %
The similarity between Document No: 1 and Document No: 3 is: 100.0 %
The similarity between Document No: 2 and Document No: 3 is: 100.0 %
```