

docdist1

```
1 def main():
2     if len(sys.argv) != 3:
3         print "Usage: docdist1.py filename_1 filename_2"
4     else:
5         filename_1 = sys.argv[1]
6         filename_2 = sys.argv[2]
7         sorted_word_list_1 = word_frequencies_for_file(filename_1)
8         sorted_word_list_2 = word_frequencies_for_file(filename_2)
9         distance = vector_angle(sorted_word_list_1, sorted_word_list_2)
10        print "The distance between the documents is: %0.6f (radians)" %
            distance
```

```
1 def word_frequencies_for_file(filename):
2     line_list = read_file(filename)
3     word_list = get_words_from_line_list(line_list)
4     freq_mapping = count_frequency(word_list)
5     return freq_mapping
```

```
1 def get_words_from_line_list(L):
2     word_list = []
3     for line in L:
4         words_in_line = get_words_from_string(line)
5         word_list = word_list + words_in_line
6     return word_list
7
8 def get_words_from_string(line):
9     word_list = []
10    character_list = []
11    for c in line:
12        if c.isalnum():
13            character_list.append(c)
14        elif len(character_list)>0:
15            word = "".join(character_list)
16            word = word.lower()
17            word_list.append(word)
18            character_list = []
19    if len(character_list)>0:
20        word = "".join(character_list)
21        word = word.lower()
22        word_list.append(word)
23    return word_list
```

```
1 def count_frequency(word_list):
2     L = []
3     for new_word in word_list:
4         for entry in L:
5             if new_word == entry[0]:
6                 entry[1] = entry[1] + 1
7                 break
8         else:
9             L.append([new_word,1])
10    return L
```

```
1 def vector_angle(L1,L2):
2     numerator = inner_product(L1,L2)
3     denominator = math.sqrt(inner_product(L1,L1)*inner_product(L2,L2))
4     return math.acos(numerator/denominator)
```

```
1 def inner_product(L1,L2):
2     sum = 0.0
3     for word1, count1 in L1:
4         for word2, count2 in L2:
5             if word1 == word2:
6                 sum += count1 * count2
7     return sum
```

docdist2

```
1 if __name__ == "__main__":
2     import cProfile
3     cProfile.run("main()")
```

```
1 def get_words_from_line_list(L):
2     word_list = []
3     for line in L:
4         words_in_line = get_words_from_string(line)
5         word_list.extend(words_in_line)
6     return word_list
```

docdist3

```
1 def word_frequencies_for_file(filename):
2     line_list = read_file(filename)
3     word_list = get_words_from_line_list(line_list)
4     freq_mapping = count_frequency(word_list)
5     insertion_sort(freq_mapping)
6     return freq_mapping
```

```
1 def insertion_sort(A):
2     for j in range(len(A)):
3         key = A[j]
4         i = j-1
5         while i>-1 and A[i]>key:
6             A[i+1] = A[i]
7             i = i-1
8         A[i+1] = key
9     return A
```

```
1 def inner_product(L1,L2):
2     sum = 0.0
3     i = 0
4     j = 0
5     while i<len(L1) and j<len(L2):
6         # L1[i:] and L2[j:] yet to be processed
7         if L1[i][0] == L2[j][0]:
8             # both vectors have this word
9             sum += L1[i][1] * L2[j][1]
10            i += 1
11            j += 1
12        elif L1[i][0] < L2[j][0]:
13            # word L1[i][0] is in L1 but not L2
14            i += 1
15        else:
16            # word L2[j][0] is in L2 but not L1
17            j += 1
18    return sum
```

docdist4

```
1 def count_frequency(word_list):
2     D = {}
3     for new_word in word_list:
4         if new_word in D:
5             D[new_word] = D[new_word]+1
6         else:
7             D[new_word] = 1
8     return D.items()
```

docdist5

```
1 translation_table = string.maketrans(string.punctuation+string.  
    uppercase,  
2                                     " "*len(string.punctuation)+string.lowercase)  
3  
4 def get_words_from_string(line):  
5     line = line.translate(translation_table)  
6     word_list = line.split()  
7     return word_list
```

docdist6

```
1 def word_frequencies_for_file(filename):  
2     line_list = read_file(filename)  
3     word_list = get_words_from_line_list(line_list)  
4     freq_mapping = count_frequency(word_list)  
5     freq_mapping = merge_sort(freq_mapping)  
6     return freq_mapping
```

```
1 def merge_sort(A):  
2     n = len(A)  
3     if n==1:  
4         return A  
5     mid = n//2  
6     L = merge_sort(A[:mid])  
7     R = merge_sort(A[mid:])  
8     return merge(L,R)  
9  
10 def merge(L,R):  
11     i = 0  
12     j = 0  
13     answer = []  
14     while i<len(L) and j<len(R):  
15         if L[i]<R[j]:  
16             answer.append(L[i])  
17             i += 1  
18         else:  
19             answer.append(R[j])  
20             j += 1  
21     if i<len(L):  
22         answer.extend(L[i:])  
23     if j<len(R):  
24         answer.extend(R[j:])  
25     return answer
```

docdist7

```
1 def count_frequency(word_list):
2     D = {}
3     for new_word in word_list:
4         if new_word in D:
5             D[new_word] = D[new_word]+1
6         else:
7             D[new_word] = 1
8     return D
```

```
1 def word_frequencies_for_file(filename):
2     line_list = read_file(filename)
3     word_list = get_words_from_line_list(line_list)
4     freq_mapping = count_frequency(word_list)
5     return freq_mapping
```

```
1 def inner_product(D1,D2):
2     sum = 0.0
3     for key in D1:
4         if key in D2:
5             sum += D1[key] * D2[key]
6     return sum
```

docdist8

```
1 def get_words_from_text(text):
2     text = text.translate(translation_table)
3     word_list = text.split()
4     return word_list
5
6 def word_frequencies_for_file(filename):
7     text = read_file(filename)
8     word_list = get_words_from_text(text)
9     freq_mapping = count_frequency(word_list)
10    return freq_mapping
```