faiza khurshid faiza.khurshid@uni-bonn.de (mailto:faiza.khurshid@uni-bonn.de)

# Exercise 1 (Computing Disk Space Usage, 6 Points)

a) The overall size (in bytes) of files in the specified directory, including its complete directory tree

(It must return one overall size which include all subdirectories and their subdirectories, etc.)

b) The overall size (in bytes) of those files (within the same tree) that end in the given file extension.

c) The percentage of space that is taken up by files with the given extension. Take special care to make your code robust. In particular, when the user specifies a non-existing directory or some other error occurs while traversing it, do not print the above-mentioned information. Instead, output a clear error message that indicates what went wrong. Also take care to correctly deal with empty directories. Include several calls to your function that illustrate its correct behavior in the different cases.

In [85]:
```
import os
def size(directory,file_extention):
    file_size=0
    directory_size = 0
    if not os.path.isdir(directory):
        raise Exception ("Invalid directory")
    else:
        for path, dirs, files in os.walk(directory):
            for file in files:
                joinn = os.path.join(path, file)
                directory_size += os.path.getsize(joinn)
# file overall size of files in given extention
                if file.endswith(file_extention):
                   p=os.path.join(path,file)
                   f_size=os.path.getsize(p)
                   file_size+=f_size


    if directory_size== 0:
        print('empty directory')

    if file_size==0:
        print('file not exist')


    print('the overall size of directory:',directory_size)
    print('file size in given extention:',file_size)

# find the percentage space taken up by files
    space_percentage= file_size/directory_size*100
    print('space_percentage :',str(space_percentage) + '%')

size('C:\\Users\\toqee\\music','.txt')
```

```
the overall size of directory: 10424379
file size in given extention: 14920
space_percentage : 0.1431260317760895%
```

# • Write function AtlasDict(input) that takes atlas xml file path and returns the dictionary of index to structural name. For example in ”Harvard-Oxford Cortical Structural Atlas”, the index of ”Postcentral Gyrus” is 16. (8p)

In [1]:
```
def AtlasDict(inputt):
    with open (inputt,'r') as f:
        lis=[]
        file_split=f.read().split('/')[18:]
    for i in range(len(file_split)):
        file_rep = file_split[i].replace('>','').replace('<','')
        S=file_rep.split('"')
        lis.append(S)
    lis_slic=lis[:-3]
    dictionary={j[1]:j[8] for j in lis_slic}
    return dictionary

print(AtlasDict('C:\\Users\\toqee\\Desktop\\HarvardOxford-Subcortical.xml'))
```

```
{'0': 'Left Cerebral White Matter', '1': 'Left Cerebral Cortex ', '2': 'Left La
teral Ventrical', '3': 'Left Thalamus', '4': 'Left Caudate', '5': 'Left Putame
n', '6': 'Left Pallidum', '7': 'Brain-Stem', '8': 'Left Hippocampus', '9': 'Lef
t Amygdala', '10': 'Left Accumbens', '11': 'Right Cerebral White Matter', '12':
'Right Cerebral Cortex ', '13': 'Right Lateral Ventricle', '14': 'Right Thalamu
s', '15': 'Right Caudate', '16': 'Right Putamen', '17': 'Right Pallidum', '18':
'Right Hippocampus', '19': 'Right Amygdala', '20': 'Right Accumbens'}
```

In [2]:
```
print(AtlasDict('C:\\Users\\toqee\\Desktop\\HarvardOxford-Cortical.xml'))
```

```
{'0': 'Frontal Pole', '1': 'Insular Cortex', '2': 'Superior Frontal Gyrus',
'3': 'Middle Frontal Gyrus', '4': 'Inferior Frontal Gyrus, pars triangularis',
'5': 'Inferior Frontal Gyrus, pars opercularis', '6': 'Precentral Gyrus', '7':
'Temporal Pole', '8': 'Superior Temporal Gyrus, anterior division', '9': 'Super
ior Temporal Gyrus, posterior division', '10': 'Middle Temporal Gyrus, anterior
division', '11': 'Middle Temporal Gyrus, posterior division', '12': 'Middle Tem
poral Gyrus, temporooccipital part', '13': 'Inferior Temporal Gyrus, anterior d
ivision', '14': 'Inferior Temporal Gyrus, posterior division', '15': 'Inferior
Temporal Gyrus, temporooccipital part', '16': 'Postcentral Gyrus', '17': 'Super
ior Parietal Lobule', '18': 'Supramarginal Gyrus, anterior division', '19': 'Su
pramarginal Gyrus, posterior division', '20': 'Angular Gyrus', '21': 'Lateral O
ccipital Cortex, superior division', '22': 'Lateral Occipital Cortex, inferior
division', '23': 'Intracalcarine Cortex', '24': 'Frontal Medial Cortex', '25':
'Juxtapositional Lobule Cortex (formerly Supplementary Motor Cortex)', '26': 'S
ubcallosal Cortex', '27': 'Paracingulate Gyrus', '28': 'Cingulate Gyrus, anteri
or division', '29': 'Cingulate Gyrus, posterior division', '30': 'Precuneous Co
rtex', '31': 'Cuneal Cortex', '32': 'Frontal Orbital Cortex', '33': 'Parahippoc
ampal Gyrus, anterior division', '34': 'Parahippocampal Gyrus, posterior divisi
on', '35': 'Lingual Gyrus', '36': 'Temporal Fusiform Cortex, anterior divisio
n', '37': 'Temporal Fusiform Cortex, posterior division', '38': 'Temporal Occip
ital Fusiform Cortex', '39': 'Occipital Fusiform Gyrus', '40': 'Frontal Opercul
um Cortex', '41': 'Central Opercular Cortex', '42': 'Parietal Operculum Corte
x', '43': 'Planum Polare', '44': "Heschl's Gyrus (includes H1 and H2)", '45':
'Planum Temporale', '46': 'Supracalcarine Cortex', '47': 'Occipital Pole'}
```

# • Write function summarize(input, output) that takes atlas xml file path and output path and produces a new file that contains the name of

-

# atlas, the short name, structures name and corresponding index and the total number of structural region. (5p)

In [3]:
```
def summarize(inputt,output):
    with open (inputt,'r') as f1:
        l=[]
        Slice=f1.read().split('<')[4:7:2]
        for i in range(len(Slice)):
            R= Slice[i].replace('/','')
            new_split=R.split('>')
            l.append(new_split)
            AD=AtlasDict(inputt)
        with open (output,'w') as f2:
            atlx_dic={item[0]:item[1] for item in l}
            atlx_dic.update(AD)
            atlx_dic['number of structures in atlas']=len(AD)
            x=f2.write(str(atlx_dic))

summarize('C:\\Users\\toqee\\Desktop\\HarvardOxford-Cortical.xml','C:\\Users\\toq
summarize('C:\\Users\\toqee\\Desktop\\HarvardOxford-Subcortical.xml','C:\\Users\\
```

In [4]:
```
with open('C:\\Users\\toqee\\atlas_output2.txt') as output2:
    x=output2.read()
    print(x.replace(',','\n'))
```

```
{'name': 'Harvard-Oxford Subcortical Structural Atlas'
 'shortname': 'HOSPA'
 '0': 'Left Cerebral White Matter'
 '1': 'Left Cerebral Cortex '
 '2': 'Left Lateral Ventrical'
 '3': 'Left Thalamus'
 '4': 'Left Caudate'
 '5': 'Left Putamen'
 '6': 'Left Pallidum'
 '7': 'Brain-Stem'
 '8': 'Left Hippocampus'
 '9': 'Left Amygdala'
 '10': 'Left Accumbens'
 '11': 'Right Cerebral White Matter'
 '12': 'Right Cerebral Cortex '
 '13': 'Right Lateral Ventricle'
 '14': 'Right Thalamus'
 '15': 'Right Caudate'
 '16': 'Right Putamen'
 '17': 'Right Pallidum'
 '18': 'Right Hippocampus'
 '19': 'Right Amygdala'
 '20': 'Right Accumbens'
 'number of structures in atlas': 21}
```

In [5]:
```
with open('C:\\Users\\toqee\\atlas_output1.txt') as output1:
    f=output1.read()
    print(f.replace(',','\n'))
```

```
{'name': 'Harvard-Oxford Cortical Structural Atlas'
 'shortname': 'HOCPA'
 '0': 'Frontal Pole'
 '1': 'Insular Cortex'
 '2': 'Superior Frontal Gyrus'
 '3': 'Middle Frontal Gyrus'
 '4': 'Inferior Frontal Gyrus
 pars triangularis'
 '5': 'Inferior Frontal Gyrus
 pars opercularis'
 '6': 'Precentral Gyrus'
 '7': 'Temporal Pole'
 '8': 'Superior Temporal Gyrus
 anterior division'
 '9': 'Superior Temporal Gyrus
 posterior division'
 '10': 'Middle Temporal Gyrus
 anterior division'
 '11': 'Middle Temporal Gyrus
 posterior division'
 '12': 'Middle Temporal Gyrus
 temporooccipital part'
 '13': 'Inferior Temporal Gyrus
 anterior division'
 '14': 'Inferior Temporal Gyrus
 posterior division'
 '15': 'Inferior Temporal Gyrus
 temporooccipital part'
 '16': 'Postcentral Gyrus'
 '17': 'Superior Parietal Lobule'
 '18': 'Supramarginal Gyrus
 anterior division'
 '19': 'Supramarginal Gyrus
 posterior division'
 '20': 'Angular Gyrus'
 '21': 'Lateral Occipital Cortex
 superior division'
 '22': 'Lateral Occipital Cortex
 inferior division'
 '23': 'Intracalcarine Cortex'
 '24': 'Frontal Medial Cortex'
 '25': 'Juxtapositional Lobule Cortex (formerly Supplementary Motor Cortex)'
 '26': 'Subcallosal Cortex'
 '27': 'Paracingulate Gyrus'
 '28': 'Cingulate Gyrus
 anterior division'
 '29': 'Cingulate Gyrus
 posterior division'
 '30': 'Precuneous Cortex'
 '31': 'Cuneal Cortex'
 '32': 'Frontal Orbital Cortex'
 '33': 'Parahippocampal Gyrus
```

```
 anterior division'
 '34': 'Parahippocampal Gyrus
 posterior division'
 '35': 'Lingual Gyrus'
 '36': 'Temporal Fusiform Cortex
 anterior division'
 '37': 'Temporal Fusiform Cortex
 posterior division'
 '38': 'Temporal Occipital Fusiform Cortex'
 '39': 'Occipital Fusiform Gyrus'
 '40': 'Frontal Operculum Cortex'
 '41': 'Central Opercular Cortex'
 '42': 'Parietal Operculum Cortex'
 '43': 'Planum Polare'
 '44': "Heschl's Gyrus (includes H1 and H2)"
 '45': 'Planum Temporale'
 '46': 'Supracalcarine Cortex'
 '47': 'Occipital Pole'
 'number of structures in atlas': 48}
```

# Exercise 3 (Taylor Series, 6 Points)

• Write a generator function that takes as input x and computes partial sums of the Taylor series shown above. Each new value that is generated should result from adding one more term to the previously computed sum. (4P)

• The Taylor series shown above only converges if |x| < 1. Modify the generator function so that it raises an exception when this condition is violated. When using the function, make sure that you handle this exception by displaying a meaningful error message. (2P)

In [91]:
```
def genrator(x):
    if not abs(x)<1:
         raise Exception("Enter the value of x that |x|<1 otherwise this series w
    else:
        partial_sum=1
        i=1
        while True:
            partial_sum+=x**i
            i+=1
            yield partial_sum

y=genrator(0.8)
print(next(y))
print(next(y))
print(next(y))
```

```
1.8
2.4400000000000004
2.9520000000000004
```