Itinerary Suggester

- Simple Tkinter program which accepts data such as link of the destination, desired duration of stay & budget
- 3 types of itineraries are created, for the user to pick from

#Code

```
import pandas as pd
import numpy as np
import re
import tkinter as tk
from tkinter import filedialog, Text
import os
import tkinter as tk
from tkinter import filedialog, Text
import os
import pdfkit
path_wkhtmltopdf = r'C:\Users\dayne\wkhtmltopdf\bin\wkhtmltopdf.exe'
config = pdfkit.configuration(wkhtmltopdf=path_wkhtmltopdf)
import pdfkit
from tkinter import Entry
root=tk.Tk()

def software():
    from selenium import webdriver
    from bs4 import BeautifulSoup
    import pandas as pd
    import os
    import re
    a=[]
    driver = webdriver.Chrome(executable_path=r"C:\Users\dayne\chromedriver.exe") #Set the path
to chromedriver
    headers = {'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:77.0) Gecko/20100101
Firefox/77.0'}
    l=str(entry.get())
    t=1
    driver.get(l)
    content = driver.page_source
    soup = BeautifulSoup(content)
    number = soup.findAll('h2',href=False, attrs={'class':'small product-count text-body mr-2 mb-0 d-
inline-block'})
    number=str(number)
    result=re.findall('>\d+,\d+', number)
    ##result=re.findall('>\d+',number)
    k=result[0]
    result=k.replace(">","")
    result=result.replace(",","")
```

```
result=int(result)
l=l+'/'+str(t)
b=[]
c=[]
e=[]
h=[]
for i in range(0,int(result/24)):
    t=i+1
    l=str(entry.get())
    l=l+'/'+str(t)
    driver.get(l)
    content = driver.page_source
    soup = BeautifulSoup(content)
    for parent in soup.find_all(class_="tracked-elements"):
        a_tag = parent.find("a", target="_blank")
        a_tag=str(a_tag)
        result1=re.findall('>.*<', a_tag)
        result1=result1[0]
        result1=result1.replace("<","")
        result1=result1.replace(">","")
        a.append(result1)
        b_tag=parent.find('span',class_='small text-body text-nowrap reviewLink')
        b_tag=str(b_tag)
        b_tag=re.findall('>.*<', b_tag)
        if len(b_tag)>0:
            b_tag=b_tag[0]
            b_tag=b_tag.replace("<","")
            b_tag=b_tag.replace(">","")
            b_tag=int(b_tag)
        else:
            b_tag=0
        c.append(b_tag)
        d_tag=parent.find('div',class_='h3 line-height-same mb-0 price-font text-md-right')
        d_tag=str(d_tag)
        d_tag=re.findall('>.*<', d_tag)
        if len(d_tag)>0:
            d_tag=d_tag[0]
            d_tag=d_tag.replace("<","")
            d_tag=d_tag.replace(">","")
            d_tag=d_tag.replace(",","")
            d_tag=re.findall('\d+', d_tag)
            d_tag=d_tag[0]
            d_tag=int(d_tag)
        else:
            d_tag=100000
        e.append(d_tag)
        f_tag=parent.find('span',class_='pr-1 px-1 align-middle product-card-footer-text')
        f_tag=str(f_tag)
```

```
        f_tag=re.findall('\d+', f_tag)
        if len(f_tag)==3:
            f_tag1=f_tag[2]
            int(f_tag1)
        elif len(f_tag)==4:
            if int(f_tag[3])>=24:
                f_tag1=float(f_tag[2])+float(int(f_tag[3])/60)
                f_tag1=float(f_tag1)
        else:
            f_tag1=100000
        h.append(f_tag1)



import nltk
import re
import numpy as np

stop_words = nltk.corpus.stopwords.words('english')

def normalize_document(doc):
    # lower case and remove special characters\whitespaces
    doc = re.sub(r'[^a-zA-Z0-9\s]', '', doc)
    doc = doc.lower()
    doc = doc.strip()
    # tokenize document
    tokens = nltk.word_tokenize(doc)
    # filter stopwords out of document
    filtered_tokens = [token for token in tokens if token not in stop_words]
    # re-create document from filtered tokens
    doc = ' '.join(filtered_tokens)
    return doc

normalize_corpus = np.vectorize(normalize_document)

norm_corpus = normalize_corpus(a)
len(norm_corpus)

from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(ngram_range=(1, 2), min_df=10, max_df=0.8)
cv_matrix = cv.fit_transform(norm_corpus)
cv_matrix.shape

from sklearn.cluster import KMeans

NUM_CLUSTERS = int(result/80)
```

```
    km = KMeans(n_clusters=NUM_CLUSTERS, max_iter=1000, n_init=50,
random_state=42).fit(cv_matrix)
    km
    df=pd.DataFrame({'Activity':a,'Number Of
Reviews':c,'Cost':e,'Duration':h,'Cluster':list(km.labels_)})
    df=df[df['Cost']<100000]
    df["Duration"] = pd.to_numeric(df["Duration"], downcast="float")
    df=df[df['Duration']<100000]
    df1=df.groupby('Cluster')
    df2=df1.sum('Number Of Reviews')
    df2=df2[df2['Number Of Reviews']>=1000]
    df3=list(df2.sort_values(by='Number Of Reviews',ascending=False).index)
    df4=df.sort_values(by='Number Of Reviews',ascending=False)
    df5=list(df4.index)
    f=[]
    g=[]
    q=[]
    y=[]
    z=[]
    p=0
    for r in range(0,len(df3)):
        t=0
        p=p+1
        for r1 in range(0,len(df5)):
            if t==6:
                pass
            elif df4['Cluster'][df5[r1]]==df3[r]:
                if t%2==0:
                    f.append(df4['Activity'][df5[r1]])
                    g.append(df4['Cost'][df5[r1]])
                    q.append(df4['Duration'][df5[r1]])
                    y.append(df3[r])
                    z.append(p)
                    print(p)
                t=t+1
    df6=pd.DataFrame({'Top Activities':f,'Cost':g,'Duration':q,'Cluster':y,'Cluster Rank':z})
    df6['Average Cost']=df6['Cost']/df6['Duration']
    df6.to_csv("suggested-tours.csv")
    df7=df6.sort_values(by='Cluster Rank')
    f=[]
    g=[]
    q=[]
    f1=[]
    g1=[]
    q1=[]
    f2=[]
    g2=[]
    q2=[]
```

```
y=[]
z=[]
m2=0
d=0
d1=0
d2=0
d3=[]
d4=[]
d5=[]
while(m2<len(df7)):
    s=0
    for i in range(m2,m2+2):
        a=df7['Average Cost'][i]
        if a>s:
            s=a
            largest=i
    o=s
    for i in range(m2,m2+2):
        b=df7['Average Cost'][i]
        if b<o:
            o=b
            smallest=i
    for i in range(m2,m2+3):
        if i!=smallest:
            if i!=largest:
                medium=i
    m2=m2+3
    f.append(df7['Top Activities'][largest])
    g.append(df7['Cost'][largest])
    q.append(df7['Duration'][largest])
    d=d+df7['Duration'][largest]
    d3.append(d)
    f1.append(df7['Top Activities'][medium])
    g1.append(df7['Cost'][medium])
    q1.append(df7['Duration'][medium])
    d1=d1+df7['Duration'][medium]
    d4.append(d)
    f2.append(df7['Top Activities'][smallest])
    g2.append(df7['Cost'][smallest])
    q2.append(df7['Duration'][smallest])
    d2=d1+df7['Duration'][smallest]
    d5.append(d2)

f.append('Total')
g.append(sum(g))
q.append(sum(q))
f1.append('Total')
g1.append(sum(g1))
```

```
q1.append(sum(q1))
f2.append('Total')
g2.append(sum(g2))
q2.append(sum(q2))
d3.append('NA')
d4.append('NA')
d5.append('NA')
df10=pd.DataFrame({'Top Activities':f,'Cost':g,'Duration':q,'Cummulative':d3})
df11=pd.DataFrame({'Top Activities':f1,'Cost':g1,'Duration':q1,'Cummulative':d4})
df12=pd.DataFrame({'Top Activities':f2,'Cost':g2,'Duration':q2,'Cummulative':d5})
df10.to_csv("Luxury-tours.csv")
df11.to_csv("Economy-tours.csv")
df12.to_csv("Budget-tours.csv")

f=[]
g=[]
q=[]
f1=[]
g1=[]
q1=[]
f2=[]
g2=[]
q2=[]
t=int(entry1.get())*6
t1=0
t2=0
t3=0
for i in range(0,len(df10)):
   if t1<t:
      f.append(df10['Top Activities'][i])
      g.append(df10['Cost'][i])
      q.append(df10['Duration'][i])
      t1=t1+df10['Duration'][i]
df13=pd.DataFrame({'Top Activities':f,'Cost':g,'Duration':q})

for i in range(0,len(df11)):
   if t2<t:
      f1.append(df11['Top Activities'][i])
      g1.append(df11['Cost'][i])
      q1.append(df11['Duration'][i])
      t2=t2+df11['Duration'][i]
df14=pd.DataFrame({'Top Activities':f1,'Cost':g1,'Duration':q1})

for i in range(0,len(df12)):
   if t3<t:
      f2.append(df12['Top Activities'][i])
      g2.append(df12['Cost'][i])
      q2.append(df12['Duration'][i])
```

```
        t3=t3+df10['Duration'][i]
df15=pd.DataFrame({'Top Activities':f2,'Cost':g2,'Duration':q2})


df13.to_csv("Luxury-tours(18).csv")
df14.to_csv("Economy-tours(18).csv")
df15.to_csv("Budget-tours(18).csv")



f=[]
g=[]
q=[]
f1=[]
g1=[]
q1=[]
f2=[]
g2=[]
q2=[]
t=int(entry2.get())
t1=0
t2=0
t3=0
for i in range(0,len(df10)):
    if t1<t:
        f.append(df10['Top Activities'][i])
        g.append(df10['Cost'][i])
        q.append(df10['Duration'][i])
        t1=t1+df10['Cost'][i]
df13=pd.DataFrame({'Top Activities':f,'Cost':g,'Duration':q})

for i in range(0,len(df11)):
    if t2<t:
        f1.append(df11['Top Activities'][i])
        g1.append(df11['Cost'][i])
        q1.append(df11['Duration'][i])
        t2=t2+df11['Cost'][i]
df14=pd.DataFrame({'Top Activities':f1,'Cost':g1,'Duration':q1})

for i in range(0,len(df12)):
    if t3<t:
        f2.append(df12['Top Activities'][i])
        g2.append(df12['Cost'][i])
        q2.append(df12['Duration'][i])
        t3=t3+df10['Cost'][i]
df15=pd.DataFrame({'Top Activities':f2,'Cost':g2,'Duration':q2})
```

```
    df13.to_csv("Luxury-tours(b).csv")
    df14.to_csv("Economy-tours(b).csv")
    df15.to_csv("Budget-tours(b).csv")

canvas=tk.Canvas(root,height=200,width=800,bg="#263D42")
canvas.pack()
frame=tk.Frame(root,bg="white")
frame.place(relwidth=0.8,relheight=0.8,relx=0.1,rely=0.1)
label=tk.Label(frame,text="AI Itinerary Generator",bg="gray")
label.pack()
label=tk.Label(frame,text="Please Enter The Link Of The City In The Following Format",bg="gray")
label.pack()
label=tk.Label(frame,text="Example: https://www.viator.com/en-IN/Paris/d479-ttd",bg="gray")
label.pack()
entry=Entry()
entry.config(font=('Ink Free',20))
entry.config(bg="White")
entry.pack()
label=tk.Label(frame,text="Whats The Duration Of Your Stay In Days?",bg="gray")
label.pack()
entry1=Entry()
entry1.config(font=('Ink Free',20))
entry1.config(bg="White")
entry1.pack()
label=tk.Label(frame,text="Whats Your Budget?",bg="gray")
label.pack()
entry2=Entry()
entry2.config(font=('Ink Free',20))
entry2.config(bg="White")
entry2.pack()
label=tk.Label(frame,text="Click On The Start Button To Begin",bg="gray")
label.pack()
openFile=tk.Button(root,text="Start",padx=10,pady=5,fg="white",bg="#263D42",command=softwar
e)
openFile.pack()
root.mainloop()
```