2. Vizualizace dat

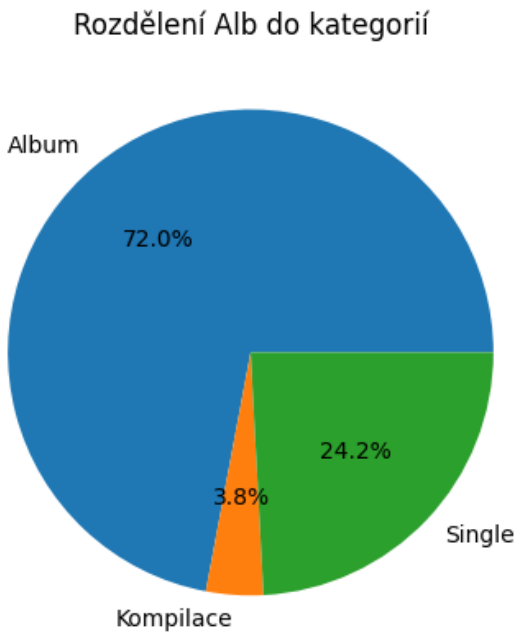
Zadání:

V jednom ze cvičení jste probírali práci s moduly pro vizualizaci dat. Mezi nejznámější moduly patří matplotlib (a jeho nadstavby jako seaborn), pillow, opencv, aj. Vyberte si nějakou zajímavou datovou sadu na webovém portále Kaggle a proveďte datovou analýzu datové sady. Využijte k tomu různé typy grafů a interpretujte je (minimálně alespoň 5 zajímavých grafů). Příklad interpretace: z datové sady pro počasí vyplynulo z liniového grafu, že v létě je vyšší rozptyl mezi minimální a maximální hodnotou teploty. Z jiného grafu vyplývá, že v létě je vyšší průměrná vlhkost vzduchu. Důvodem vyššího rozptylu může být absorpce záření vzduchem, který má v létě vyšší tepelnou kapacitu.

Řešení:

Jako zdroj dat jsem využil Kaggle a jejich obsáhlou databázi datasetů. Dataset který jsem se nakonec rozhodl využít „Spotify and Youtube“ dataset. Dataset obsahuje 20000 unikátnách songů. Sloupců je dohromady 28 s informacemi které se týkají, počtu zhlédnutí a streamů, generované sloupce vypovídající o jejich vlastnostech a dalších. Zajímalo mě hned několik témat mezi nimi bylo i využití generovaných sloupců ke kategorizaci.

Na vizualizaci dat jsem využil matplotlib a pandas. Vytvořil jsem 5 grafů (2 další jsem bral jako nedokonalé a rozhodl jsem se je vynechat). Program začíná načtením dat, přičemž následují jednotlivé funkce, které definují jednotlivé grafy. Za každou funkcí se zavolají. To mi sloužilo hlavně k rychlému debuggingu, abych pouze ukazoval grafy, na kterých zrovna pracuji.



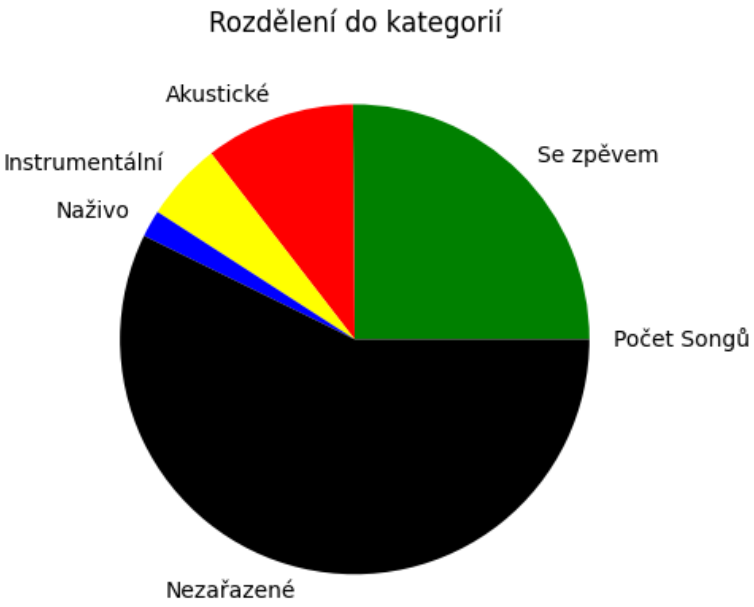
Z grafu lze vyčíst, že většina songů v datasetu je zařazená v albech poté v menším počtu následují singly a ve velicé malé míře jsou to jen kompilace.

```python
def TypySongu(DataSet):

    '''
    Rozdělení songů podle typu:
        Album,
        Single,
        Kompilace
    '''
    grouped = DataSet.groupby(["Album_type"]).size()
    grouped = grouped.rename(lambda x: x.title())
    grouped = grouped.rename(
        {
            "Compilation":"Kompilace",
            "Single": "Single"
        }
    )
    Indexes = grouped.index
    Indexes.name = ""

    fig, ax = plt.subplots()
    grouped.plot().pie(x=grouped,labels=Indexes, autopct='%1.1f%%')
    ax.set_title("Rozdělení Alb do typů")

    plt.show()
```



Z grafu lze vyčíst pár kategorií do kterých jsem songy zařadil. Kvůli nejednotnosti, jestli song je určité kategorie, jsem za pomocí této funkce zjišťoval přes prahy, jestli je daný song určitého typu, jelikož sloupce obsahují hodnoty „jistoty“ která je v rozmezí od 0-1.

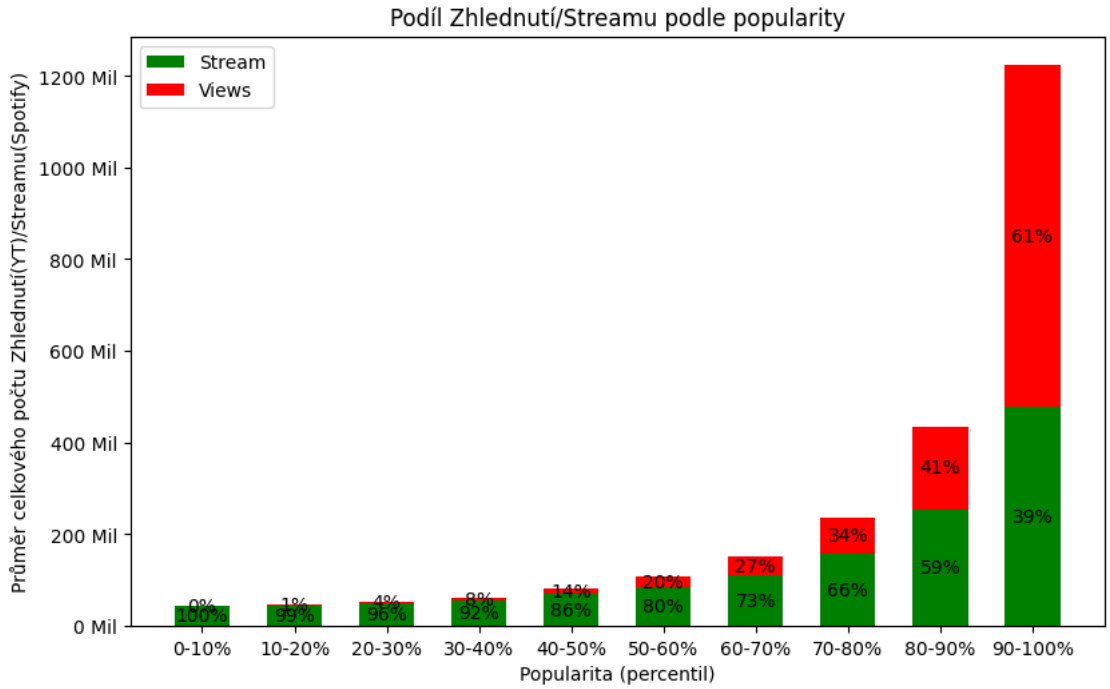
Mnoho songů je nezařazených, ale určitě pokud by se pohrálo správně s prahy, tak by jich mohlo být o dost méně a však by to bylo za cenu jistoty, že song je určitého typu.

```python
def zpracujDataSet(x):
    '''
    Zde jde o nastavení hranic jak moc chceme být jistý, že určité songy jsou určitého charakteru
    Vytáhnuto z popisu Datasetu:
        Speechiness: detects the presence of spoken words in a track.
            The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66
            describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech,
            either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
        Acousticness: a confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
        Instrumentalness: predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context.
            Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0,
            the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks,
            but confidence is higher as the value approaches 1.0.
        Liveness: detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.
            A value above 0.8 provides strong likelihood that the track is live.
    '''
    x["Speechiness"] = x["Speechiness"] > 0.1 and x["Speechiness"] < 0.66
    x["Acousticness"] = x["Acousticness"] > 0.75
    x["Instrumentalness"] = x["Instrumentalness"] > 0.5
    x["Liveness"] = x["Liveness"] > 0.8
    return x
```

```python
    TypovyDataSet = DataSet[["#","Speechiness","Acousticness","Instrumentalness","Liveness"]]
    TypovyDataSet = TypovyDataSet.apply(zpracujDataSet, axis=1)

    SongCount = TypovyDataSet["#"].count()
    TypovyDataSet=TypovyDataSet[TypovyDataSet==True].count(axis=0)
    TypovyDataSet["Nezařazené"] = SongCount-sum(TypovyDataSet)
    TypovyDataSet=TypovyDataSet.rename(
        {
            "#":"Počet Songů",
            "Speechiness": "Se zpěvem",
            "Acousticness": "Akustické",
            "Instrumentalness": "Instrumentální",
            "Liveness": "Naživo"
        }
    )

    bar_colors = ['black', 'green', 'red', 'yellow', 'blue']
    helpMe = TypovyDataSet.plot.pie(colors=bar_colors)
    helpMe.set_title("Rozdělení do kategorií")
    plt.show()
```



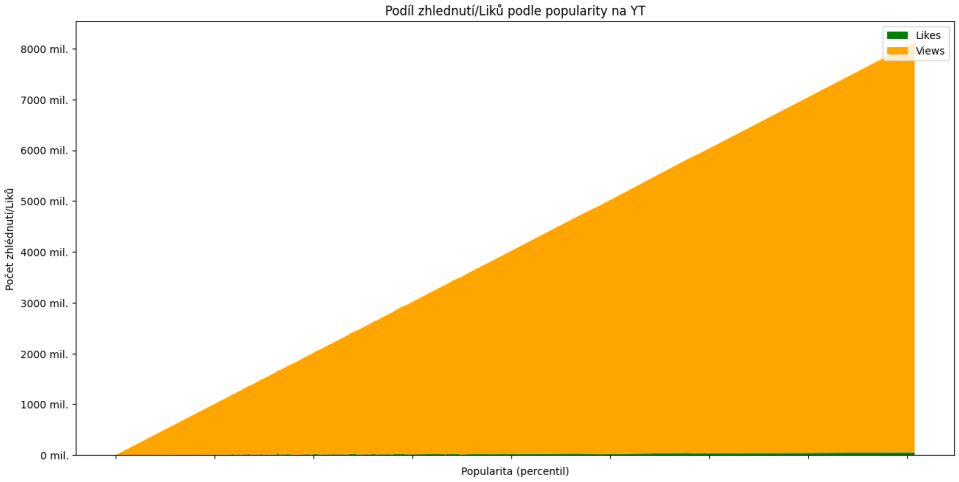
Graf, který ukazuje průměrný zhlednutí a počet streamů s viditelným podílem Spotify vs YouTube. Lze vidět, že méně známé songy se více poslouchají na Spotify mezitím co ty známější jsou více a více spouštěny na YouTube.

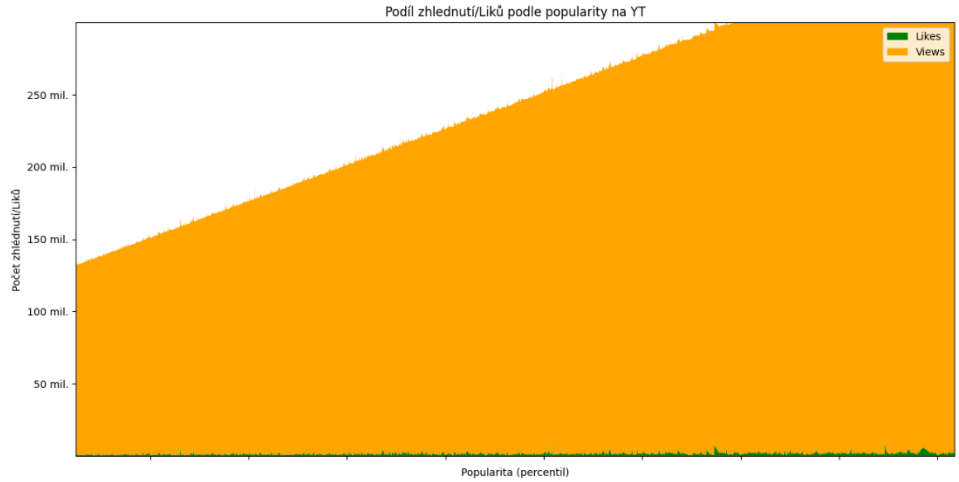
```python
def ZhlednutiAStreamy(DataSet):
    views_duration = pd.DataFrame(zip(DataSet["Stream"],DataSet["Views"]), columns=['Stream','Views'])
    views_duration["Stream"] = views_duration["Stream"].apply(lambda x : x /1000000)
    views_duration["Views"] = views_duration["Views"].apply(lambda x : x /1000000)
    views_duration = views_duration.sort_values(by="Views")
    views_duration = views_duration.reset_index(drop=True)
    views_duration = views_duration.groupby(pd.qcut(views_duration.index, 10)).mean()
    views_duration = views_duration.reset_index(drop=True)
    label1 = "Stream"
    label2 = "Views"
    fig,ax1 =  plt.subplots()


    views_duration["Stream"] = views_duration["Stream"].apply(lambda x : round(x,3))
    views_duration["Views"] = views_duration["Views"].apply(lambda x : round(x,3))
    views_duration["ViewsRatio"] = views_duration["Views"] / (views_duration["Views"] +  views_duration["Stream"]) * 100
    views_duration["StreamRatio"] = views_duration["Stream"] / (views_duration["Views"] +  views_duration["Stream"]) * 100

    ax1.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y,pos: f"{y:.0f} Mil"))
    for index, row in views_duration.iterrows():
        p = plt.bar(str((index)*10)+"-"+str((index+1)*10)+"%",row["Stream"],width=0.6,label=label1, color="green")
        ax1.bar_label(p,fmt=f"{row.StreamRatio:.0f}%", label_type='center')
        p = plt.bar(str((index)*10)+"-"+str((index+1)*10)+"%",row["Views"],width=0.6,label=label2,bottom=row["Stream"],color="red")
        ax1.bar_label(p,fmt=f"{row.ViewsRatio:.0f}%", label_type='center')

    ax1.set_xlabel("Popularita (percentil)")
    ax1.set_ylabel("Průměr celkového počtu Zhlednutí(YT)/Streamu(Spotify)")
    ax1.set_title("Podíl Zhlednutí/Streamu podle popularity")
    ax1.legend(["Stream","Views"])
    print(views_duration)
    plt.show()
```





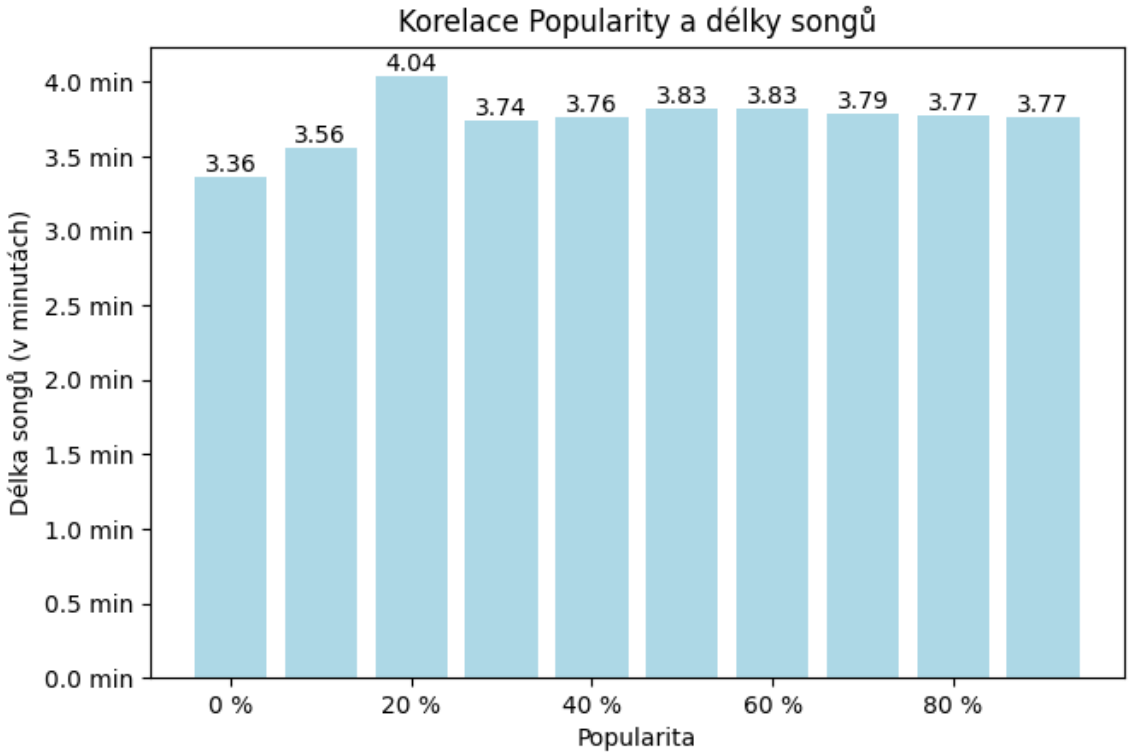
Graf, na kterém jsem chtěl ukázat podíl zhlédnutí a lajků které song na YouTube dostal. Jde vidět, jak absurdně malý podíl to je (podíl je lajků na: 0.6993400313173325 %). I přes malou čitelnost grafu jsem se ho rozhodl ponechat, aby byla vidět rozdíl mezi tím kolik lidí video zhlédlo a kolik jich jej lajknulo

```python
def SongsAndLikes(DataSet):
    '''
        Youtube songy popularita a počet liků
    '''
    views_likes = pd.DataFrame(zip(DataSet["Views"],DataSet["Likes"]), columns=['Views','Likes'])
    views_likes = views_likes.dropna()
    views_likes = views_likes.sort_values(by="Views")
    views_likes = views_likes.reset_index(drop=True)

    label1 = "Views"
    label2 = "Likes"
    fig,ax1 =  plt.subplots()
    print(f"Průměrný podíl je: {( views_likes['Likes'].sum() / (views_likes['Views'].sum() + views_likes['Likes'].sum()))*100}%")

    views_likes["LikesRatio"] = views_likes["Likes"] / views_likes["Views"] * 100
    views_likes["ViewsRatio"] = 100 -  views_likes["LikesRatio"]

    ax1.xaxis.set_major_formatter(ticker.FuncFormatter(lambda x,pos: ""))
    ax1.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y,pos: f"{y/1000000:.0f} mil."))
    ax1.stackplot(views_likes["Views"], (views_likes["Likes"],views_likes["Views"]),colors=("green","orange"))
    ax1.set_xlabel("Popularita (percentil)")
    ax1.set_ylabel("Počet zhlédnutí/Liků")
    ax1.set_title("Podíl zhlednutí/Liků podle popularity na YT")
    ax1.legend(["Likes","Views"])
    plt.show()
```



Tímto grafem jsem chtěl zjistit, jestli je nějaká korelace mezi délkou songů a popularity. Lze vidět, že na délce songu zas tolik nezáleží, ale lze vidět mírné výkyvy u méně známějších písniček

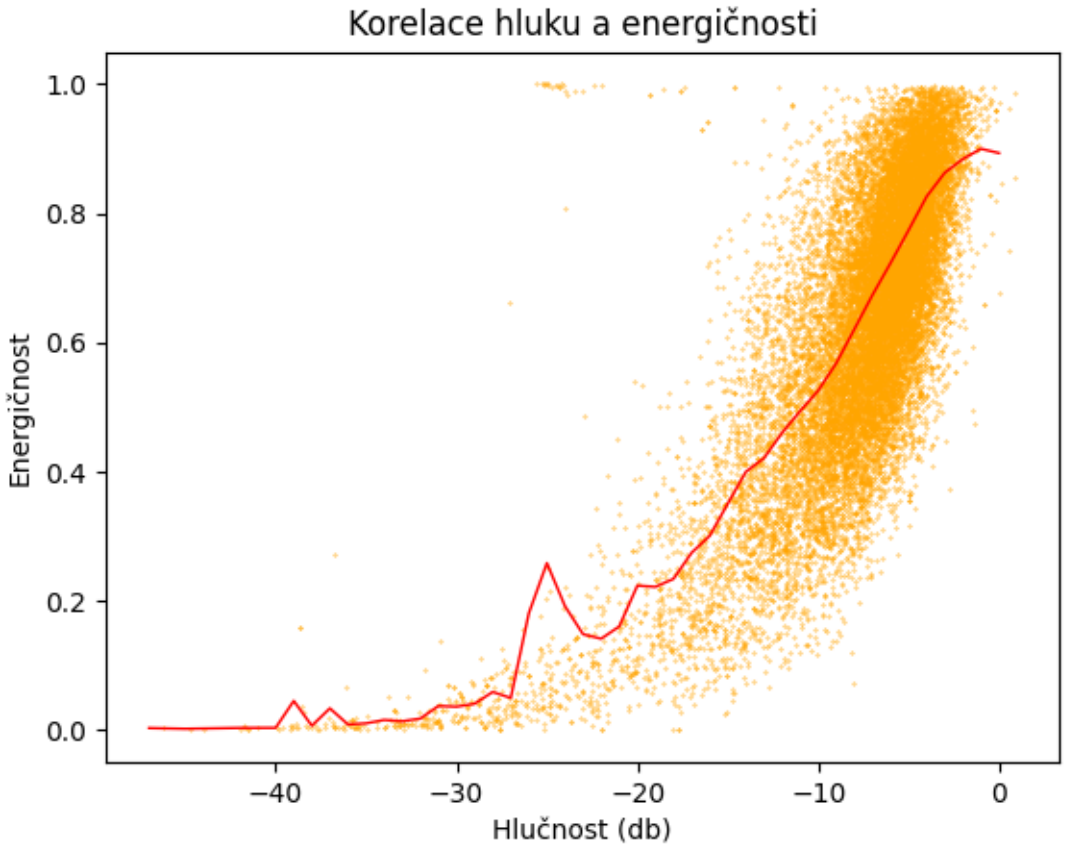
```
def PopularitaADelkaSongu(DataSet):
    duration = pd.DataFrame(zip(DataSet["Duration_ms"],DataSet["Stream"],DataSet["Views"]), columns=['Duration_ms','Stream','Views'])

    duration = duration.apply(lambda x : x /1000 / 60)
    duration["Popularity"] = duration.apply(lambda x : x["Stream"] + x["Views"] ,axis=1)
    duration = duration.sort_values(by="Popularity")
    duration = duration.reset_index(drop=True)
    duration = duration.groupby(pd.qcut(duration["Popularity"], 10)).mean()
    duration = duration.reset_index(drop=True)
    fig,ax1 =  plt.subplots()

    ax1.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y,pos: f"{y:.1f} min"))
    ax1.xaxis.set_major_formatter(ticker.FuncFormatter(lambda x,pos: f"{x*10:.0f} %"))


    ax1.set_title("Průměrná délka písniček v po 10 percentilech")

    bars = ax1.bar(duration.index,duration["Duration_ms"],color="lightblue")
    ax1.bar_label(bars,fmt="%.2f")
    ax1.set_title("Korelace Popularity a délky songů")
    ax1.set_xlabel("Popularita")
    ax1.set_ylabel("Délka songů (v minutách)")
    plt.show()
```



U tohohle grafu mě zajímalo, jestli hlučnost autoři použili ke generování měřítka energetičnosti. Lze vidět nějaká korelace mezi těmito dvěma vlastnostmi, přesto lze vidět spoustu výjimek jako například obrovský nárůst a pokles v půlce grafu

```
180  def HlukAEnergeticnost(DataSet):
181      duration = pd.DataFrame(zip(DataSet["Loudness"],DataSet["Energy"]), columns=['Loudness','Energy'])
182      duration = duration.sort_values(by="Loudness")
183      duration = duration.reset_index(drop=True)
184
185      fig,ax1 =  plt.subplots()
186      ax1.scatter(duration["Loudness"],duration["Energy"],s=0.1,color="orange")
187
188      line = duration
189      line = line.dropna()
190      line["Loudness"] = line["Loudness"].apply(lambda x : math.floor(x))
191
192      line = line.groupby("Loudness").mean()
193      ax1.set_title("Korelace hluku a energičnosti")
194      ax1.set_xlabel("Hlučnost (db)")
195      ax1.set_ylabel("Energičnost")
196
197      ax1.plot(line.index,line["Energy"], label='Data', linewidth=1, color="red")
198      plt.show()
```