Seminar 1 - Grundlagen

Hinweis: Speichern eines Notebooks immer über: STRG+S/ CMD+S oder das Disketten-Symbol. Die Option File --> Save funktioniert bei Notebooks nicht!!

1. Podcast-Sendezeit

Sie haben es geschafft und konnten sich einen der begehrten Praktikumsplätze im Team des Podcast-Newcomers "OR" ergattern. Ihre erste Aufgabe besteht darin, herauszufinden, wie viele Sendeminuten der Podcast wöchentlich hat.

1. Eine Podcastfolge dauert im Durchschnitt 70 Minuten. Dies soll aber variabel anpassbar sein. (Variable)
2. Zusätzlich werden pro Folge 3 Werbeblöcke à 3 Minuten eingespielt.
3. Wöchentlich werden 2 Podcasts veröffentlicht.
4. Die Redaktionschefin bittet Sie, das Ergebnis in Stunden gerundet auf zwei Nachkommastellen auszugeben.
5. Des Weiteren soll nicht nur einfach eine Zahl ausgegeben werden, sondern ein Satz: "Bei einer Podcastlänge von XX Minuten ergeben sich Y.YY Stunden Sendezeit pro Woche."

a.)

Schreiben Sie bitte ein Skript, welches die Berechnung übernimmt und geben Sie das Ergebnis aus.

In [ ]:
```
pf = 70
wb = 3
mal = 2

min = (pf + 3 * wb) * 2

print(f'Bei einer Podcastlänge von 70 Minuten ergeben sich {round(min/60, 2)} Stunden
```

```
Bei einer Podcastlänge von 70 Minuten ergeben sich 2.63 Stunden Sendezeit pro Woche
```

b.)

Informieren Sie sich bitte im Internet, welche weiteren Möglichkeiten es gibt, Zeichenketten für eine Ausgabe zu formatieren.

Tipp:

▶ 详情

In [ ]:
```
print('Bei einer Podcastlänge von 70 Minuten ergeben sich ' + str(round(min/60, 2)) +
```

```
Bei einer Podcastlänge von 70 Minuten ergeben sich 2.63 Stunden Sendezeit pro Woche
```

Erwarteter Output:

```
Bei einer Podcastlänge von 70 Minuten ergeben sich ...
```

2. Chaos in der Mailingliste

Die Hörer des Podcasts haben die Möglichkeit, sich für einen Newsletter anzumelden. So erfahren sie schnellstmöglich, wann es eine neue Podcast-Folge gibt. Dabei ist aufgefallen, dass manche Mailadressen den Namen erst hinter dem @-Zeichen haben.

a.)

Entwickeln Sie bitte einen Python-Ausdruck, der die Namen hinter dem @-Zeichen als Elemente einer Liste ausgibt.

In [ ]:
```
mail = "service@klara_Klarna.com"
name = mail.split('@')[1].split('.')[0].split('_')
print(name)
```

Out[ ]:
```
['klara', 'Klarna']
```

Erwarteter Output:

```
['klara', 'Klarna']
```

b.)

Nachdem Sie nun die Namen der AbonnetInnen zu den jeweiligen E-Mail-Adressen zuordnen können, stellt die Chefredakteurin Ihnen eine neue Aufgabe. Überführen Sie bitte die folgende Tabelle in ein Dictionary bestehend aus den Namen als Keys und den E-Mail-Adressen als Values:

Name	E-Mail
Klara	service@klara_Klarna.com
Sebastian	Sebastian_Duesentrieb@gmail.com
Britta	bwiebritta@yahoo.com
Klaus	Klaus@peter.de

Legen Sie dazu zunächst zwei Listen an, die Sie dann zusammenführen können.

Tipp:

▶ 详情

In [ ]:
```
names = ['Klara', 'Sebastian', 'Britta', 'Klaus']
mails = ['service@klara_Klarna.com', 'Sebastian_Duesentrieb@gmail.com',
         'bwiebritta@yahoo.com', 'Klaus@peter.de']

name_mail = dict()
for n, m in zip(names, mails):
```

```
    name_mail[n] = m
print(name_mail)
```

```
{'Klara': 'service@klara_Klarna.com', 'Sebastian': 'Sebastian_Duesentrieb@gmail.com',
'Britta': 'bwiebritta@yahoo.com', 'Klaus': 'Klaus@peter.de'}
```

3. Anpassung der Werbeeinnahmen

Die Freude im Podcast-Team ist rießig. Zuletzt kamen immer mehr interessierte ZuhörerInnen hinzu und so konnte die Zuhörerzahl in den letzten 3 Monaten um 400% gesteigert werden. Der kaufmännische Leiter des Teams will daher die Einnahmen durch die angebotenen Werbepakete "optimieren".

Im Moment existieren vier Werbepakete für den Podcast. Bisher wurden immer kundenspezifische Preise bis zu einer Höchstgrenze für jedes Paket verhandelt.

1. Paket "Einleitung", Kosten bis 350 €
2. Paket "Mitte", Kosten bis 300 €
3. Paket "Komplett", Kosten bis 600 €
4. Paket "Zwischendurch", bis 800 €

a.)

Der kaufmännische Leiter hat beschlossen, dass aufgrund der deutlich vergrößerten Reichweite folgende Preisanpassungen wohl gerechtfertigt sind: Die Pakete bis 400 € werden um 10% angehoben, die Pakete zwischen 400 und 600 € um 20% und die Pakete über 600 € um 25%.

Bitte setzen Sie die Preisanpassung in einem Pythonskript um, sodass der neue Preis in Abhängigkeit eines variablen Eingabepreises ausgegeben wird.

In [ ]:

```
input_price = 333

def preisanpassung(input):
    if input <= 400:
        input *= 1.1
    elif input <= 600:
        input *= 1.2
    else:
        input *= 1.25
    return input

print(preisanpassung(input_price))

print(preisanpassung(350))
print(preisanpassung(300))
print(preisanpassung(600))
print(preisanpassung(800))
```

```
366.3
385.00000000000006
330.0
720.0
1000.0
```

Erwarteter Output:

```
366.3
```

b.)

Durch den Erfolg des Podcasts sind nun auch die Zeiten der kundenspezifischen Preise und Verhandlungen vorbei. Jedes Paket hat ab jetzt den oben genannten Maximalpreis als Festpreis.

Bitte speichern Sie die Paketpreise in eine Liste und generieren Sie mithilfe dieser Liste eine neue Liste, in der die neuen Preis zu finden sind.

In [ ]:
```
old = [350, 300, 600, 800]

new = []

for input_price in old:
    new_price = preisanpassung(input_price)
    new.append(int(new_price))

print(new)
```

```
[385, 330, 720, 1000]
```

Erwarteter Output:

```
[385, 330, 720, 1000]
```

c.)

Nun ist Kassensturz angesagt! Ihr kaufmännischer Leiter will jetzt wissen, was die Preisanpassung in den vergangenen Wochen gebracht hat.

Dafür soll eine Funktion geschrieben werden, welcher zwei Listen (Preise und Verkaufszahlen) als Parameter übergeben werden und diese Funktion, dann den **Umsatz** zurückgibt.

Hinweis: Sie können annehmen, dass die Indizierung der beiden Listen identisch ist, d.h. der erste Eintrag in der Preisliste gehört zum ersten Eintrag in der Liste der Verkaufszahlen.

In [ ]:
```
sales = [3, 5, 7, 4]
oldPrice = [350, 300, 600, 800]
newPrice = [385, 330, 720, 1000]


def umsatz(anzahl, preis):
    umsatz = []
    for i in range(len(anzahl)):
        umsatz.append(anzahl[i] * preis[i])
    return sum(umsatz)

before = umsatz(sales, oldPrice)
after = umsatz(sales, newPrice)
print(f'Umsatz vor der Preisanpassung: {before}\nUmsatz nach der Preisanpassung: {afte
```

```
Umsatz vor der Preisanpassung: 9950
Umsatz nach der Preisanpassung: 11845
Steigerung: 1895
```

Erwarteter Output:

```
1895
```

d.)

Informieren Sie sich bitte über die Möglichkeiten der lambda-Funktion. Setzen Sie bitte anschließend die Funktion **Umsatz** als lambda-Funktion um.

In [ ]:
```
umsatz_lam = lambda anzahl, preis: sum([anzahl[i] * preis[i] for i in range(len(sale
print(umsatz_lam(sales, newPrice) - umsatz_lam(sales, oldPrice))
```

```
1895
```

Wie können lambda-Funktionen noch eingesetzt werden?

4. Expansionswahnsinn

Vom neuerlichen Erfolg des Podcasts getrieben schlägt der kaufmännische Leiter nun vor, mit einem englischsprachigen Ableger des Podcasts in die USA zu expandieren. Die Chefredakteurin ist noch skeptisch und beauftragt Sie mit der Analyse des Marktpotenzials. In der Zielgruppe des Podcasts sind laut Studien die meisten Personen zwischen 25 und 35 Jahren alt.

a.)

Finden Sie daher heraus, wie viele US-BürgerInnen derzeit in der relevanten Altergruppe sind (einschließlich der genannten Grenzen). Leider hat die Chefredakteurin nur eine Tabelle auftreiben können, die alle Geburten nach Bundesstaaten und Vornamen gliedert. Versuchen Sie die benötigten Informationen aus diesem Datensatz zu generieren(births.csv)!

In [ ]:
```
import pandas as pd
df = pd.read_csv("births.csv")
```

In [ ]:
```
print(df.head())
```

```
  state Gender  Year    Name  Count  Age
0    IN      F  1961    Lisa   1469   60
1    IN      F  1961    Mary   1151   60
2    IN      F  1961  Karen    941   60
3    IN      F  1961  Susan    941   60
4    IN      F  1961  Linda    919   60
```

In [ ]:
```
df['Age'] = 2021 - df['Year']
df_age = df[(df['Age'] >= 25) & (df['Age'] <= 35)]
print(sum(df_age.Count))
```

```
36976366
```

Erwarteter Output:

```
36976366
```

b.)

Die Chef-Redakteurin ist von Ihrem Ergebnis angetan. Um das Programm passgenau auf die meisten potenziellen ZuhörerInnen abzustimmen, bittet sie Sie um eine weitere Analyse.

Bitte finden Sie heraus, welches Alter die meisten Personen innerhalb der relevanten Altersgruppe haben. Nutzen Sie dabei ein Dictionary, welches als Schlüssel das Alter und als Wert die jeweilige Personenanzahl beinhaltet.

In [ ]:
```
print(df_age)
```

```
        state Gender  Year       Name  Count  Age
18841      IN      F  1986     Ashley   1456   35
18842      IN      F  1986    Jessica   1153   35
18843      IN      F  1986     Amanda   1107   35
18844      IN      F  1986   Jennifer    791   35
18845      IN      F  1986      Sarah    756   35
...       ...    ...   ...        ...    ...  ...
4169370    DE      M  1996    Ricardo      5   25
4169371    DE      M  1996  Shaquille      5   25
4169372    DE      M  1996     Taylor      5   25
4169373    DE      M  1996     Victor      5   25
4169374    DE      M  1996    Zackary      5   25

[750444 rows x 6 columns]
```

In [ ]:
```
age_num = df_age.groupby("Age")["Count"].sum()
age_num_dict = dict(age_num)
print(age_num_dict)
```

```
{25: 3225793, 26: 3249949, 27: 3307859, 28: 3364630, 29: 3442904, 30: 3503407, 31: 3567577, 32: 3475994, 33: 3342082, 34: 3267158, 35: 3229013}
```

In [ ]:
```
max_key = max(age_num_dict, key=age_num_dict.get)
print(max_key)
print(age_num_dict[max_key])
```

```
31
3567577
```

Erwarteter Output:

```
31
3567577
```