Andrejs Cvečkovskis
ac24005
Kurss Zinātniskā programmēšana fiziķiem

Laboratorijas darba

Pandas
Protokols

1. nodarbība.

1. uzdevums.

Uzrakstīt kodu, kas visos trīs veidos piekļūst Annas vecumam. Kāda vērtība tiek izvadīta?

Atbilde: Visos trīs gadījumos tiek izvadīta vērtība **21**.

Piezīme:

```
print(cilveki.iloc[1, 1])
print(cilveki.loc['Anna', 'vecums'])
print(cilveki.vecums[1])
```

2. uzdevums.

Izvēlēties tās novērojumu stacijas, kuru ģeogrāfiskais garums (lon) ir starp 24 un 26 un kuras atrodas augstumā zem 50 metriem. Kādas un cik loģiskās operācijas tiek izmantotas?

Atbilde: Tiek izmantotas divas salīdzinošās operācijas un viena loģiskā (AND) operācija, kas apvieno šos divus nosacījumus.

	station	lon	lat	elevation
0	Ainazi	24.3660	57.8679	6.28
2	Bauska	24.1829	56.4158	30.18
13	Riga	24.1161	56.9506	6.15
17	Skulte	24.4122	57.3006	7.66

Piezīme: `stacijas[(stacijas.lon>24)*(stacijas.lon<26)*(stacijas.elevation<50)]`

3. uzdevums.

1. Rokgrupas 'Queen' solists Fredijs Merkūrijs piedzima 1946. gada 5. septembrī un mira 1991. gada 24. novembrī. Izmantojot pandas pieejamās iespējas, aprēķini, cik dienas nodzīvoja Fredijs Merkūrijs.

   Atbilde: 16516.0

   Piezīme:

```
birth = pd.to_datetime("1946-09-05 00:00:00")
death = pd.to_datetime("1991-11-24 00:00:00")
print((death-birth)/pd.Timedelta(days=1))
```

2. Izmantojot `pandas` pieejamās iespējas noteikt nedēļas dienu, kurā piedzima Fredijs Merkūrijs. Noteikt gan konkrēto dienu, gan tās indeksu.

   Atbilde: 3; 1946-09-05 00:00:00

   Piezīme:

```
birth_date = pd.to_datetime('1946-09-05 00:00:00')
date = pd.to_datetime(birth_date)
print(birth_date.dayofweek)
print(date)
```

3. Izmantojot pandas iespējas, nosaki, kurš laika moments ir vēlāks:
   2023. gada 9. februāris 19:00 pēc Ņujorkas, ASV (America/New York) laika vai 2023. gada 10. februāra 06:00 pēc Šanhajas, Ķīna (Asia/Shanghai) laika. Pārveidot abus uz Latvijas laiku (Europe/Riga).

   Atbilde: 2023. gada 9. februāris 19:00 pēc Ņujorkas ir vēlāks.
   New York: 2023-02-10 02:00:00+02:00
   Shanghai: 2023-02-10 00:00:00+02:00
   Piezīme:

```
new_york_time = pd.to_datetime
("2023-02-09 19:00:00").tz_localize('America/New_York', ambiguous=False)
shanghai_time = pd.to_datetime
("2023-02-10 06:00:00").tz_localize('Asia/Shanghai', ambiguous=False)

print(new_york_time > shanghai_time)
print(new_york_time < shanghai_time)
print(new_york_time.tz_convert('Europe/Riga'))
print(shanghai_time.tz_convert('Europe/Riga'))
```

4. 2021 gada 2. februārī es nopirku jaunu datoru, kuram garantijas ilgums ir 1000 dienas. Izmantojot `pandas` iespējas, noteikt datumu, kad datoram beidzās garantija.
   Atbilde: 2023-10-30 00:00:00
   Piezīme:

```
buy_date = pd.to_datetime('2021-02-02 00:00:00')
dt = pd.Timedelta(days=1000)
exp_date = (buy_date+dt)
print(exp_date)
```

4. uzdevums.

Izmantojot `loc` piekļūt temperatūrai 2010. gada 15. februārī 11:00 pēc Etc/GMT-2 laika zonas.

Atbilde: T = -5.4°C

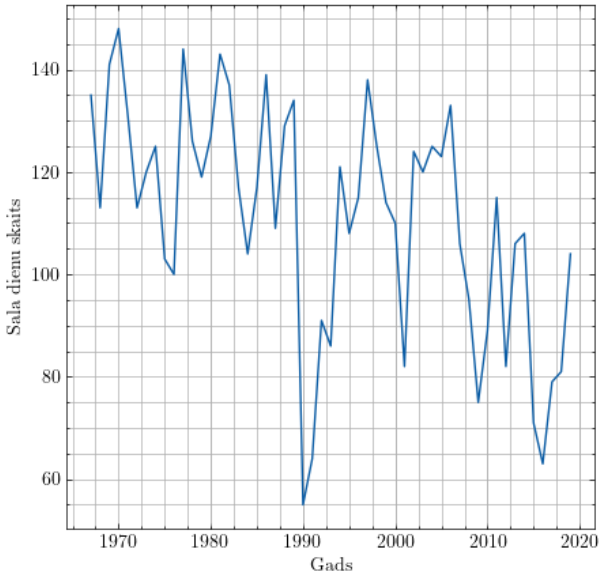
Piezīme:

```
timestamp = pd.Timestamp('2010-02-15 11:00+02:00').tz_convert(df.index.tz)
temp_2010 = df.loc[timestamp]
print(temp_2010)
```

5. uzdevums.

1. Par sala dienu sauc dienu, kad minimālā gaisa temperatūra ir zem 0C. Aprēķināt un uzzīmēt, kā sala dienu skaits ir mainījies pa gadiem.

Atbilde:

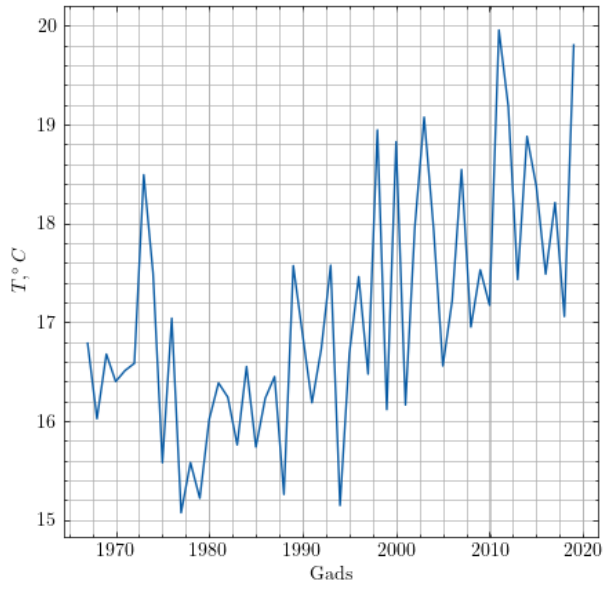


Piezīme:

```
df2 = df.resample('d').min()
df2 = df2[df2.values<0]
df2 = df2.resample('y').count()

plt.figure(figsize=(5,5))
plt.plot(df2)
plt.grid(which='both')
plt.xlabel('Gads')
plt.ylabel('Sala dienu skaits')
plt.show()
```

2. Aprēķināt un uzzīmēt, kā ir mainījusies vasaras (jūn, jūl, aug) vidējā gaisa temperatūra pa gadiem.



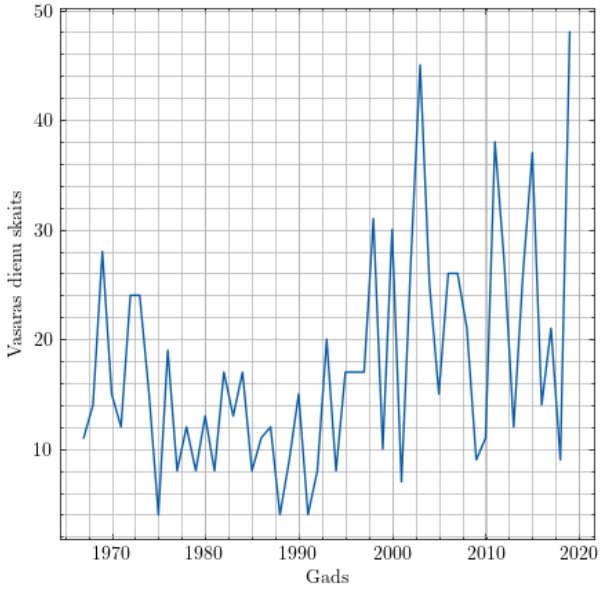
Atbilde:

Piezīme:

```
df2 = df[(df.index.month>=6)*(df.index.month<=8)]
df2 = df2.resample('y').mean()

plt.figure(figsize=(5,5))
plt.plot(df2)
plt.grid(which='both')
plt.xlabel('Gads')
plt.ylabel('$T, ^\circ C$')
plt.show()
```

3. Par vasaras dienu sauc dienu, kad maksimālā gaisa temperatūra pārsniedz 25C. Aprēķināt un uzzīmēt, kā vasaras dienu skaits ir mainījies pa gadiem.



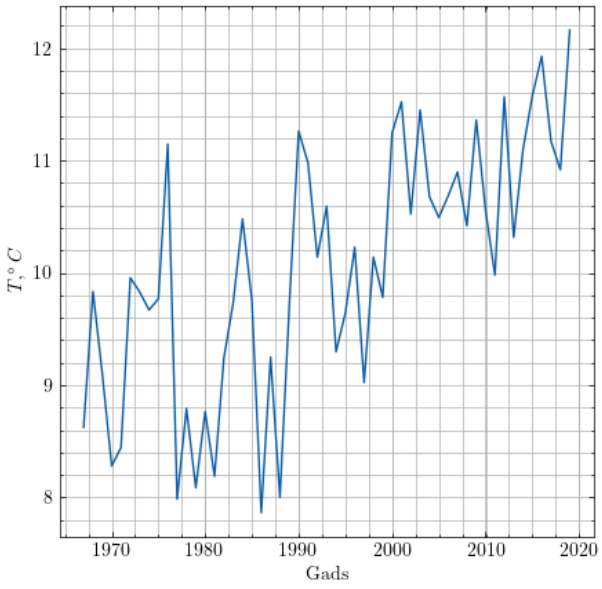
Atbilde:

Piezīme:

```
df2 = df.resample('d').max()
df2 = df2[df2.vals>25]
df2 = df2.resample('y').count()

plt.figure(figsize=(5,5))
plt.plot(df2)
plt.grid(which='both')
plt.xlabel('Gads')
plt.ylabel('Vasaras dienu skaits')
plt.show()
```

4. Aprēķināt un uzzīmēt, kā ir mainījusies diennakts maksimālās temperatūras vidējā vērtība pa gadiem.



Atbilde:

Piezīme:

```
df2 = df.resample('d').max()
df2 = df2.resample('y').mean()

plt.figure(figsize=(5,5))
plt.plot(df2)
plt.grid(which='both')
plt.xlabel('Gads')
plt.ylabel('$T, ^\circ C$')
plt.show()
```

5. Izmantojot temperatūras datus no 1970. līdz 2000. gadam katrai gada dienai (dayofyear) aprēķināt vidējo temperatūru, ko sauksim par gaisa temperatūras normu. Uzzīmēt vienā grafikā diennakts gaisa temperatūras normu un katras diennakts vidējo temperatūru 2016. gadā. Aprēķināt, cik dienās gaisa temperatūra bija virs normas un cik dienās zem normas. Ja nepieciešams, pārveidojiet pandas datu masīvus uz numpy datu masīviem. Par garajiem gadiem neuztraukties un vienkārši grupēt pēc dayofyear.

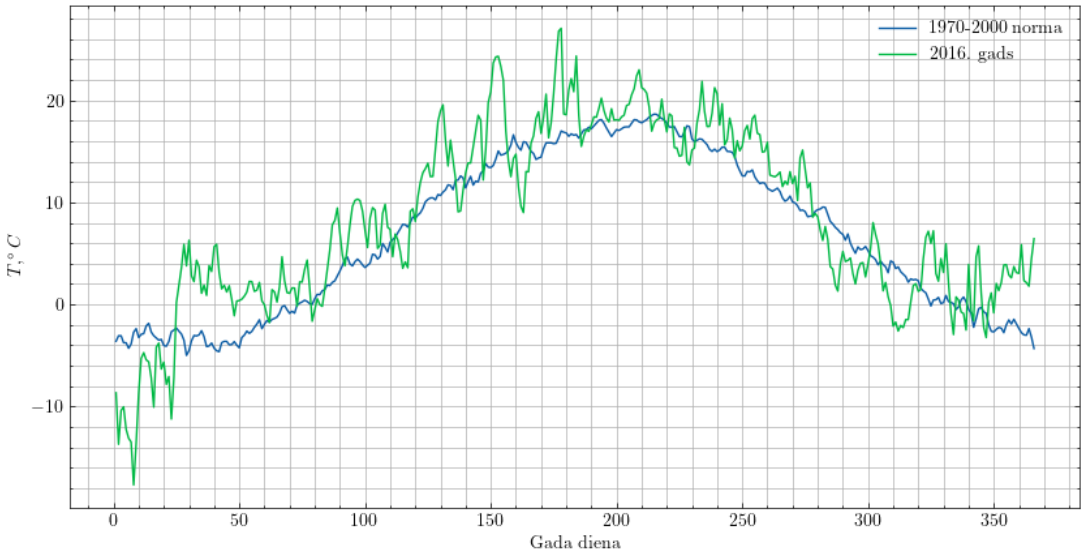
Atbilde: 249 dienās gaisa temperatūra virs normas.
117 dienās gaisa temperatūra zem normas.
Piezīme:

```
norma = df[(df.index.year>=1970)*(df.index.year<=2000)]
norma = norma.groupby(norma.index.dayofyear).mean()

gads = df[df.index.year==2016]
gads = gads.groupby(gads.index.dayofyear).mean()

print(int(np.sum(gads>norma)), 'dienās gaisa temperatūra virs normas')
print(int(np.sum(gads<norma)), 'dienās gaisa temperatūra zem normas')

plt.figure(figsize=(10, 5))
plt.plot(norma)
plt.plot(gads)
plt.grid(which='both')
plt.xlabel('Gada diena')
plt.ylabel('$T, ^\circ C$')
plt.legend(['1970-2000 norma', '2016. gads'])
plt.show()
```



6. uzdevums.

Mapē temperatura *.txt failos atrodas temperatūras novērojumu dati no meteoroloģiskajām stacijām visā Latvijā. Izmantojot staciju sarakstu no masīva stacijas, ielasīt datus no visām stacijām un apvienot tās vienā pandas masīvā. To var izdarīt, vispirms ielasot vienas stacijas datus. Tad, ejot ciklā cauri staciju sarakstam, šim masīvam var pievienot pa vienam datus no citām stacijām. Pievienot arī attēlu ar iegūto tabulu.

Atbilde:

date	Ainazi	Aluksne	Bauska	Daugavpils	Dobele	Gulbene	Jelgava	Kolka	Liepaja	Mersrags	...	Rezekne	Riga	Rujiena	Saldus	Skriveri	Skulte	Stende	Ventspils	Zilani	Zoseni
1966-01-01 00:00:00	-5.9	-7.3	1.3	1.2	1.2	-5.9	1.3	-4.1	-2.0	-3.6	...	NaN	-2.6	NaN	-2.1	0.4	-5.5	-4.2	NaN	0.3	-5.5
1966-01-01 02:00:00	NaN	NaN	NaN	1.2	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1966-01-01 03:00:00	-6.1	-7.7	1.1	0.9	-0.5	-6.6	1.1	-4.0	-3.6	-3.6	...	7.0	-3.6	-7.1	-3.1	0.3	-5.5	-4.4	-3.4	0.1	-6.6
1966-01-01 06:00:00	-6.5	-8.0	0.7	0.9	-2.2	-6.6	0.6	-4.3	-3.6	-3.6	...	-4.8	-4.4	-7.7	-4.2	-3.7	-5.4	-4.6	-3.4	-5.0	-6.7
1966-01-01 09:00:00	-7.5	-8.3	-2.4	0.5	-4.2	-7.1	-3.6	-4.3	-3.8	-4.1	...	-6.3	-4.5	-8.3	-4.7	-5.4	-5.5	-4.8	-4.4	-5.8	-6.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2018-12-31 19:00:00	0.3	-2.3	-0.4	-1.7	-0.3	-1.7	-0.3	-0.1	1.2	0.1	...	-1.8	0.1	-0.7	0.0	-0.6	0.0	-0.2	NaN	-1.3	-2.1
2018-12-31 20:00:00	0.9	-2.4	-0.4	-2.0	0.0	-2.0	-0.2	0.2	2.8	0.3	...	-2.2	-0.1	-0.8	0.1	-0.6	-0.2	0.0	NaN	-1.4	-2.1
2018-12-31 21:00:00	0.9	-2.8	-0.2	-2.3	0.5	-2.2	0.1	0.8	3.3	0.3	...	-2.3	NaN	-0.9	0.1	-0.9	-0.5	0.2	NaN	-1.2	-1.7
2018-12-31 22:00:00	1.0	-3.1	-0.1	-2.2	0.3	-2.3	0.2	1.3	3.4	0.6	...	-2.9	0.2	-1.2	0.2	-1.0	-0.4	0.6	NaN	-1.3	-1.7
2018-12-31 23:00:00	1.0	-3.1	0.1	-2.6	0.1	-2.2	0.4	1.5	3.6	1.0	...	-3.3	0.5	-1.3	0.4	-0.8	0.1	0.8	NaN	-1.2	-1.9

Piezīme:

```
df = pd.read_csv('temperatura/Ainazi.txt', sep='\t')
df.set_index('date', inplace=True)
df.index = pd.to_datetime(df.index)
df.columns = ['Ainazi']
```

```
for i in range(1, 22):
    df1 = pd.read_csv('temperatura/' + stacijas.station[i] + '.txt', sep='\t')
    df1.set_index('date', inplace=True)
    df1.index = pd.to_datetime(df1.index)
    df1.columns = [stacijas.station[i]]

    df = df.join(df1, how='outer')

df = df[(df.index.year>=1966)*(df.index.year<=2018)]

df
```

2. nodarbība.

1. uzdevums.

Ielasīt datus, kas glabājas failā taxis.csv, dati failā atdalīti ar komatiem. Piešķirt kolonnām pickup un dropoff datuma formātu. Pataisīt kolonnu pickup kā datu masīva indeksa kolonnu. Pievienot arī attēlu ar iegūto tabulu.

Atbilde:

pickup	dropoff	passengers	distance	fare	tip	tolls	total	color	payment	pickup_zone	dropoff_zone	pickup_borough	dropoff_borough
2019-03-23 20:21:09	2019-03-23 20:27:24	1	1.60	7.0	2.15	0.0	12.95	yellow	credit card	Lenox Hill West	UN/Turtle Bay South	Manhattan	Manhattan
2019-03-04 16:11:55	2019-03-04 16:19:00	1	0.79	5.0	0.00	0.0	9.30	yellow	cash	Upper West Side South	Upper West Side South	Manhattan	Manhattan
2019-03-27 17:53:01	2019-03-27 18:00:25	1	1.37	7.5	2.36	0.0	14.16	yellow	credit card	Alphabet City	West Village	Manhattan	Manhattan
2019-03-10 01:23:59	2019-03-10 01:49:51	1	7.70	27.0	6.15	0.0	36.95	yellow	credit card	Hudson Sq	Yorkville West	Manhattan	Manhattan
2019-03-30 13:27:42	2019-03-30 13:37:14	3	2.16	9.0	1.10	0.0	13.40	yellow	credit card	Midtown East	Yorkville West	Manhattan	Manhattan
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2019-03-31 09:51:53	2019-03-31 09:55:27	1	0.75	4.5	1.06	0.0	6.36	green	credit card	East Harlem North	Central Harlem North	Manhattan	Manhattan
2019-03-31 17:38:00	2019-03-31 18:34:23	1	18.74	58.0	0.00	0.0	58.80	green	credit card	Jamaica	East Concourse/Concourse Village	Queens	Bronx
2019-03-23 22:55:18	2019-03-23 23:14:25	1	4.14	16.0	0.00	0.0	17.30	green	cash	Crown Heights North	Bushwick North	Brooklyn	Brooklyn
2019-03-04 10:09:25	2019-03-04 10:14:29	1	1.12	6.0	0.00	0.0	6.80	green	credit card	East New York	East Flatbush/Remsen Village	Brooklyn	Brooklyn
2019-03-13 19:31:22	2019-03-13 19:48:02	1	3.85	15.0	3.36	0.0	20.16	green	credit card	Boerum Hill	Windsor Terrace	Brooklyn	Brooklyn

Piezīme:

```
df = pd.read_csv('taxis.csv', parse_dates=['pickup', 'dropoff'])
df.set_index('pickup', inplace=True)
df
```

2. uzdevums.

Pievienot datu masīvam kolonnas:

1. Kolonnā **time** aprēķināt brauciena ilgumu minūtēs.
   Atbilde:

   ```
   df['time'] = df.dropoff - df.index
   df['time'] = df.time.dt.total_seconds()
   df['time_minutes'] = df.time/60
   ```

2. Kolonnā **speed** aprēķināt brauciena vidējo ātrumu.
   Atbilde:

   ```
   df['mean_speed'] = (df.distance*1000)/(df.time)
   ```

3. Kolonnā **fare per kilometer** aprēķināt brauciena izmaksas (fare) uz kilometru
   Atbilde:

   ```
   df['fare_per_kilometre'] = df.total/df.distance
   ```

3. uzdevums.

Izvadīt informāciju par braucienu, kas notika 2019. gada 11. marta 20:45:26.

Atbilde:

```
dropoff            2019-03-11 20:50:27
passengers                           1
distance                          1.09
fare                               6.0
tip                               1.96
tolls                              0.0
total                            11.76
color                           yellow
payment                    credit card
pickup_zone               East Village
dropoff_zone                  Gramercy
pickup_borough               Manhattan
dropoff_borough              Manhattan
Name: 2019-03-11 20:45:26, dtype: object
```

Piezīme:

```
df.loc[pd.to_datetime('2019-03-11 20:45:26')]
```

4. uzdevums.

Cik cilvēki maksājuši ar karti, cik skaidrā?

Atbilde: 4546 cilvēki maksājuši ar karti.
1795 maksājuši skaidrā.
Piezīme:

```
payment_counts = df['payment'].value_counts()
print(payment_counts)
```

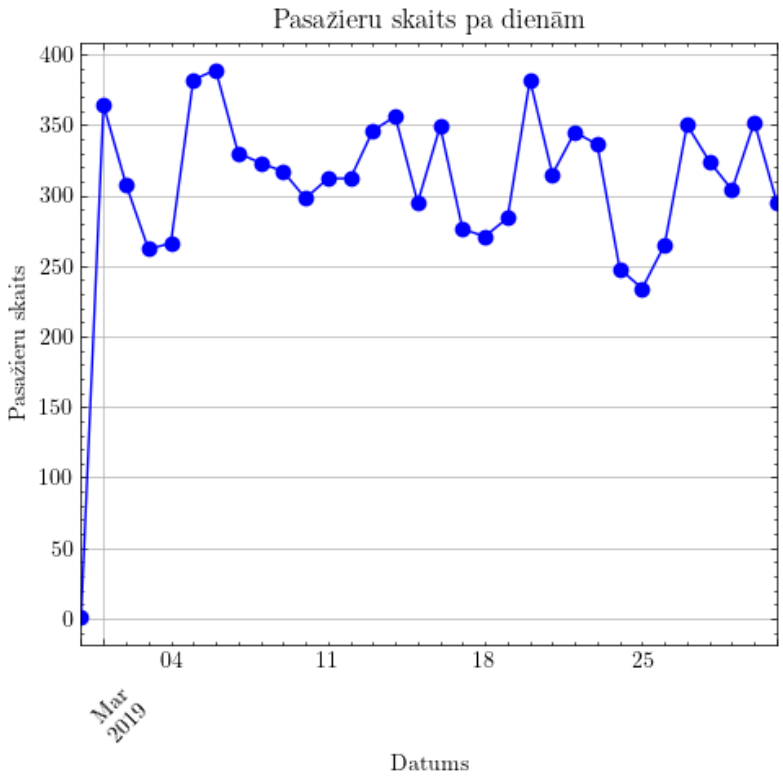
5. uzdevums.

Kuras krāsas taksometriem dod procentuāli lielāku dzeramnaudu? Cik procenti?
Atbilde: Dzelteniem taksometriem dod lielāku dzeramnaudu, 10.9%.
Piezīme:

```
plot = sns.scatterplot(x='color', y='tip', data=df)
df['tip_percentage'] = df.tip/df.total * 100
avg_tip_by_color = df.groupby('color')['tip_percentage'].mean()
print(avg_tip_by_color)
```

6. uzdevums.

Uzzīmēt grafikā pasažieru skaitu pa dienām.

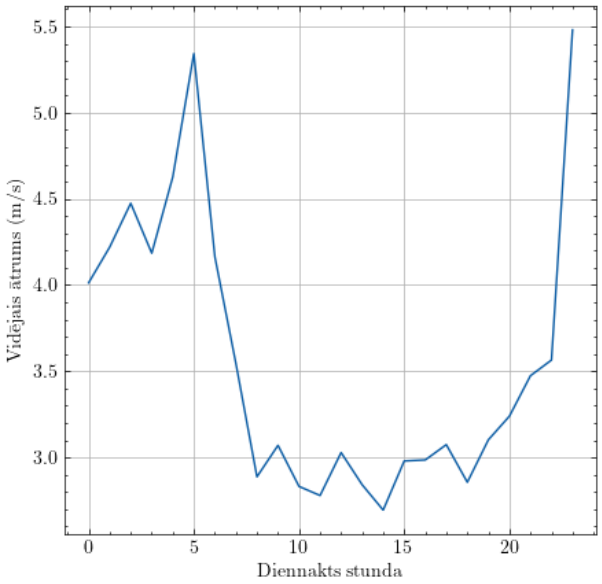


Atbilde:
Piezīme:

```
passengers_per_day = df['passengers'].resample('D').sum()
plt.figure(figsize=(5, 5))
passengers_per_day.plot(marker='o', linestyle='-', color='b')
plt.title('Pasažieru skaits pa dienām')
plt.xlabel('Datums')
plt.ylabel('Pasažieru skaits')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

7. uzdevums.

Uzzīmēt grafikā brauciena vidējo ātrumu atkarībā no diennakts stundas.

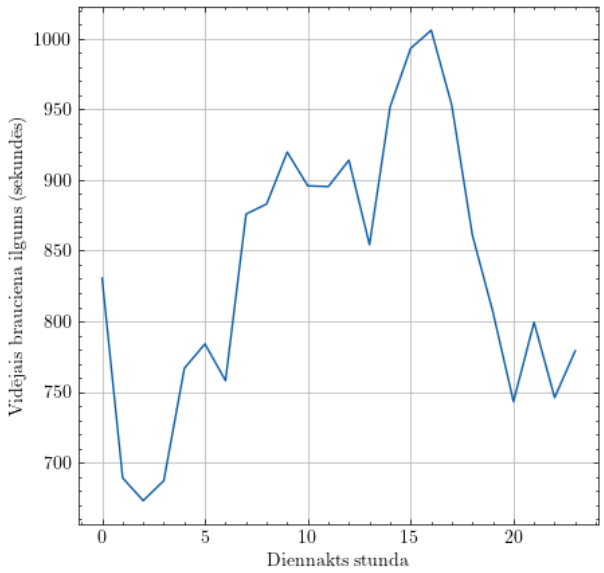


Atbilde:
Piezīme:

```
mean_speed = df['mean_speed'].groupby(df.index.hour).mean()
plt.figure(figsize=(5, 5))
mean_speed.plot()
plt.xlabel('Diennakts stunda')
plt.ylabel('Vidējais ātrums (m/s)')
plt.grid(True)
plt.show()
```

8. uzdevums.

Uzzīmēt grafikā vidējo brauciena ilgumu atkarībā no diennakts stundas.



Atbilde:
Piezīme:

```
distance_hour = df['time'].groupby(df.index.hour).mean()
plt.figure(figsize=(5, 5))
distance_hour.plot()
```

```
plt.xlabel('Diennakts stunda')
plt.ylabel('Vidējais brauciena ilgums (sekundēs)')
plt.grid(True)
plt.show()
```

9. uzdevums.

Cik braucienu ir bijis katrā maršrutā? (apskatīt braucienu skaitu atkarībā no sākuma un beigu borough)
Atbilde:

```
('Bronx', 'Bronx') 66
('Bronx', 'Brooklyn') 4
('Bronx', 'Manhattan') 25
('Bronx', 'Queens') 4
('Brooklyn', 'Bronx') 5
('Brooklyn', 'Brooklyn') 280
('Brooklyn', 'Manhattan') 67
('Brooklyn', 'Queens') 26
('Manhattan', 'Bronx') 54
('Manhattan', 'Brooklyn') 151
('Manhattan', 'Manhattan') 4857
('Manhattan', 'Queens') 162
('Manhattan', 'Staten Island') 2
('Queens', 'Bronx') 11
('Queens', 'Brooklyn') 62
('Queens', 'Manhattan') 223
('Queens', 'Queens') 342
route count = 17
num route = 6341
```

Piezīme:

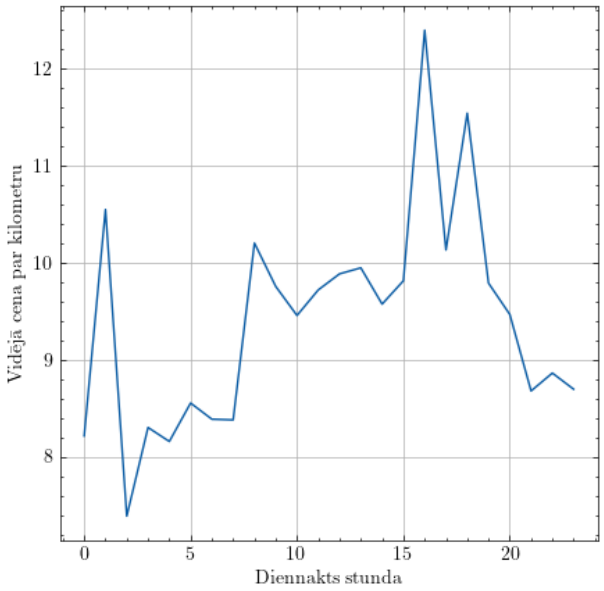
```
# all possible routes
routes = df.groupby(['pickup_borough', 'dropoff_borough']).size()
routes_count = []

for route in routes.index:
    print(route, routes[route])
    routes_count.append(routes[route])

print(np.count_nonzero(routes_count))
print(np.sum(routes_count))
```

10. uzdevums.

Atlasīt tikai tos gadījumus, kad brauciena attālums lielāks par 0 (lai izvairītos no kļūdas). Kāda ir vidējā brauciena cena par kilometru atkarībā no diennakts stundas?



Atbilde:
Piezīme:

```
df_filtered = df[df.distance > 0]
price_per_km_bigger_o = df_filtered['fare_per_kilometre'].
groupby(df_filtered.index.hour).mean()

plt.figure(figsize=(5, 5))
price_per_km_bigger_o.plot()
plt.xlabel('Diennakts stunda')
plt.ylabel('Vidējā cena par kilometru')
plt.grid(True)
plt.show()
```

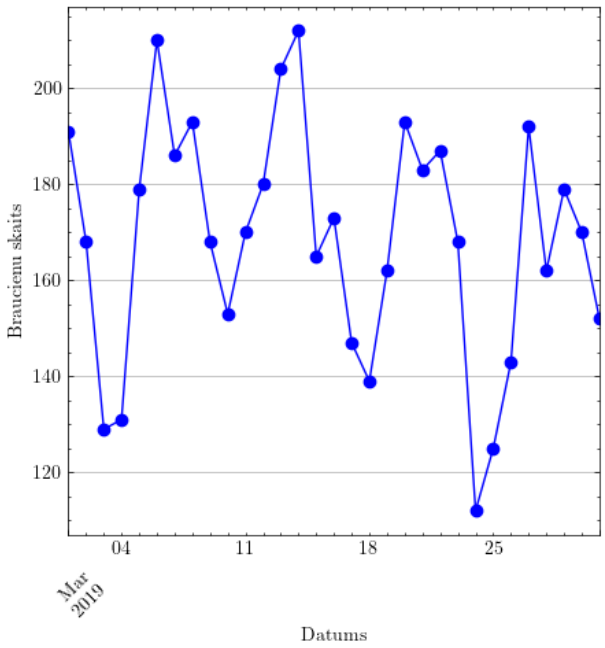
11. uzdevums.

Cik braucienu, kas garumā starp 5 km un 10 km, ir notikuši?
Atbilde: 554
Piezīme:

```
df_filtr = df[(df.distance > 5) & (df.distance < 10)]
count = df_filtr['distance'].count()
print(count)
```

12. uzdevums.

Uzzīmēt pasažieru kopējo skaitu pa dienām, kuri sāk braucienu *Manhattan*.



Atbilde:
Piezīme:

```
manhetenas_skaits = df[df.pickup_borough == 'Manhattan'].resample('D').size()
manhetenas_skaits.plot(figsize=(5, 5), marker='o', linestyle='-', color='b')
plt.ylabel('Braucienu skaits')
plt.xlabel('Datums')
plt.grid(True)
plt.xticks(rotation=45)
plt.show()
```

13. uzdevums.

Atlasīt tikai tos braucienus, kam sākuma un beigu borough sakrīt. Kāds ir vidējais brauciena ilgums katra borough robežās?
Atbilde:

	Borough	Average Trip Duration (minutes)
0	Bronx	18.28636363636364
1	Brooklyn	14.642559523809524
2	Manhattan	11.460191476219888
3	Queens	11.845808966861599

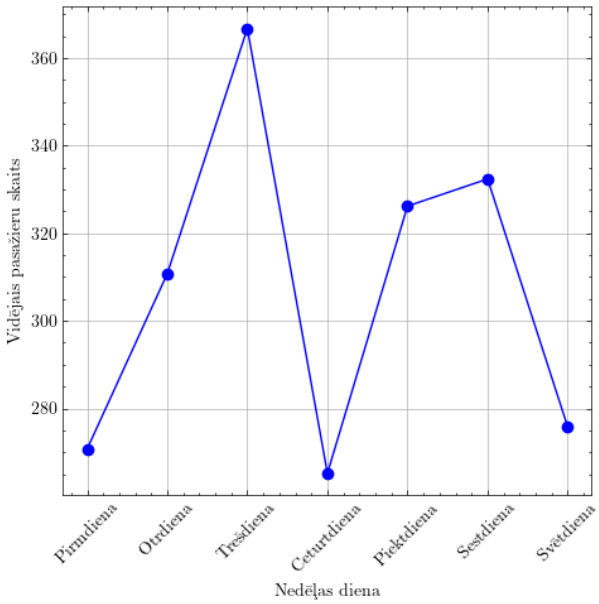
Piezīme:

```
same_borough_trips = df[df['pickup_borough'] == df['dropoff_borough']]

average_trip_duration = same_borough_trips.groupby('pickup_borough')['time_minutes'].
print(average_trip_duration)
```

14. uzdevums.

Uzzīmēt diennakts vidējo pasažieru skaitu atkarībā no nedēļas dienas.



Atbilde:

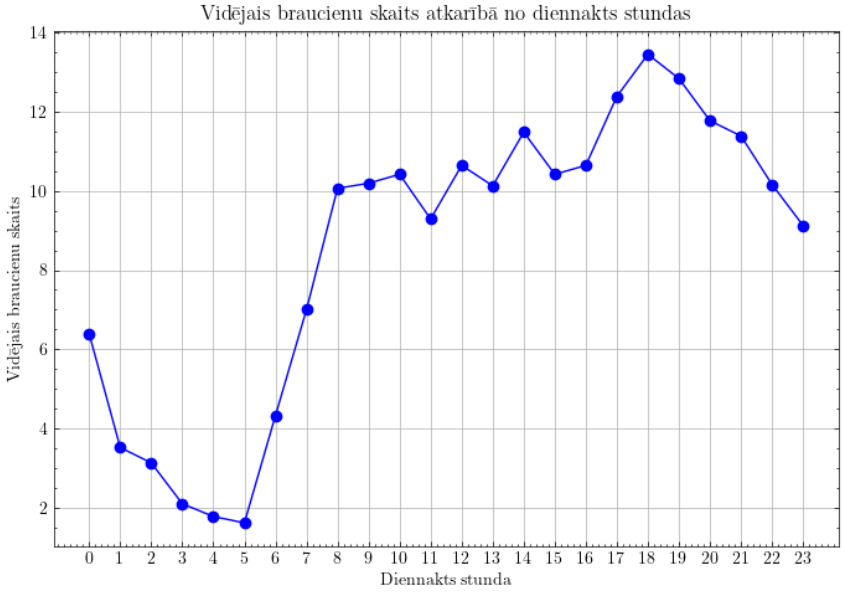
Piezīme:

```
average_passengers_by_weekday = passengers_per_day.groupby(passengers_per_day.index.
dayofweek).mean()
average_passengers_by_weekday.index = dienas

plt.figure(figsize=(5, 5))
plt.plot(average_passengers_by_weekday.index, average_passengers_by_weekday.values,
             marker='o', linestyle='-', color='b')
plt.xlabel('Nedēļas diena')
plt.ylabel('Vidējais pasažieru skaits')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

15. uzdevums.

Uzzīmēt grafikā vidējo braucienu skaitu atkarībā no diennakts stundas.



Atbilde:

Piezīme:

```
average_trips_per_hour = df.resample('H').size().groupby(df.resample('H').size().inde

plt.figure(figsize=(7, 5))
average_trips_per_hour.plot(marker='o', linestyle='-', color='b')
plt.xlabel('Diennakts stunda')
plt.ylabel('Vidējais braucienu skaits')
plt.title('Vidējais braucienu skaits atkarībā no diennakts stundas')
plt.grid(True)
plt.xticks(range(24))
plt.tight_layout()
plt.show()
```

16. uzdevums.

Atlasīt tikai tos braucienus, kam sākuma un beigu borough sakrīt. Atlasīt tikai tos gadījumus, kad brauciena attālums lielāks par 0 (lai izvairītos no kļūdas). Kāda ir vidējā brauciena cena par kilometru katrā no borough?

Atbilde:

Borough	Average Ride Price per km
Bronx	7.50
Brooklyn	7.44
Manhattan	10.71
Queens	9.22

Piezīme:

```
df_same = df[(df.pickup_borough == df.dropoff_borough)*(df.distance > 0)]
```

```
avg_fare_per_km = df_same.groupby("pickup_borough")["fare_per_kilometre"].mean()

print(avg_fare_per_km)
```