Link do ankiety:

https://docs.google.com/forms/d/e/1FAIpQLSfSvI1VL-kzSj9aQ0YsjidOl1WpG-DmVrQCqeui_crEb_gjLg/viewform

```
import pandas as pd

#create dataset
df = pd.DataFrame ({'hours': [1, 2, 4, 5, 6, 6, 7, 8, 10, 11, 11, 12, 12, 14],
'score': [64, 66, 76, 73, 81, 83, 82, 80, 88, 84, 82, 91, 93, 89]})
```

```
# view first six rows of dataset
df[0:6]
```

```
import matplotlib.pyplot as plt
plt.scatter (df.hours, df.score)
plt.title ('Hours studies vs. Exam Score')
plt.xlabel ('Hours')
plt.ylabel ('Score')
plt.show()
```

```
df.boxplot (column=['score'])
```

```
import pandas as pd
import statsmodels.api as sm

#define response variable
y = df['score']

#define explanatory variable
x = df[['hours']]

# add constant to predictior variables - dodawanie stałej do zmiennych predykcyjnych)
x = sm.add_constant (x)

# fit linear regression model
model = sm.OLS(y,x).fit()

# view model summary
print(model.summary())
```

```
# define figure size
fig = plt.figure (figsize=(12.8, 8))

#prodce residual plote
fig = sm.graphics.plot_regress_exog (model, 'hours', fig=fig)

plt.show()
```

```
# define residals
res = model.resid


# create QQ plot
fig = sm.qqplot (res, fit=True, line="45")
plt.show()
```

Zad 2.

```
import pandas as pd

# create data
df = pd.DataFrame({'hours': [1, 2, 2, 4, 2, 1, 5, 4, 2, 4, 4, 3, 6, 5, 3, 4, 6, 2, 1, 2, ],
                   'exams': [1, 3, 3, 5, 2, 2, 1, 1, 0, 3, 4, 3, 2, 4, 4, 4, 5, 1, 0, 1],
                   'score': [76, 78, 85, 88, 72, 69, 94, 94, 88, 92, 90, 75, 96, 90., 82, 85, 99, 83, 62, 76]})

#view data
df
```

```
import statsmodels.api as sm

# define response variable
y = df['score']

# define predictor variables
x = df[['hours', 'exams']]

# add constant to predictior variables
x = sm.add_constant(x)

# fit linear regression model
model = sm.OLS(y, x).fit()

# view model summary
print(model.summary())
```

Zad 3

```
import numpy as np
import pandas as pd

# create dataset
df = pd.DataFrame({'rating': [90,85, 82, 88, 94, 90, 76, 75, 87, 86],
                   'points': [25, 20, 14, 16, 27, 20, 12, 15, 14, 19],
                   'assists': [5, 7, 7, 8, 5, 7, 6, 9, 9, 5],
                   'rebounds': [11, 8, 10, 6, 6, 9, 6, 10, 10, 7]})

# view dataset
df
```

```
from statsmodels.formula.api import ols

# fit mltiple linear regression model
model = ols('rating ~ points + assists + rebounds', data=df).fit()

#view model summary
print(model.summary())
```

```
from statsmodels.stats.stattools import durbin_watson

#perform Drbin-Watson test
durbin_watson(model.resid)
```

```
2.3920546872335327
```

Przykład

```
import numpy as np
import pandas as pd

#create dataset
df = pd.DataFrame({'rating': [90, 85, 82, 88, 94, 90, 76, 75, 87, 86],
                   'points': [25, 20, 14, 16, 27, 20, 12, 15, 14, 19],
                   'assists': [5, 7, 7, 8, 5, 7, 6, 9, 9, 5],
                   'rebounds': [11, 8, 10, 6, 6, 9, 6, 10, 10, 7]})

# view dataset
df
```

```
import statsmodels.formula.api as smf

# fit regression model
fit =smf.ols('rating ~points+assists+rebounds', data=df).fit()

# view model summary
print(fit.summary())
```

```
from statsmodels.compat import lzip
import statsmodels.stats.api as sms

# perform Bresch-Pagan test
names = ['Lagrange multiplier statistic', 'p-value', 'f-value', 'f p-value']
test = sms.het_breuschpagan(fit.resid, fit.model.exog)

lzip(names, test)
```

17.11.2025

Zadanie regresją kwadratową

```
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

# define variables
hours = [6, 9, 12, 12, 15, 21, 24, 24, 27, 30, 36, 39, 45, 48, 57, 60]
happ = [12, 18, 30, 42, 48, 78, 90, 96, 96, 90, 84, 78, 66, 54, 36, 24]
```

```
# create scatterplot
plt.scatter(hours, happ)
plt.xlabel("Hours")
plt.ylabel("Happiness")
plt.show()
```

```
import numpy as np

#polynomial fit with degree = 2
model = np.poly1d(np.polyfit(hours, happ, 2))

#add fitted polynomial line to scatterplot
polyline = np.linspace(1, 60, 50)
plt.scatter(hours, happ)
plt.plot(polyline, model(polyline))
plt.show()
```

```
#define function to calculate r-squared
def polyfit(x, y, degree):
    results = {}
    coeffs = np.polyfit(x, y, degree)
    p = np.poly1d(coeffs)
    #calculate r-squared
    yhat = p(x)
    ybar = np.sum(y)/len(y)
    ssreg = np.sum((yhat-ybar)**2)
    sstot = np.sum((y - ybar)**2)
    results['r_squared'] = ssreg / sstot
    return results
#find r-squared of polynomial model with degree = 3
polyfit(hours, happ, 2)
```

Zad

```
import matplotlib.pyplot as plt

# Dane
x = [2, 3, 4, 5, 6, 7, 7, 8, 9, 11, 12]
y = [18, 16, 15, 17, 20, 23, 25, 28, 31, 30, 29]

# Tworzenie wykresu punktowego
plt.scatter(x, y)
plt.xlabel("X values")
plt.ylabel("Y values")
plt.title("Scatter Plot Example")
plt.show()
```

```
import numpy as np

#polynomial fit with degree = 3
model = np.poly1d(np.polyfit(x, y, 3))

#add fitted polynomial line to scatterplot
polyline = np.linspace(1, 12, 50)
plt.scatter(x, y)
plt.plot(polyline, model(polyline))
plt.show()
```

```python
import numpy as np

def polyfit_r2(x, y, degree):
    """
    Dopasowuje wielomian do danych i zwraca R-kwadrat.
    """
    # Dopasowanie wielomianu
    coeffs = np.polyfit(x, y, degree)
    p = np.poly1d(coeffs)

    # Przewidywane wartości
    yhat = p(x)
    ybar = np.mean(y)

    # Obliczenie R²
    ssreg = np.sum((yhat - ybar) ** 2)
    sstot = np.sum((y - ybar) ** 2)
    r_squared = ssreg / sstot

    return {"coefficients": coeffs, "r_squared": r_squared}

# Przykład użycia
x = np.array([1, 2, 3, 4, 5])
y = np.array([2.2, 2.8, 3.6, 4.5, 5.1])

result = polyfit_r2(x, y, 3)
print("Współczynniki:", result["coefficients"])
print("R²:", result["r_squared"])
```

01.12.2025

```python
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

# Generowanie danych
np.random.seed(42)
x = np.random.rand(50)
y = np.random.rand(50)
z = np.random.rand(50)

# Tworzenie wykresu
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111, projection='3d')

# Wykres punktowy 3D
ax.scatter(x, y, z, c='red', marker='o')

# Etykiety osi
ax.set_xlabel('Oś X')
ax.set_ylabel('Oś Y')
ax.set_zlabel('Oś Z')
ax.set_title('Wykres punktowy 3D')

plt.show()
```

Zadanie 2

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

np.random.seed(42)

n_points = 200
x = np.random.normal(loc=0.0, scale=1.0, size=n_points)
y = np.random.normal(loc=0.0, scale=1.0, size=n_points)
z = np.random.normal(loc=0.0, scale=1.0, size=n_points)

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

sc = ax.scatter(x, y, z, c=z, cmap='viridis', s=30, depthshade=True)

cbar = fig.colorbar(sc, ax=ax, pad=0.1)
cbar.set_label('Wartość współrzędnej z', rotation=270, labelpad=15)

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Wykres 3D – 200 punktów, kolor według z')

ax.view_init(elev=20, azim=30)

plt.tight_layout()
plt.show()
```

Zad 3

```python
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
import numpy as np

# Ustawienie figury (dwukrotnie szersza niż wysoka)
fig = plt.figure(figsize=plt.figaspect(0.5))

#=================
# Wykres 1: Surface
#=================
ax1 = fig.add_subplot(1, 2, 1, projection='3d')

# Tworzenie danych
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

# Tworzenie powierzchni
surf = ax1.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.coolwarm,
                        linewidth=0, antialiased=False)

ax1.set_title("Wykres 3D: Powierzchnia")
ax1.set_xlabel("X")
ax1.set_ylabel("Y")
ax1.set_zlabel("Z")
ax1.set_zlim(-1.01, 1.01)

# Pasek kolorów
fig.colorbar(surf, ax=ax1, shrink=0.5, aspect=10)

#==================
# Wykres 2: Wireframe
#==================
ax2 = fig.add_subplot(1, 2, 2, projection='3d')

# Te same dane
Z2 = np.cos(R)  # Inna funkcja Z, by zobaczyć różnicę

# Tworzenie siatki (wireframe)
ax2.plot_wireframe(X, Y, Z2, rstride=2, cstride=2)

ax2.set_title("Wykres 3D: Siatka")
ax2.set_xlabel("X")
ax2.set_ylabel("Y")
ax2.set_zlabel("Z")

# Pokazanie wykresów
plt.tight_layout()
plt.show()
```

Zad3

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

# Tworzymy dane wejściowe
X = np.linspace(-3, 3, 100)
Y = np.linspace(-3, 3, 100)
X, Y = np.meshgrid(X, Y)

# Nowa funkcja - dzwon Gaussa z zakłóceniem
Z = np.exp(-X**2 - Y**2) * np.cos(3*X) * np.sin(3*Y)

# Tworzenie figury i osi 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Rysujemy WYPEŁNIONY wykres konturowy
cset = ax.contourf(X, Y, Z, levels=50, cmap=cm.viridis)

# Etykiety osi
ax.set_title("Wypełniony wykres konturowy 3D")
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

# Pasek kolorów
fig.colorbar(cset, ax=ax, shrink=0.5, aspect=5)

plt.tight_layout()
plt.show()
```

07.12.2025

Zad

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

# Tworzymy dane wejściowe
X = np.linspace(-3, 3, 100)
Y = np.linspace(-3, 3, 100)
X, Y = np.meshgrid(X, Y)

# Nowa funkcja - dzwon Gaussa z zakłóceniem
Z = np.exp(-X**2 - Y**2) * np.cos(3*X) * np.sin(3*Y)

# Tworzenie figury i osi 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Rysujemy WYPEłNIONY wykres konturowy
cset = ax.contourf(X, Y, Z, levels=50, cmap=cm.viridis)

# Etykiety osi
ax.set_title("Wypełniony wykres konturowy 3D")
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

# Pasek kolorów
fig.colorbar(cset, ax=ax, shrink=0.5, aspect=5)

plt.tight_layout()
plt.show()
```

Zad

```python
# polygon_plot.py
# Przykład: wielokąty w przestrzeni 3D tworzone z obszaru "pod wykresem"
# Wymagane: numpy, matplotlib
# Uruchom: python polygon_plot.py

from matplotlib.collections import PolyCollection
import matplotlib.pyplot as plt
import numpy as np
import os

# Ustawienie ziarna generatora losowego – pozwala powtarzalnie otrzymywać te same "losowe" wyniki.
np.random.seed(19680801)

def polygon_under_graph(xlist, ylist):
    """
    Zwraca listę wierzchołków (x,y) tworzących wielokąt, który wypełnia obszar
    pod wykresem y = f(x) aż do osi y=0.
    Zakłada, że xlist jest posortowane rosnąco.
    """
    # zaczynamy od punktu (pierwsze x, 0), potem wszystkie (x,y) i na końcu (ostatnie x, 0)
    return [(xlist[0], 0.0), *zip(xlist, ylist), (xlist[-1], 0.0)]

def main():
    # Tworzymy figurę (fig) i oś 3D (ax)
    fig = plt.figure(figsize=(9, 6))
    ax = fig.add_subplot(111, projection='3d')

    # Wspólny wektor x użyty dla wszystkich "przekrojów" wielokątów
    xs = np.linspace(0.0, 10.0, 26)  # 26 punktów równomiernie od 0 do 10

    # Utworzymy kilka wielokątów umieszczonych na różnych płaszczyznach y = 0,1,2,3
    zs = range(4)  # generator: 0,1,2,3
    verts = []     # lista list punktów (każdy element to wierzchołki jednego wielokąta)

    # Dla każdego z płaszczyzn z = 0..3 tworzymy "wykres" - losowe y dla tych xs
    for i in zs:
        ys = np.random.rand(len(xs))  # losowe wartości w [0,1)
        verts.append(polygon_under_graph(xs, ys))
```

```
    # Tworzymy PolyCollection z listy wielokątów. alpha=0.6 -> częściowa przezroczystość.
    # (Można przekazać facecolors jeśli chcemy sterować kolorami.)
    poly = PolyCollection(verts, alpha=0.6)

    # Dodajemy kolekcję do osi 3D; zs określa pozycję każdego wielokąta w kierunku 'y'
    ax.add_collection3d(poly, zs=zs, zdir='y')

    # Etykiety i zakresy osi
    ax.set_xlabel('X')
    ax.set_ylabel('Y (index slice)')
    ax.set_zlabel('Z (height)')
    ax.set_xlim(0, 10)
    ax.set_ylim(-1, 4)  # od -1 do 4, żeby było trochę miejsca wokół płaszczyzn 0..3
    ax.set_zlim(0, 1)

    # Zapewniamy ładne rozmieszczenie elementów i zapisujemy plik
    out_path = os.path.join(os.getcwd(), "polygon_plot.png")
    plt.tight_layout()
    plt.savefig(out_path, dpi=150)
    print("Wykres zapisany jako:", out_path)

    # Wyświetlamy okno z wykresem (w środowisku lokalnym otworzy okno z interaktywnym wykresem)
    plt.show()

f __name__ == "__main__":
    main()
```

Zad

```python
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # rejestruje projekcję 3D

# Tworzenie figury
fig = plt.figure(figsize=(9, 6))

# Tworzenie osi 3D
ax = fig.add_subplot(111, projection='3d')

# --- DEMO 1: Użycie różnych wartości zdir ---
zdirs = (None, 'x', 'y', 'z', (1, 1, 0), (1, 1, 1))
xs = (1, 4, 4, 9, 4, 1)
ys = (2, 5, 8, 10, 1, 2)
zs = (10, 3, 8, 9, 1, 8)

for zdir, x, y, z in zip(zdirs, xs, ys, zs):
    label = f"({x}, {y}, {z}), zdir={zdir}"
    ax.text(x, y, z, label, zdir=zdir)

# --- DEMO 2: Kolor tekstu ---
ax.text(9, 0, 0, "red", color='red')

# --- DEMO 3: Tekst 2D (stała pozycja na ekranie) ---
ax.text2D(0.05, 0.95, "2D Text", transform=ax.transAxes)

# Ograniczenia i etykiety osi
ax.set_xlim(0, 10)
ax.set_ylim(0, 10)
ax.set_zlim(0, 10)
ax.set_xlabel("X axis")
ax.set_ylabel("Y axis")
ax.set_zlabel("Z axis")

plt.tight_layout()
plt.show()
```

```python
# DEMO 1 z kolorami
colors = ["red", "green", "blue", "purple", "brown", "orange"]

for zdir, x, y, z, c in zip(zdirs, xs, ys, zs, colors):
    label = f"({x}, {y}, {z}), zdir={zdir}"
    ax.text(x, y, z, label, zdir=zdir, color=c)

# DEMO 2 kolor czerwony
ax.text(9, 0, 0, "red", color='red')

# DEMO 3 tekst 2D z kolorem
ax.text2D(0.05, 0.95, "2D Text", transform=ax.transAxes, color='darkred')
```