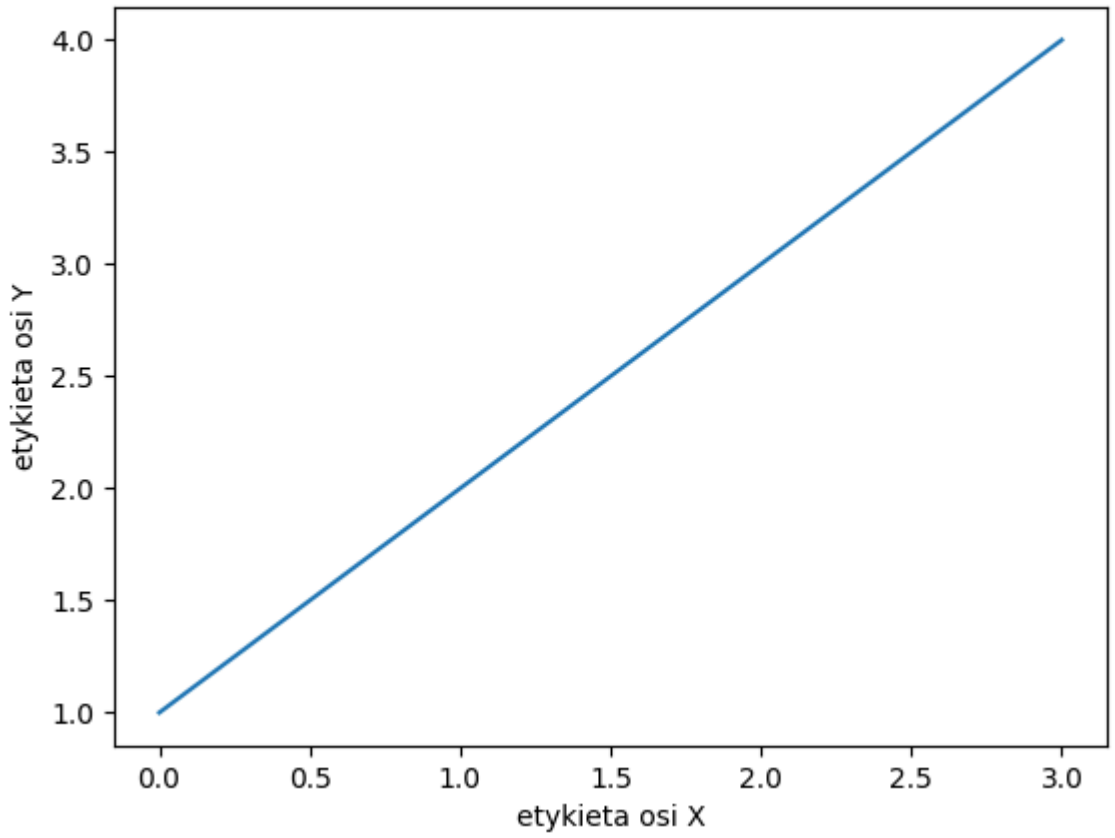
Język skryptowy lab6

Wprowadzenie do pyplot`a

`matplotlib.pyplot` jest zbiorem funkcji, które sprawiają, że `matplotlib` działa jak MATLAB. Każda funkcja `pyplot` dokonuje jakiejś zmiany w figurze: np. tworzy figurę, tworzy obszar wykreślania w figurze, wykreśla jakieś linie w obszarze wykreślania, dekoruje wykres etykietami, itp. W `matplotlib.pyplot` różne stany są zachowywane przez wywołania funkcji, więc śledzi takie rzeczy jak bieżąca figura i obszar kreślenia, a funkcje kreślenia są kierowane do bieżących osi (proszę zauważyć, że "osie" tutaj i w większości miejsc w dokumentacji odnosi się do części osiowej figury, a nie ścisłego terminu matematycznego dla więcej niż jednej osi). Generowanie wizualizacji za pomocą `pyplot` jest bardzo szybkie:

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.ylabel('etykieta osi Y')
plt.xlabel('etykieta osi X')
plt.show()
```



Dlaczego oś x ma zakres od 0-3, a oś y od 1-4?

Jeśli podasz pojedynczą listę lub tablicę do wykreślenia, matplotlib zakłada, że jest to sekwencja wartości `y` i automatycznie generuje dla Ciebie wartości `x`. Ponieważ zakresy Pythona zaczynają się od 0, domyślny wektor `x` ma taką samą długość jak `y`, ale zaczyna się od 0. Stąd wartości `x` to `[0, 1, 2, 3]`. `Plot` jest uniwersalną funkcją i przyjmuje dowolną liczbę argumentów. Na przykład, aby wykreślić `x` względem `y`, można napisać:

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.show()
```

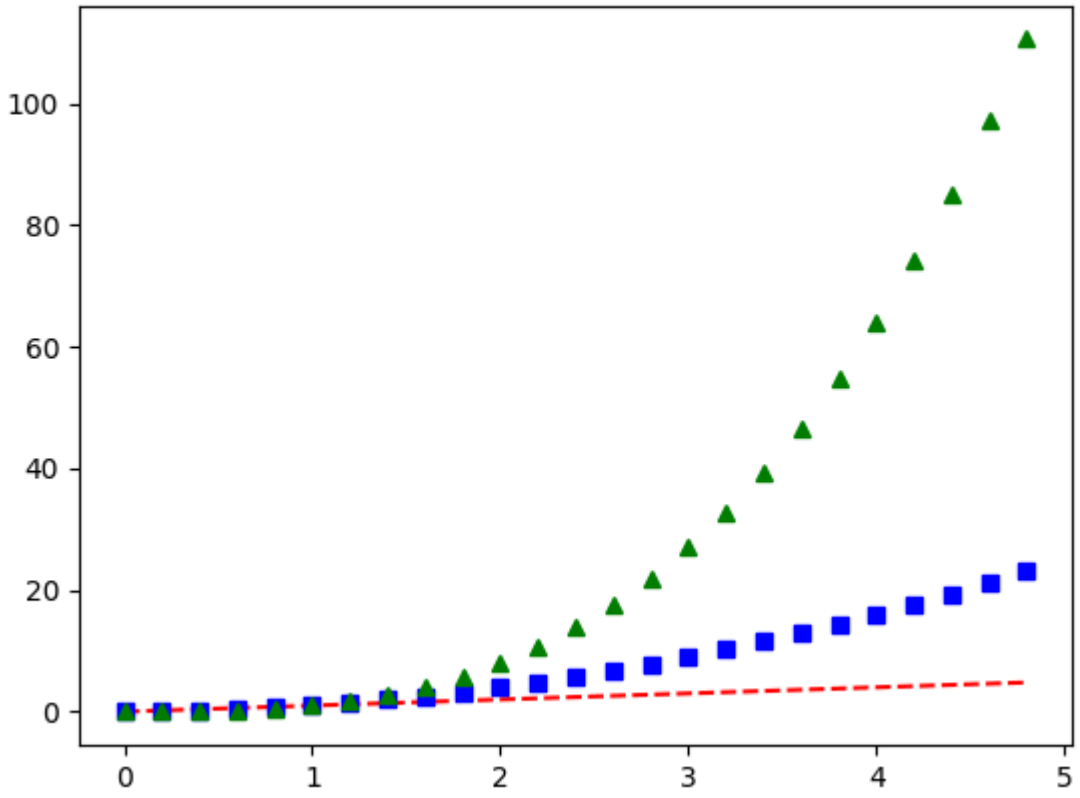
Formatowanie stylu wykresu

Dla każdej pary argumentów `x`, `y` istnieje opcjonalny trzeci argument, który jest łańcuchem formatu wskazującym kolor i typ linii wykresu. Litery i symbole łańcucha formatu pochodzą z MATLABa, i łączy się łańcuch koloru z łańcuchem stylu linii. Domyślnym łańcuchem formatu jest '`b-`', który jest niebieską linią ciągłą. Na przykład, aby wykreślić wcześniejszy wykres używając czerwonych punktów, należy wydać polecenie

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'r.')
plt.axis([0, 6, 0, 20])
plt.show()
```

Pełna lista stylów linii i łańcuchów formatowych znajduje się w dokumentacji polecenia plot: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot. Funkcja `axis` w powyższym przykładzie przyjmuje listę `[xmin, xmax, ymin, ymax]` i określa zakresy wartości przedstawiane na osiach. Gdyby `matplotlib` był ograniczony do pracy z listami, byłby dość bezużyteczny do przetwarzania liczb. W rzeczywistości wszystkie sekwencje są wewnętrznie konwertowane na tablice `numpy`. Poniższy przykład ilustruje wykreślanie kilku linii o różnych stylach formatowania w jednym wywołaniu funkcji przy użyciu tablic.

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

# równomiernie próbkowany czas w odstępach 200ms
t = np.arange(0., 5., 0.2)

# czerwone kreski, niebieskie kwadraty i zielone trójkąty
plt.plot(t, t, 'r--', t, t**2, 'bs', t,t**3,'g^')
plt.show()
```



Rysowanie kilku wykresów w jednym obszarze roboczym może odbywać się przez

- wielokrotne użycie funkcji `plot`

```
y1 = [2, 4, 6]
plt.plot(x, y1, 'bo')
plt.plot(x, [value * 3 for value in y1], 'go')
plt.show()
```

- `x` i/lub `y` są tablicami 2D, wtedy dla każdej kolumny zostanie narysowany osobny zestaw danych

```
x = [1, 2, 3]
y = np.array([[1,2],[3,4],[5,6]])
plt.plot(x, y)
plt.show()
```

```
x = [1, 2, 3]
y = np.array([[1, 2], [3, 4], [5, 6]])
for col in range(y.shape[1]):
    plt.plot(x, y[:, col])
plt.show()
```

- określenie wielu zestawów składających się z współrzędnych `[x]`, `y` oraz z stringa definiującego kolor i styl linii [fmt]:

```
x = [1, 2, 3]
y1 = [2, 4, 6]
plt.plot(x, y1, 'bo', x, [value * 3 for value in y1], 'go')
plt.show()
```

Dodatkowo dla serii danych można parametry tj. szerokość linii, etykietę legendy i inne

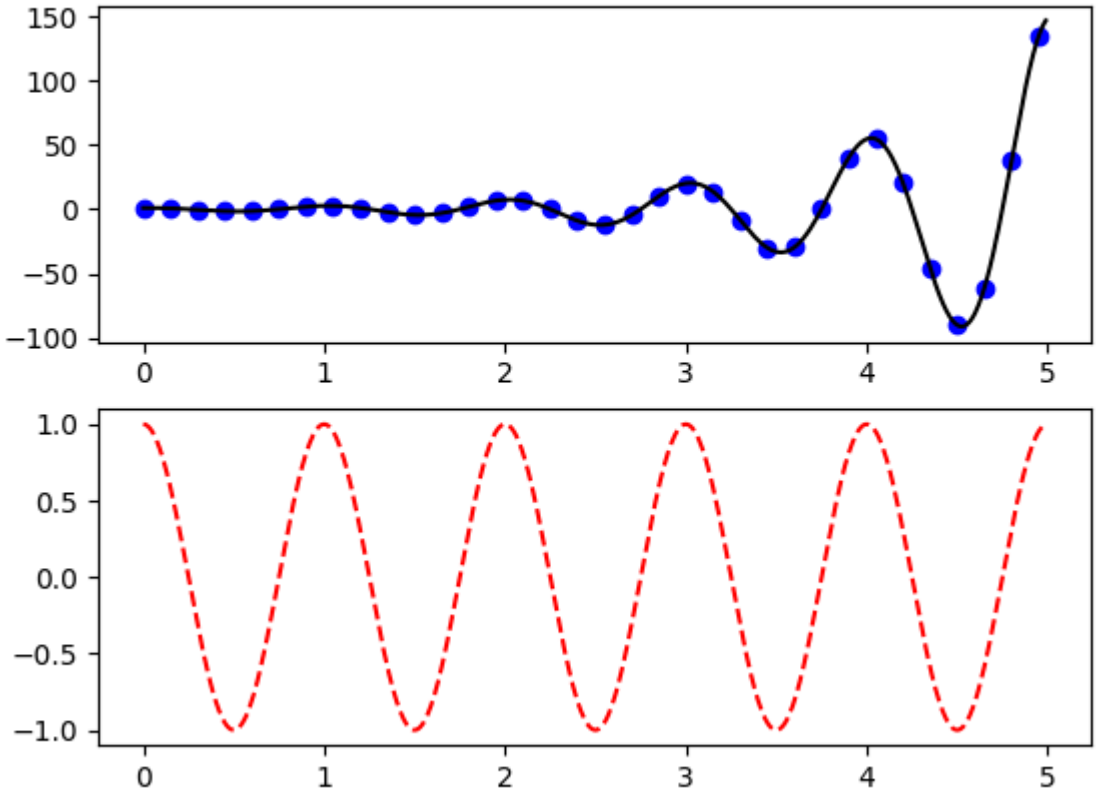
```
plt.plot([1, 2, 3], [1, 2, 3], 'go-', label='line 1', linewidth=4)
plt.plot([1, 2, 3], [1, 4, 9], 'md:', label='line 2', markersize=15)
plt.legend()
plt.show()
```

Więcej informacji dotyczących legend pod adresem https://matplotlib.org/stable/tutorials/intermediate/legend_guide.html#sphxglr-tutorials-intermediate-legend-guide-py

Praca z wieloma figurami i osiami

MATLAB, oraz `pyplot`, umożliwiają wykorzystanie bieżącego obszaru roboczego do rysowania wykresów względem osi (jeden pod drugim lub obok drugiego). Wszystkie funkcje kreślenia odnoszą się do bieżących osi. Funkcja `gca` zwraca bieżące osie (instancja `matplotlib.axes.Axes`), a `gcf` zwraca bieżącą figurę (instancja `matplotlib.figure.Figure`). Normalnie nie musisz się tym przejmować, ponieważ wszystko jest załatwiane w tle. Poniżej znajduje się skrypt tworzący dwa podwykresy.

```
def f(t):
    return np.exp(t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.15)
t2 = np.arange(0.0, 5.0, 0.01)

plt.figure()
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```



Wywołanie funkcji `figure` jest opcjonalne, ponieważ figura zostanie utworzona, jeśli nie istnieje, tak samo jak oś zostanie utworzona (odpowiednik jawnego wywołania `subplot()`), jeśli nie istnieje. Wywołanie `subplot` określa `numrows, numcols, plot_number,` gdzie `plot_number` mieści się w zakresie od `1` do `numrows*numcols` . Przecinki w wywołaniu `subplot` są opcjonalne, jeśli `numrows*numcols<10` . Zatem `subplot(211)` da taki sam rezultat jak `subplot(2, 1, 1)` . Możesz stworzyć dowolną liczbę podwykresów i osi. Jeśli chcesz umieścić oś ręcznie, tj. nie na siatce prostokątnej, użyj `axes` , który pozwala określić położenie jako `axes([left, bottom, width, height])` , gdzie wszystkie wartości są we współrzędnych ułamkowych (0 do 1).

Przykład ręcznego umieszczania osi
https://matplotlib.org/stable/_downloads/fbec90da3a9f58258ab121e0d2037693/axes_demo.py

Przykład z dużą ilością podwykresów
https://matplotlib.org/stable/gallery/subplots_axes_and_figures/subplots_demo.html

Inne typy wykresów

Kołowy

```
import matplotlib.pyplot as plt

# Wykres kołowy, w którym plasterki będą uporządkowane i wykreślone w kierunku przeciwnym do r
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
```

```
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0) # wyeksponowany tylko drugi plaster (tj. "Hogs")

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=True, startangle=90)
ax1.axis('equal') # Równe proporcje zapewniają, że wykres jest rysowany jako okrąg.

plt.show()
```

Inne warianty wykresów kołowych dostępne pod adresem
https://matplotlib.org/stable/gallery/pie_and_polar_charts/index.html

Słupkowy

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

fruits = ['apple', 'blueberry', 'cherry', 'orange']
counts = [40, 100, 30, 55]
bar_labels = ['red', 'blue', '_red', 'orange']
bar_colors = ['tab:red', 'tab:blue', 'tab:red', 'tab:orange']

ax.bar(fruits, counts, label=bar_labels, color=bar_colors)

ax.set_ylabel('fruit supply')
ax.set_title('Fruit supply by kind and color')
ax.legend(title='Fruit color')

plt.show()
```

Inne warianty wykresów słupkowych i liniowych:
https://matplotlib.org/stable/gallery/lines_bars_and_markers/index.html

Wykres 3D

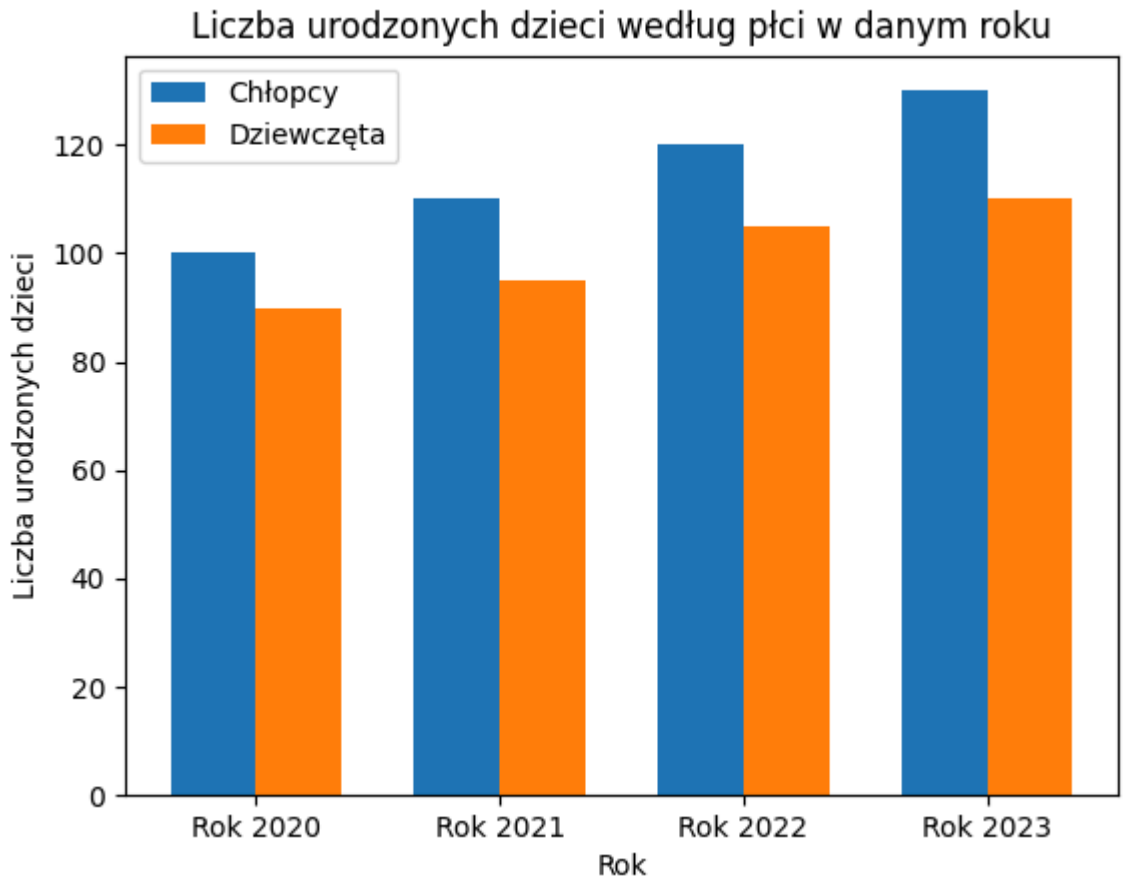
```
import matplotlib.pyplot as plt
import numpy as np

ax = plt.axes(projection="3d")
u = np.linspace(0, 2 * np.pi, 100)
v = np.linspace(0, np.pi, 100)
r = 10
x = r * np.outer(np.cos(u), np.sin(v))
y = r * np.outer(np.sin(u), np.sin(v))
z = r * np.outer(np.ones(np.size(u)), np.cos(v))
```

```
ax.plot_surface(x, y, z, rstride=5, cstride=5, cmap=plt.cm.coolwarm)
plt.show()
```

Inne warianty wykresów 3D: https://matplotlib.org/stable/gallery/lines_bars_and_markers/index.html

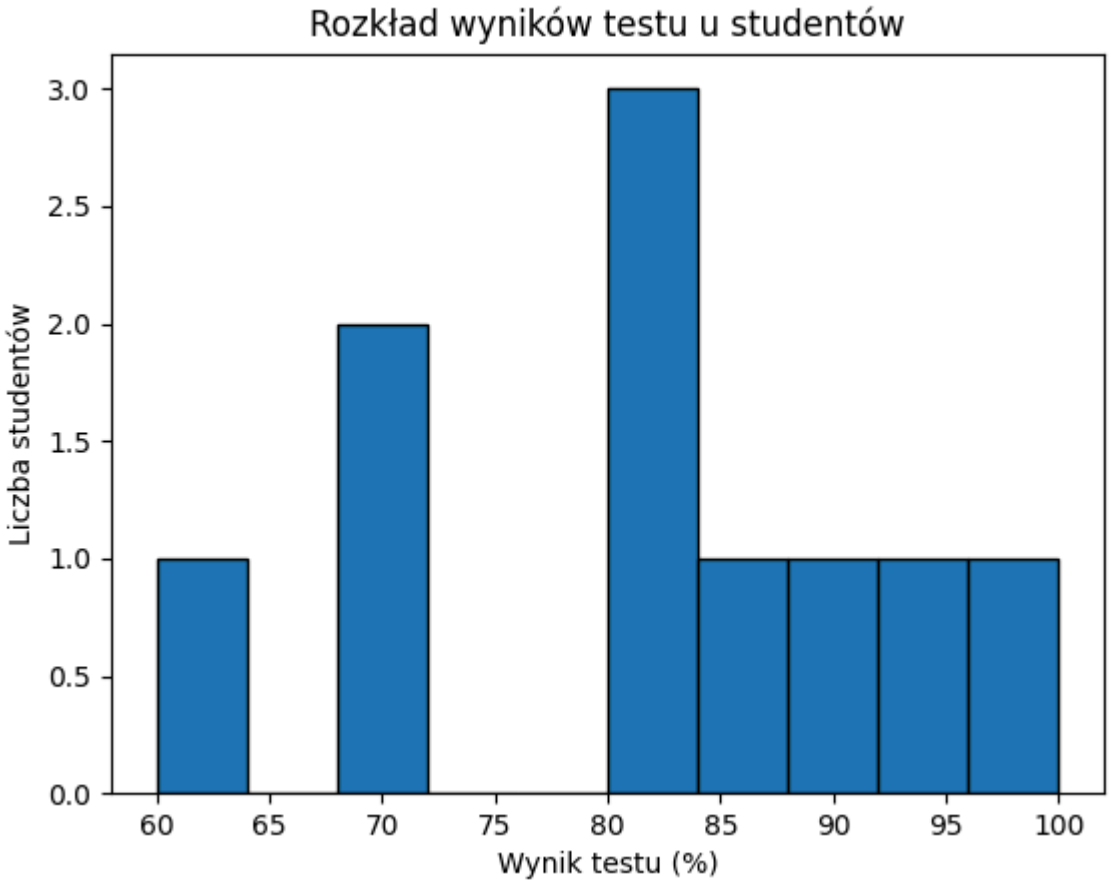
Zadania do wykonania

1. Wypróbuj kod z listingów znajdujących się w instrukcji i sprawdź ich działanie.
2. Utwórz wykres słupkowy przedstawiający temperatury w poszczególnych dniach jakie wystąpiły w ciągu wybranego tygodnia. Temperatury możesz odczytywać z listy/tablicy.
3. Narysuj wykres funkcji `y = x^2` dla `x` z zakresu od `-5` do `5`.
4. Narysuj wykres funkcji `sin(x)` i `cos(x)` dla `x` z zakresu od `0` do `2π`.
5. Narysuj wykres 3D powierzchni `z=x^2+y^2` dla `x` z zakresu od `-2` do `2` i `y` z zakresu od `-2` do `2`.
6. Narysuj wykres słupkowy przedstawiający liczbę urodzonych dzieci według płci w danym roku. Przyjmij, że dane na temat urodzeń to lista krotek np. `dane = [(100, 90), (110, 95), (120, 105), (130, 110)]` Dla podanej listy wykres może wyglądać następująco:

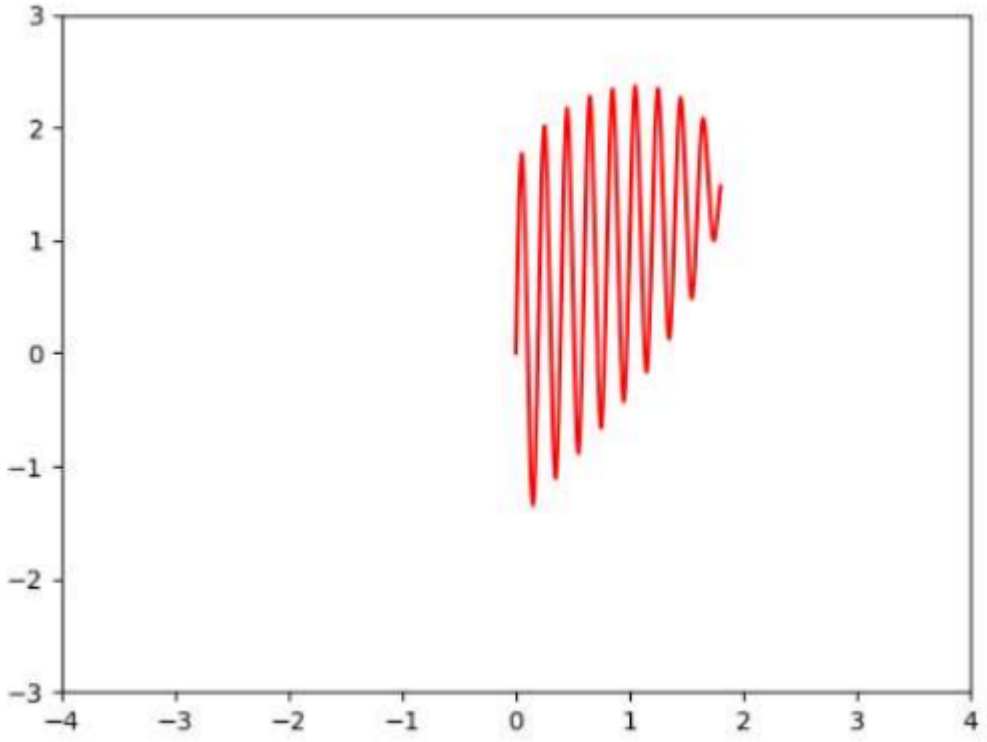


7. Narysuj wykres punktowy przedstawiający zależność masy ciała od wzrostu dla grupy osób.

8. Narysuj wykres kołowy przedstawiający rozkład procentowy różnych gatunków owoców w koszu. Przyjmij, że dane na temat owoców to lista krotek np. `data = [('jabłka', 30), ('gruszki', 20), ('śliwki', 15), ('banany', 25), ('cytryny', 10)]`
9. Narysuj histogram rozkładu wyników testu u studentów. Przyjmij, że dane odnośnie do wyników to lista przechowująca procent uzyskanych punktów przez studentów np. `dane = [60, 70, 80, 90, 100, 70, 80, 80, 85, 95]` Dla podanej listy histogram będzie wyglądał następująco:



10. Używając funkcji `sine(x)` narysuj wykres liniowy, powinien on wyglądać jak na prezentowanym rysunku:



Funkcja `sine(x)`

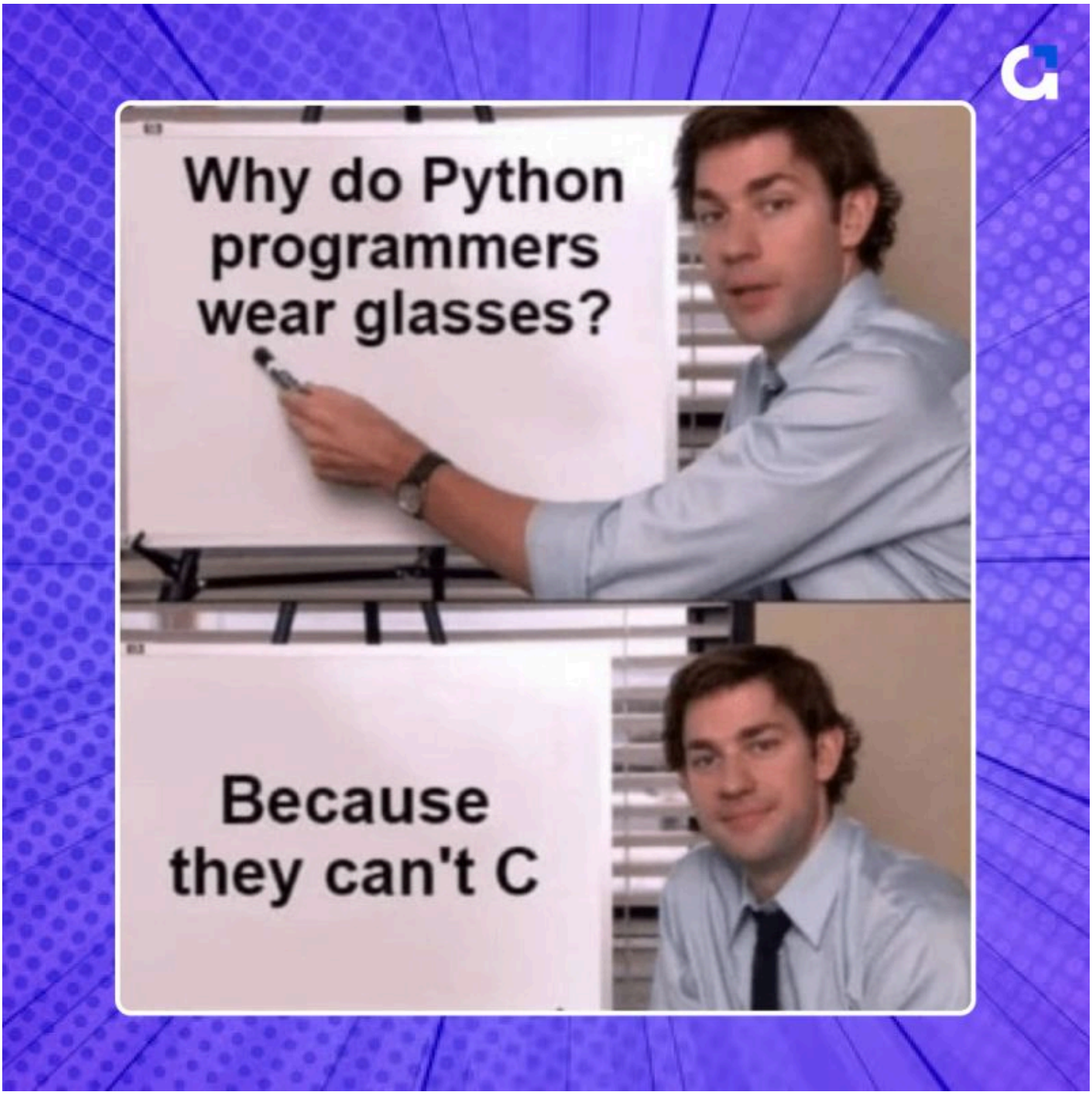
```
def sine(x):
    return np.power(x, (2.0 / 3)) + 0.9 * (3.3 - x ** 2) ** (1 / 2) * np.sin(10 * np.pi * x)
```

Następnie dodaj do wykresu odbicie lustrzane narysowanej funkcji, aby utworzyć pełne serduszko.

11. Poniższy kod wyświetla mapę temperatur dla 5 europejskich miast, temperatury pobierane są z API przeanalizuj kod i zmodyfikuj go tak aby wynik był wzbogacony o co najmniej jedno miasto więcej.

```
import matplotlib.pyplot as plt
import numpy as np
import requests
import json

# Podaj swój klucz API OpenWeatherMap
api_key = "7a44285f2b6c4bee04cb011236bcb5a0"

# Utwórz listę współrzędnych geograficznych miast w Europie
coordinates = [
     (51.509865, -0.118092), # Londyn
     (48.856613, 2.352222), # Paryż
     (52.520008, 13.404954), # Berlin
     (40.416775, -3.703790), # Madryt
     (41.902782, 12.496366), # Rzym
]
```

```
towns = ['', 'Londyn', 'Paryż', 'Berlin', 'Madryt', 'Rzym']

# Pobierz dane temperatury dla każdego z miast
temperatures = []
for latitude, longitude in coordinates:
     response = requests.get(“http://api.openweathermap.org/data/2.5/weather?lat={}&lon={}&uni
     data = response.json()
     if 'main' in data:
         temperature = data['main']['temp']
         temperatures.append(temperature)
     else:
        print("Nie udało się pobrać danych dla współrzędnych: ({}, {})".format(latitude, longi

# Stwórz tablicę numpy z danymi temperatury
temperatures = np.array(temperatures)

# Stwórz mapę cieplną z danymi temperatury
temperatures = temperatures.reshape(-1, 1)
plt.imshow(temperatures, cmap='coolwarm')
plt.colorbar()

# definicja podpisów osi
ax = plt.gca()
ax.get_xaxis().set_visible(False)
ax.set_yticklabels(towns)

# Pokaż wykres
plt.show()
```


Why do Python programmers wear glasses?
Because they can't C