

PRZYPOMNIENIE Z PYTHONA

ORAZ NARZĘDZIA AI

ORGANIZACJA PRZEDMIOTU

Przedmiot „Inteligencja obliczeniowa” ma na celu przybliżenie algorytmów i technologii związanej ze sztuczną inteligencją, w szczególności modeli „miękkich”, bioinspirowanych (sieci neuronowych, algorytmów genetycznych, itp.)

Literatura do przedmiotu jest podana w sylabusie, ale zachęcam do szukania informacji w źródłach internetowych. Ta dyscyplina mocno się zmienia w obecnych czasach i książki się dezaktualizują.

Na zaliczenie laboratoriów składać się będą:

- W wysokości **33,3% oceny**: zadania oddawane prowadzącym w trakcie zajęć. Zadania należy rozwiązywać na zajęciach, a niedokończone oddawać na zajęciach kolejnych. **12 x 1 pkt (1 pkt za laby)**.
- W wysokości **66,7% oceny**: projekty, czyli większe prace domowe. Czas na wykonanie projektów będzie ustalany z prowadzącym. Każdy ze studentów wybiera indywidualny temat projektu, najlepiej zgodny z zainteresowaniami. W semestrze przewidziane są dwa projekty po **12 pkt** (w sumie **24 pkt**). Tematy projektów można wybrać samodzielnie lub skorzystać z listy sugerowanej przez prowadzących zajęcia.

Punkty z powyższych części są sumowane, a procent całości (36 pkt) wyznacza wystawianą ocenę: 5 ($\geq 90\%$), 4.5 ($\geq 80\%$), 4 ($\geq 70\%$), 3.5 ($\geq 60\%$), 3 ($\geq 50\%$), 2 ($< 50\%$).



Zaliczenie z wykładu bierze pod uwagę egzamin i ocenę/punkty z laboratoriów:

- W wysokości **25% oceny**: egzamin pisemny (na kartkach) w formie testu ABCD. Będzie dotyczył zagadnień poruszanych na wykładzie. Odbywa się w sesji egzaminacyjnej. **Max 12 pkt**.
- W wysokości **75% oceny**: laboratoria (**max 36 pkt**).

Punkty z powyższych części są sumowane, a procent całości (48 pkt) wyznacza wystawianą ocenę: 5 ($\geq 90\%$), 4.5 ($\geq 80\%$), 4 ($\geq 70\%$), 3.5 ($\geq 60\%$), 3 ($\geq 50\%$), 2 ($< 50\%$). Wówczas warto udokumentować terminowo zrobione zajęcia wysyłając je na Teams lub do repozytorium - zalecaną drogę wskaże prowadzący zajęcia.

Studenci są zobowiązani do uczestnictwa w laboratoriach. Nieobecności proszę usprawiedliwiać. W przypadku złego samopoczucia (kaszel, katar) proszę nie przychodzić na zajęcia i pracować z domu – taka nieobecność może być usprawiedliwiona, ale proszę jej nie nadużywać. Wówczas warto udokumentować terminowo zrobione zajęcia wysyłając je na teams lub do repozytorium - zalecaną drogę wskaże prowadzący zajęcia.

Oddawanie zadań:

- Studenci oddają zadania, krótko je prezentując przy prowadzącym. Można oddać częściowo zrobione zadania na laboratoriach. Niedokończone zadania zostają jako praca domowa na kolejne zajęcia. Na kolejnych zajęciach, trzeba pokazać już wszystkie zadania.
- Nie trzeba zadań nigdzie przesyłać, chyba że zażyczy sobie tego prowadzący (github? gitlab? teams?). Być może jednak warto trzymać swoje rozwiązania w chmurze/repozytorium - i jako dokumentacja pracy, i jako wyjście awaryjne rozlczenia zadań gdybyśmy nie mogli oddać ich na zajęciach.

PYTHON

Zajęcia Python na komputerach w pracowniach komputerowych jest zainstalowany. Na komputerach domowych należy go zainstalować (gdy nie ma)

- a) Pythona należy zainstalować korzystając ze strony <https://www.python.org/downloads/>. Aktualnie najlepsza wersja to chyba **3.10** (nowa, ale już sprawdzona).

Taka czysta wersja Pythona już wystarcza do programowania, ale my chcemy sobie ułatwić sprawę i zainstalować przyjemne środowisko. Po udanym zainstalowaniu Pythona, przejdź do następnego podpunktu.

- b) W Pythonie wygodnie programuje się w specjalnych środowiskach programistycznych np.

- Visual Studio Code <https://code.visualstudio.com/>
(środowisko dla wielu języków programowania, wg wielu rankingów to najlepszy program do kodowania, możliwość instalacji wielu rozszerzeń)
- PyCharm <https://www.jetbrains.com/pycharm/>
(wygodne środowisko w wersji darmowej i płatnej)
- Spyder <https://www.spyder-ide.org/> (alternatywna dla PyCharm)
- Anaconda <https://www.anaconda.com/products/individual> (wygodne zarządzanie paczkami i środowiskami)
- Jupyter Online <https://jupyter.org/> (programowanie za pomocą notebooków pythonowych, online)
- Google Colab https://colab.research.google.com/?utm_source=scs-index (notebooki pythonowe z użyciem Googla, online)

Przejrzyj linki, wybierz najwygodniejsze dla Ciebie środowisko. Ja zachęcam do Visual Studio Code, z którego sam korzystam.

ŹRÓDŁA WIEDZY O PYTHONIE

Zapoznaj się z tutorialami dotyczącymi programowania w Pythonie:

- Fajny tutorial o programowaniu w Pythonie: <https://www.w3schools.com/python/>
Oferuje nawet możliwość testowania swoich programów online.
- Jak ładnie pisać w Pythonie. Poradnik stylu PEP8:
<https://peps.python.org/pep-0008/>
- Fajna książka z zadaniami do Pythona
https://www.brianheinold.net/python/A_Practical_Introduction_to_Python_Programming_Heinold.pdf

NARZĘDZIA AI JAKO POMOC PROGRAMISTYCZNA

Żyjemy w czasach rewolucji AI, gdzie narzędzia oparte na sztucznej inteligencji wkradają się w coraz to nowe dziedziny życia. My, komputerowcy, też jesteśmy pod mocnym wpływem tych zmian - AI zaczyna nas wyręczać w wielu zadaniach. Nie powinniśmy zostawać w tyle i musimy zacząć korzystać z dobodziejstw AI jako pomocy w programowaniu. Istnieje wiele narzędzi, które mogą nam pomóc. Wymieńmy kilka popularnych.

CHATGPT 3.5

ChatGPT 3.5 (<https://chat.openai.com/>) – Każdy zna to narzędzie, a kto nie, ten trąba 😊 Jeśli jeszcze nie posiadasz konta na stronie, zarejestruj się. Wersja 3.5 jest darmowa i całkiem dobra. Poniżej przedstawiam kilka pomysłów na to jak można wykorzystać ChatGPT do ułatwienia sobie życia. Jeśli masz chęć, to przetestuj możliwości chata na pomysłach przedstawionych niżej lub podobnych.

Zauważ, że prompty pisałem w języku angielskim, który polecam bardziej niż polski. Po polsku jednak też chat daje radę.

1. Przedstaw swój problem jasno i czytelnie i **poproś o kompletny program** w Pythonie. Zwykle ChatGPT da poprawny program, pod warunkiem, że problem jest jasno opisany. Przykład:

 You
I have a list of numbers. I want a Python program that computes a mean of these numbers and prints it. Please display the Python code. Use some example input that and show the computed output.

Example Input:

```
css Copy code  
number_list = [10, 20, 30, 40, 50]
```

 ChatGPT
Sure! Here's a Python program that calculates the mean (average) of a list of numbers and prints the result. I'll provide you with both the code and an example input with the computed output:

```
python Copy code  
  
# Function to calculate the mean of a list of numbers  
def calculate_mean(numbers):  
    if len(numbers) == 0:  
        return None # To handle empty lists  
    else:  
        total = sum(numbers)  
        mean = total / len(numbers)  
        return mean  
  
# Example input  
number_list = [10, 20, 30, 40, 50]  
  
# Calculate the mean and print the result  
mean_result = calculate_mean(number_list)  
if mean_result is not None:  
    print("Mean:", mean_result)  
else:  
    print("The list is empty.")
```

Computed Output:

```
makefile Copy code  
Mean: 30.0
```

In this example, the program calculates the mean of the list `[10, 20, 30, 40, 50]`, which is $(10 + 20 + 30 + 40 + 50) / 5 = 30.0$, and prints the result as "Mean: 30.0".



2. Jeśli to potrzebne to kontynuuj rozmowę i poproś o ulepszenia i **rozszerzenia programu**.

 You
I forgot that I also need the standard deviation. Please add this to the program and print it together with the mean. Show me the complete updated program.

 ChatGPT
Certainly! Here's the updated Python program that calculates both the mean and standard deviation of a list of numbers and prints both results:

```
python Copy code  
  
import math  
  
# Function to calculate the mean of a list of numbers  
def calculate_mean(numbers):  
    if len(numbers) == 0:  
        return None # To handle empty lists  
    else:  
        total = sum(numbers)  
        mean = total / len(numbers)  
        return mean  
  
# Function to calculate the standard deviation of a list of numbers  
def calculate_std_deviation(numbers):  
    if len(numbers) <= 1:  
        return None # To handle lists with fewer than 2 elements  
    else:  
        mean = calculate_mean(numbers)  
        squared_diff_sum = sum((x - mean) ** 2 for x in numbers)
```

```
# Example input  
number_list = [10, 20, 30, 40, 50]  
  
# Calculate the mean and standard deviation, then print the results  
mean_result = calculate_mean(number_list)  
std_deviation_result = calculate_std_deviation(number_list)  
  
if mean_result is not None:  
    print("Mean:", mean_result)  
else:  
    print("The list is empty.")  
  
if std_deviation_result is not None:  
    print("Standard Deviation:", std_deviation_result)  
else:  
    print("The list has fewer than 2 elements, so standard deviation can't be calculated")
```

Example Input:

```
css Copy code  
number_list = [10, 20, 30, 40, 50]
```

Computed Output:

```
yaml Copy code  
Mean: 30.0  
Standard Deviation: 15.811388300841556
```

3. Poproś o debugging (**poprawa błędów składniowych**) jeśli nie wiesz dlaczego program nie działa. Po prosu wklej treść programu i błędy z terminala i poproś o poprawę.



You

I have written the following program:
a = [1, 2, 3]
s = sum(a)
print(s)
Unfortunately the program does not work. I get an error:
s = sum(a)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
How should I correct it?



ChatGPT

The error you're encountering is because you're trying to use the `sum()` function to add up the elements of a list `a`, but the elements in `a` are strings, not numbers. To correct the program and sum the elements as integers, you need to convert the strings to integers before using `sum()`. You can do this using a loop or a list comprehension. Here's an example using a list comprehension:

```
python Copy code  
  
a = ['1', '2', '3']  
integers = [int(x) for x in a] # Convert the strings to integers  
s = sum(integers)  
print(s)
```

This code will first convert the strings in the list `a` to integers and store them in a new list called `integers`, and then it will calculate the sum of those integers and print the result, which will be `6` in this case.

4. Czasami błędy są semantyczne a nie syntaktyczne. Program działa, ale nie tak jak chcemy. Nie możemy znaleźć błędu. Al poradzi sobie często ze **znalezieniem błędów semantycznych**. Przykład poniżej:



You

I have written a program to compute a mean of numbers:
a = [1, 2, 3, 4]
mean = sum(a) / 3
print(mean)
Unfortunately it doesn't work. It gives the result 3.333333333333335 but it should be 2.5. Where is the error?



ChatGPT

The error in your code is in the line where you calculate the mean. You are dividing the sum of the numbers in list `a` by 3, but you should be dividing it by the number of elements in the list to get the correct mean. In your case, there are 4 elements in the list `a`, so you should divide by 4, not 3. Here's the corrected code:

```
python Copy code  
  
a = [1, 2, 3, 4]  
mean = sum(a) / len(a)  
print(mean)
```

This will correctly calculate the mean as 2.5.



5. A jeśli nie rozumiesz kodu, poproś Chata o objaśnienia, linijka po linijce.



You

I don't know what is `len(a)`. Can you explain and give some examples?



ChatGPT

Certainly! `len(a)` is a built-in Python function that returns the number of elements in an iterable, such as a list, tuple, or string. It stands for "length."

Here are some examples to help you understand how `len()` works:

- Length of a List:

```
python
Copy code

a = [1, 2, 3, 4]
length_of_a = len(a)
print(length_of_a) # Output will be 4
```

- Wreszcie, AI może poprawiać programy, by były bardziej czytelne, podzielone na funkcje, opatrzone komentarzami. Przykład:



You

I have written a program to compute mean for a list of numbers and used it to compute 3 results. Can you rewrite it in a clear comprehensive way, including comments and functions? The program is:

```
a = [1, 2, 3, 4]
mean = sum(a) / len(a)
print(mean)
b = [2, 6, 8]
mean2 = sum(b) / len(b)
print(mean2)
b = [10, 20, 30]
mean3 = sum(c) / len(c)
print(mean3)
```



Copy code

```
# Define a function to calculate the mean of a list
def calculate_mean(numbers):
    # Check if the list is empty to avoid division by zero
    if len(numbers) == 0:
        return None # Return None if the list is empty
    else:
        # Calculate the mean by summing the elements and dividing by the length
        mean = sum(numbers) / len(numbers)
        return mean # Return the calculated mean

# Define your lists
a = [1, 2, 3, 4]
b = [2, 6, 8]
c = [10, 20, 30]

# Calculate and print the means for each list
mean1 = calculate_mean(a)
mean2 = calculate_mean(b)
mean3 = calculate_mean(c)

# Check if the mean is not None before printing
if mean1 is not None:
    print(f"Mean of list 'a': {mean1}")

if mean2 is not None:
    print(f"Mean of list 'b': {mean2}")

if mean3 is not None:
    print(f"Mean of list 'c': {mean3}")
```

Małe ostrzeżenie! ChatGPT, oraz inne narzędzia AI, mogą popełniać błędy. Uważaj podczas korzystania z nich. Sprawdź dwa razy czy wyniki zwracane przez generowane programy są poprawne.

CHATGPT 4

ChatGPT 4.0 jest potężniejszym bratem ChatGPT 3.5. Niestety, ta wersja jest płatna (20\$/miesiąc). Ma kilka fajnych funkcjonalności, których nie ma 3.: załączanie plików, uruchamianie plików w wirtualnym środowisku i zwracanie gotowych wyników, generowanie wykresów i obrazów. Przykład poniżej ilustruje załączenie pliku csv z bazą danych medyczną i wygenerowanie wykresu.

pregnant-times	glucose-concentr	blood-pressure	skin-thickness	insulin	mass-index	pedigree-func	age	class
6	148	72	35	0	33.6	0.627	50	tested_positive
1	85	66	29	0	26.6	0.351	31	tested_negative
8	183	64	0	0	23.3	0.672	32	tested_positive
1	89	66	23	94	28.1	0.167	21	tested_negative
0	137	40	35	168	43.1	2.288	33	tested_positive
5	116	74	0	0	25.6	0.201	30	tested_negative
3	78	50	32	88	31	0.248	26	tested_positive

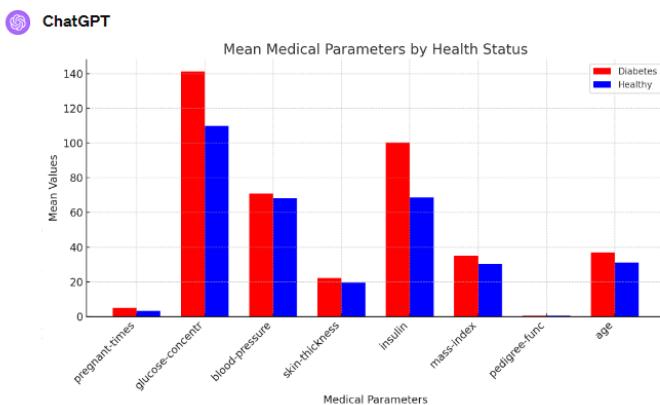
I change the model to ChatGPT4, enclose the file and give a clear prompts of what I want:

ChatGPT 4

You

diabetes.csv
Spreadsheet

I enclose a dataset of medical records for people who were healthy or had diabetes. Write a Python program that produces a barplot. On x axis we have medical parameters. For each parameter we see two bars: red for people with diabetes, and blue for healthy people. The height of the bar is the mean of the parameter for this group of people. Show me the completed graph and python code.



Here's the bar plot generated from the dataset you provided. The plot shows the mean values of various medical parameters for two groups: individuals with diabetes (in red) and healthy individuals (in blue). Each medical parameter is represented on the x-axis, and the height of each bar indicates the mean value of that parameter for the respective group.

The Python code used to generate this plot is as follows:

```
python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Load the dataset
file_path = '/mnt/data/diabetes.csv'
data = pd.read_csv(file_path)

# Grouping the data by 'class' and calculating mean for each parameter
grouped_data = data.groupby('class').mean()

# Transposing the grouped data for easier plotting
transposed_data = grouped_data.transpose()

# Extracting parameters and their means for plotting
parameters = transposed_data.index
means_positive = transposed_data['tested_positive']
means_negative = transposed_data['tested_negative']
```

GITHUB COPILOT

Github Copilot to bardzo fajne narzędzie, korzystające z ChatGPT 3.5 działające jako wtyczka do środowiska programistycznego (np. naszego VSC). Ma wiele ciekawych funkcjonalności i zalecam zatrzymać się na ich stronie:

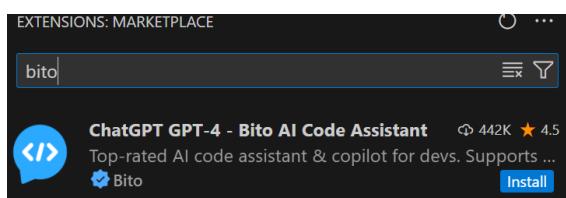
<https://github.com/features/copilot>

Licencja kosztuje 10\$ miesięcznie, ale studenci i nauczyciele mogą ją dostać za darmo po weryfikacji, która odbywa się chyba tutaj: https://education.github.com/discount_requests/application

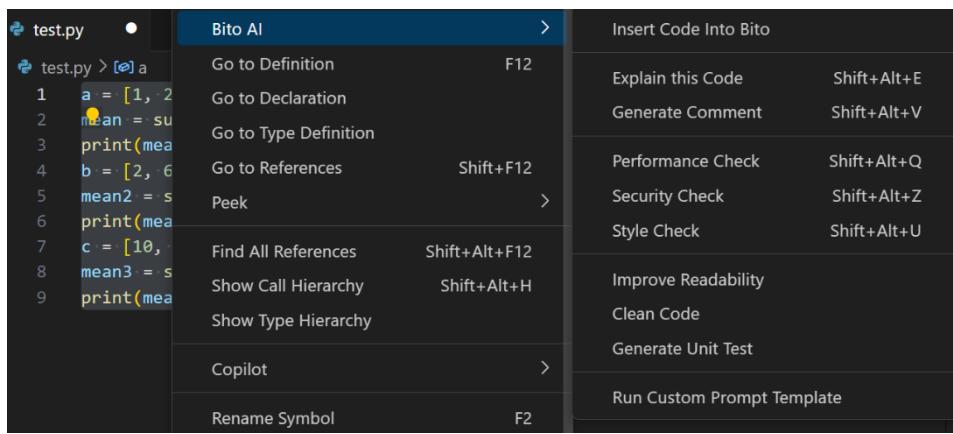
Ja (G.M.), nie korzystałem z tego narzędzia i nie mogę powiedzieć czy działa dobrze, ale ma ono bardzo dobre opinie. 😊

BITO

Podobne do Github Copilot narzędzie to Bito, ponoć działające na ChatGPT4 i w dodatku w większości darmowe. Zainstalowałem i ma kilka fajnych features. Zainstaluj wtyczkę i zobacz co dla Ciebie:



Po instalacji można korzystać z narzędzi: chat, wyjaśnienie kodu, poprawa stylu, itp. Wystarczy zaznaczyć kod programu, kliknąć PPM i wybrać BITO.



BITO poprawił ten brzydki program:

```
a = [1, 2, 3, 4]
mean = sum(a) / len(a)
print(mean)
b = [2, 6, 8]
mean2 = sum(b) / len(b)
print(mean2)
c = [10, 20, 30]
mean3 = sum(c) / len(c)
print(mean3)
```

do tego:

```
a = [1, 2, 3, 4]
mean_a = sum(a) / len(a)

b = [2, 6, 8]
mean_b = sum(b) / len(b)

c = [10, 20, 30]
mean_c = sum(c) / len(c)

print(mean_a)
print(mean_b)
print(mean_c)
```

a po performance check zmieniło się do takiego

```
def calculate_mean(lst):
    if len(lst) == 0:
        return None
    return sum(lst) / len(lst)

a = [1, 2, 3, 4]
mean = calculate_mean(a)
print(mean)

b = [2, 6, 8]
mean2 = calculate_mean(b)
print(mean2)

c = [10, 20, 30]
mean3 = calculate_mean(c)
print(mean3)
```

ZADANIA

Pora na dwa zadania, które przetestują czy umiesz korzystać z pythonowych struktur (pętle, if), wbudowanych modułów (math, datetime), tworzenia wykresów (matplotlib) oraz czy skutecznie korzystasz z narzędzi AI 😊

ZADANIE 1: BIORYTMY



problemów

Niektórzy wierzą, że pewne cykle dyktują stan naszego fizycznego, emocjonalnego i intelektualnego samopoczucia przez całe życie. Cykle te nazywane są biorytmami i jeśli zobrazujemy je na osi czasu, przyjmują one formę fal sinusoidalnych o różnych częstotliwościach.

- **Fizyczna fala:** Ta fala działa w 23-dniowym cyklu i opisuje nasze samopoczucie fizyczne: energię, siłę, witalność.

$$y_p = \sin\left(\frac{2\pi}{23}t\right)$$

- **Emocjonalna fala:** Płynąca w 28-dniowym cyklu, ta fala wpływa na nasze uczucia i nastroje (spokój, złość, melancholię, wesołość).

$$y_e = \sin\left(\frac{2\pi}{28}t\right)$$

- **Intelektualna fala:** Ten 33-dniowy cykl rządzi naszymi zdolnościami umysłowymi: szybkością, kreatywnością, zdolnością rozwiązywania

$$y_i = \sin\left(\frac{2\pi}{33}t\right)$$

Trzy fale rozpoczynają się w dniu twoich narodzin ($t = 0$) i następnie fluktuują na różnych częstotliwościach. Jeśli wartości zbliżają się do 1, czujesz się świetnie. Jeśli zbliżają się do -1, czujesz się okropnie. Aby uzyskać swój wynik, wystarczy wprowadzić numer dnia t od dnia twoich narodzin do wzorów.

Zadanie jest następujące:

- Napisz program w Pythonie, który spyta użytkownika o imię, rok urodzenia, miesiąc urodzenia i dzień urodzenia. Następnie wita użytkownika, informuje go, który dzień ich życia jest dzisiaj, oblicza ich fizyczne, emocjonalne i intelektualne wyniki biorytmów i drukuje je. Jeśli to możliwe, zrób to bez użycia narzędzi AI. Możesz korzystać z zasobów online. Sprawdź swój biorytm! Jak się czujesz dzisiaj? 😊
- Zmodyfikuj swój program. Po obliczeniu biorytmów program powinien sprawdzić, czy są one wysokie (powyżej 0,5) i pogratulować dobrego wyniku, niskie (mniejsze niż -0,5) i pocieszyć w złym dniu. W przypadku niskiego wyniku algorytm sprawdza również, czy następny dzień przyniesie wyższy czy niższy wynik. W przypadku wyższego wyniku program powinien powiedzieć coś w rodzaju „Nie martw się. Jutro będzie lepiej!”
- Zmierz (mniej więcej) ile czasu spędziłeś/aś na pisaniu programu (a-b), łącznie z badaniami online. Napisz swoją odpowiedź jako komentarz w programie.
- Poproś ChatGPT lub inne narzędzie AI o korektę programu, lub poprawę stylistyczną. Zapisz kopię programu.
- Zacznij nową, czystą rozmowę z ChatGPT¹ (lub innym narzędziem) i spróbuj poprosić o kompletny program. Zobacz, czy ChatGPT będzie w stanie napisać program samodzielnie. Musisz skonstruować swoje polecenie w dobry, jasny, precyzyjny sposób, aby zrozumiał zadanie. Zmierz, ile czasu spędziłeś/aś na uzyskaniu poprawnego programu (czy zajęło to więcej czasu niż w punkcie (c)?)

¹ChatGPT pamięta poprzednie pytania i odpowiedzi. Jeśli chcesz wyczyścić jego pamięć z poprzednich pomysłów, powinieneś rozpoczęć nową rozmowę.

ZADANIE 2: STRZELANIE Z TREBUSZA

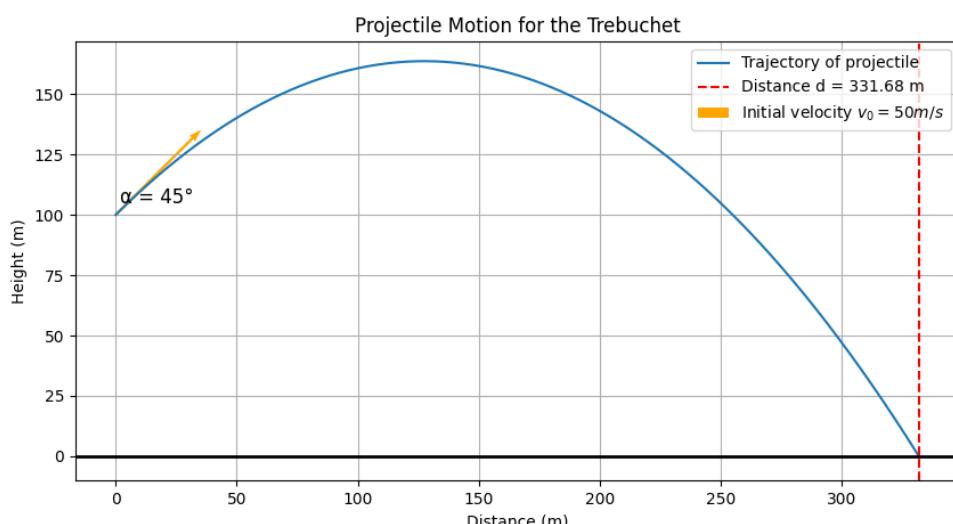


Najbardziej śmiertelionymi katapultami w średniowieczu były trebusze. Te katapulty miały obracające się ramię i do niego przymocowaną procę, która mogła wystrzeliwać ciężkie pociski z dużą prędkością i niszczyć odległe obiekty. Jednym z największych trebuszy był Warwolf używany podczas inwazji Anglii na Szkocję w latach 1298-1304 (możesz pamiętać film "Braveheart" opisujący wydarzenia z tego okresu). Ostatnim bastionem Szkotów był Zamek Stirling. Anglicy użyli Warwolfa do zniszczenia obrony zamku i wygrali bitwę.

Zobaczmy, czy możemy precyjnie strzelać z Warwolfa. Naszym zadaniem jest dostosowanie kąta strzału, aby pocisk trafił w obiekt znajdujący się w danej odległości.

Wysokość trebusza wynosiła $h=100 \text{ m}$. Początkowa prędkość pocisków wynosiła $v_0=50 \text{ m/s}$. Możemy założyć, że opór powietrza lub wiatr nie mają znaczenia w naszym przypadku. Użytkownik programu podaje **kąt α** (w stopniach), a program oblicza **odległość d** , na którą pocisk doleci.

Oto wizualizacja problemu:



Na obrazku wybrano kąt $\alpha = 45$ stopni, w wyniku czego pocisk przeleciał na odległość ponad 331 metrów.

Jak powinien działać program?

- 1) Najpierw wybierany jest cel do zniszczenia w losowej odległości w zakresie [50,340]. Ta odległość jest wyświetiana w konsoli. Cel jest zniszczony, jeśli nasz pocisk trafi w niego z marginesem błędem 5 metrów. Przykład: jeśli cel znajduje się w odległości 215 metrów, nasz pocisk powinien upaść w zakresie [210,220] metrów.
- 2) Wartości $v=50$, $h=100$ są ustalone. Użytkownik powinien określić (wprowadzić z klawiatury) jaki kąt powinien być wybrany do strzału. Jeśli pocisk chybi celu, dostajemy kolejną szansę (użyj pętli while, aby powtarzać). Jeśli pocisk trafi w cel, otrzymujemy informację „Cel trafiony!” i całkowitą liczbę naszych prób strzału, a pętla while się kończy.

Algorytm jest jasny, ale nie wspomniałem o fizycznych wzorach na ruch pocisku. W tym zadaniu interesuje nas obliczenie odległości na podstawie trzech parametrów v_0 , h , α . Możesz wybrać jeden ze sposobów, aby uzyskać wzory:

- a) Użyj swojej głowy. Tego wyjścia nie polecam, chyba że amesz czas, chęci lub wiedzę z fizyki 😊
- b) Poproś ChatGPT, aby dał ci wzory w formie LaTeX, a może nawet jako kod Pythona. Bardzo polecam spróbować tej opcji i zobaczyć, dokąd rozmowa zaprowadzi. Jeśli się nie uda, spróbuj numeru 3...
- c) Użyj już dostępnych wzorów z książek lub zasobów online (jak: <https://www.omnicalculator.com/physics/range-projectile-motion#projectile-range-formulas>). Jeśli strony nie działają, możesz również spojrzeć na koniec tego dokumentu, gdzie w formie aneksu podane są wzory.

- 3) Naszym ostatnim zadaniem jest narysowanie trajektorii pocisku Warwolf, który trafił w cel pod koniec strzelania tak jak jest to pokazane na rysunku powyżej. Możesz zignorować czerwoną przerywaną linię, wektor, tekst kąta i legendę w ramce. Zrób tylko niebieską linię trajektorii, linie siatki oraz etykiety osi i tytułu. Wykorzystaj bibliotekę matplotlib. Zapisz plik jako trajektoria.png.
- 4) Po rozwiązaniu zadania, spytaj ChatGPT lub innego narzędzia, czy program można napisać inaczej w lepszy sposób. Albo poproś o kolejne rozwiązanie tego zadania. Czy rozwiązanie zaproponowane przez AI jest inne? Lepsze? Krótsze?

ANEKS: WZORY FIZYCZNE

Jeśli prędkość startowa to $v_0 = [v_x, v_y]$, wtedy współrzędne (x,y) lecącego pocisku wyrażone w czasie opisane są znanym wzorem:

$$y = v_y t - \frac{gt^2}{2} + h \quad \text{and} \quad x = v_x t$$

Jeśli zauważymy, że $v_0 = [v_0 \cos\alpha, v_0 \sin\alpha]$ i podstawimy $t = \frac{x}{v_0 \cos\alpha}$ do formuły na y, to otrzymamy wzór na trajektorię pocisku niezależny od czasu.

$$y = -\frac{g}{2v_0^2 \cos^2 \alpha} x^2 + \frac{\sin\alpha}{\cos\alpha} x + h$$

By znaleźć, gdzie obiekt uderzy w ziemię, musimy znaleźć punkty (x,0) pasujące do powyższego wzoru. Inaczej mówiąc, musimy rozwiązać równanie kwadratowe y=0. Po wyznaczeniu delty i wyborze większego z dwóch rozwiązań mamy rozwiązanie:

$$\text{distance} = \left(v_0 \sin\alpha + \sqrt{v_0^2 \sin^2 \alpha + 2gh} \right) \cdot \frac{v_0 \cos\alpha}{g}$$