|  |  |  |  |
| --- | --- | --- | --- |
| Obraz zawierający symbol, design  Opis wygenerowany automatycznie | Politechnika Bydgoska im. J. J. Śniadeckich  **Wydział Telekomunikacji,**  **Informatyki i Elektrotechniki** | | Obraz zawierający symbol, logo, Grafika, Czcionka  Opis wygenerowany automatycznie |
| **Przedmiot** | Skryptowe języki programowania | | |
| **Prowadzący** | mgr inż. Martyna Tarczewska | | |
| **Temat** | *Więcej liczb* | | |
| **Student** | Marcin Ogórkiewicz | | |
| **Nr ćw.** | 5 | **Data wykonania** | 06.11.2023 |
| **Ocena** |  | **Data oddania spr.** | 06.11.2023 |

Zadanie 1

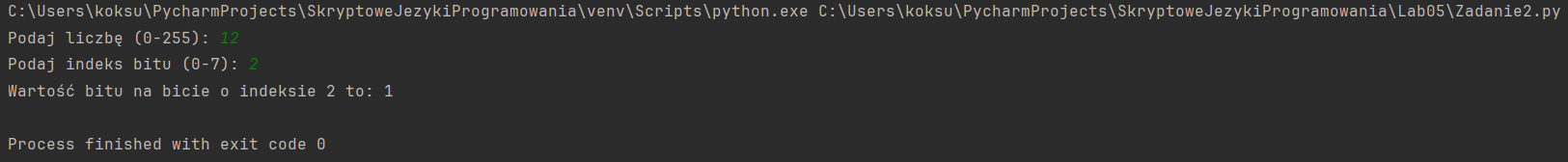
#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def get\_base() -> int:  
 while True:  
 try:  
 base = int(input("Podaj podstawę systemu liczbowego (2, 8, 10 lub 16): "))  
 if base in [2, 8, 10, 16]:  
 return base  
 else:  
 print("Podana wartość nie jest prawidłową podstawą systemu liczbowego.")  
 except ValueError:  
 print("Podaj liczbę całkowitą (2, 8, 10 lub 16).")  
  
  
def convert\_number(number: str, base: int) -> str:  
 if base == 2:  
 return bin(number)  
 elif base == 8:  
 return oct(number)  
 elif base == 10:  
 return str(number)  
 elif base == 16:  
 return hex(number)  
  
  
def main() -> None:  
 base = get\_base()  
 number = input(f"Podaj liczbę w systemie o podstawie {base}: ")  
 try:  
 number = int(number, base)  
 print(f"W postaci dwójkowej: {convert\_number(number, 2)}")  
 print(f"W postaci ósemkowej: {convert\_number(number, 8)}")  
 print(f"W postaci dziesiętnej: {convert\_number(number, 10)}")  
 print(f"W postaci heksadecymalnej: {convert\_number(number, 16)}")  
 except ValueError:  
 print("Podana liczba nie jest prawidłowa w wybranym systemie liczbowym.")  
  
  
main()

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

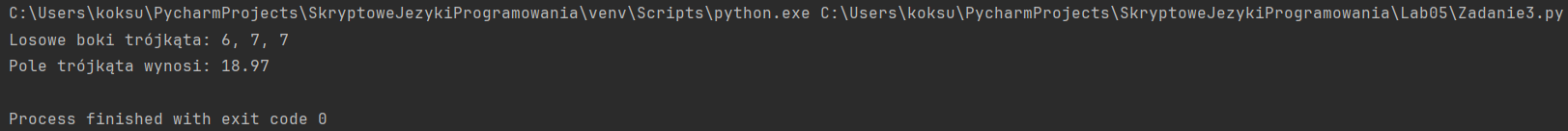
Zadanie 2

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def get\_bit\_value(number: int, bit\_index: int) -> int:  
 if 0 <= number <= 255 and 0 <= bit\_index < 8:  
 bit\_mask = 1 << bit\_index  
 bit\_value = (number & bit\_mask) >> bit\_index  
 return bit\_value  
 else:  
 print("Podano nieprawidłowe wartości. Zakres liczb to 0-255, a indeks bitu od 0 do 7.")  
  
  
def main() -> None:  
 number = int(input("Podaj liczbę (0-255): "))  
 bit\_index = int(input("Podaj indeks bitu (0-7): "))  
 result = get\_bit\_value(number, bit\_index)  
 if result is not None:  
 print(f"Wartość bitu na bicie o indeksie {bit\_index} to: {result}")  
  
  
main()



Zadanie 3

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
import random  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def calculate\_triangle\_area() -> tuple:  
 side1 = random.randint(3, 10)  
 side2 = random.randint(3, 10)  
 side3 = random.randint(3, 10)  
 if (side1 + side2 > side3) and (side1 + side3 > side2) and (side2 + side3 > side1):  
 perimeter = side1 + side2 + side3  
 half\_perimeter = perimeter / 2  
 area = (half\_perimeter \* (half\_perimeter - side1) \* (half\_perimeter - side2) \* (half\_perimeter - side3))\*\*0.5  
 return side1, side2, side3, area  
  
  
def main() -> None:  
 result = calculate\_triangle\_area()  
 if result:  
 side1, side2, side3, area = result  
 print(f"Losowe boki trójkąta: {side1}, {side2}, {side3}")  
 print(f"Pole trójkąta wynosi: {area:.2f}")  
 else:  
 print("Nie można utworzyć trójkąta z wylosowanych boków.")  
  
  
main()



Zadanie 4

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
import random  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def flip\_coin() -> int:  
 result = random.randint(0, 1)  
 return result  
  
  
def main() -> None:  
 games\_played = 0  
 games\_won = 0  
  
 while True:  
 user\_choice = input("Obstaw orła (O) lub reszkę (R), lub wprowadź 'X' aby zakończyć: ").upper()  
  
 if user\_choice == 'X':  
 break # Zakończ grę  
  
 if user\_choice not in ('O', 'R'):  
 print("Nieprawidłowy wybór. Wprowadź 'O', 'R' lub 'X'.")  
 continue  
  
 # Rzut monetą  
 coin\_result = flip\_coin()  
 if (coin\_result == 0 and user\_choice == 'O') or (coin\_result == 1 and user\_choice == 'R'):  
 print("Wygrałeś!")  
 games\_won += 1  
 else:  
 print("Przegrałeś.")  
  
 games\_played += 1  
  
 print(f"Liczba rozegranych gier: {games\_played}")  
 print(f"Liczba wygranych gier: {games\_won}")  
  
  
main()

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Zadanie 5

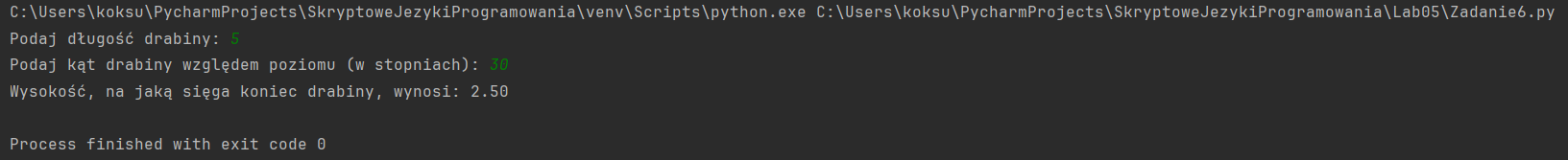
#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
import random  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def main() -> None:  
 games\_played = 0  
 games\_won = 0  
 while True:  
 user\_choice = input("Wybierz: papier (P), kamień (R), nożyce (S), lub 'X' aby zakończyć: ").upper()  
 if user\_choice == 'X':  
 break  
 if user\_choice not in ('P', 'R', 'S'):  
 print("Nieprawidłowy wybór. Wprowadź 'P', 'R', 'S' lub 'X'.")  
 continue  
 computer\_choice = random.choice(['P', 'R', 'S'])  
 print(f"Twój wybór: {user\_choice}")  
 print(f"Wybór komputera: {computer\_choice}")  
 if user\_choice == computer\_choice:  
 print("Remis!")  
 elif (user\_choice == 'P' and computer\_choice == 'R') or (user\_choice == 'R' and computer\_choice == 'S') or (user\_choice == 'S' and computer\_choice == 'P'):  
 print("Wygrałeś!")  
 games\_won += 1  
 else:  
 print("Komputer wygrał.")  
 games\_played += 1  
 print(f"Liczba rozegranych gier: {games\_played}")  
 print(f"Liczba wygranych gier: {games\_won}")  
  
  
main()

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Zadanie 6

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
import math  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def calculate\_ladder\_height(length: float, angle\_degrees: float) -> float or str:  
 if length > 0 and angle\_degrees >= 0:  
 angle\_radians = math.radians(angle\_degrees)  
 ladder\_height = length \* math.sin(angle\_radians)  
 return ladder\_height  
 else:  
 return "Nieprawidłowa długość drabiny bądź kąt"  
  
  
def main() -> None:  
 length = float(input("Podaj długość drabiny: "))  
 angle\_degrees = float(input("Podaj kąt drabiny względem poziomu (w stopniach): "))  
 ladder\_height = calculate\_ladder\_height(length, angle\_degrees)  
 print(f"Wysokość, na jaką sięga koniec drabiny, wynosi: {ladder\_height:.2f}")  
  
  
main()



Zadanie 7

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
import math  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def main() -> None:  
 for angle\_degrees in range(91):  
 angle\_radians = math.radians(angle\_degrees)  
 sin\_alfa = math.sin(angle\_radians)  
 cos\_alfa = math.cos(angle\_radians)  
 if pow(sin\_alfa, 2)+pow(cos\_alfa, 2) != 1:  
 print(f"Równanie nie zostało spełnione dla kąta {angle\_degrees}.")  
 elif angle\_degrees == 90:  
 print("Koniec sprawdzania.")  
  
  
main()

Obraz zawierający zrzut ekranu

Opis wygenerowany automatycznie

Jedynka trygonometryczna nie sprawdza się w Pythonie.

Zadanie 8

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def xor\_cipher(text: str, key: str) -> str:  
 encrypted\_text = ""  
 for i in range(len(text)):  
 char = text[i]  
 key\_char = key[i]  
 encrypted\_char = chr(ord(char) ^ ord(key\_char))  
 encrypted\_text += encrypted\_char  
 return encrypted\_text  
  
  
def main() -> None:  
 plaintext = input("Podaj tekst:")  
 keyword = input("Podaj słowo szyfrowe:")  
 xor\_text = xor\_cipher(plaintext, keyword)  
 xor\_text2 = xor\_cipher(xor\_text, keyword)  
 print("Tekst po operacji xor:", xor\_text, xor\_text2)  
  
  
main()

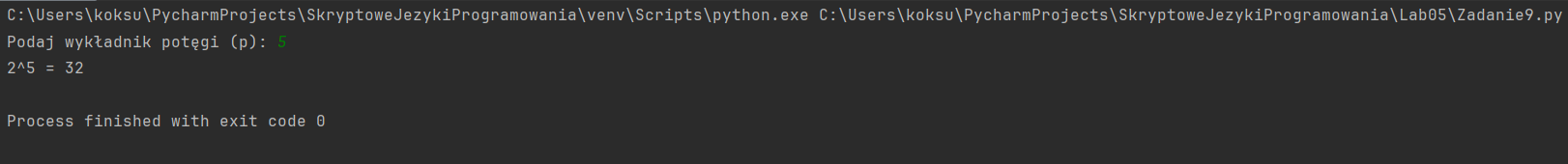
Obraz zawierający zrzut ekranu, tekst

Opis wygenerowany automatycznie

Użycie funkcji ze słowem niejawnym i tym samym słowem kodującym powoduje odszyfrowanie. Program nie zawsze jest w stanie wyświetlić niektóre znaki, ponieważ nie posiadają one graficznej reprezentacji.

Zadanie 9

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def power\_of\_two\_using\_bit\_shift(p: int) -> int:  
 if p < 0:  
 print("Potęgi liczb ujemnych nie są obsługiwane")  
 return 1 << p  
  
  
def main() -> None:  
 try:  
 exponent = int(input("Podaj wykładnik potęgi (p): "))  
 result = power\_of\_two\_using\_bit\_shift(exponent)  
 print(f"2^{exponent} = {result}")  
 except ValueError:  
 print("Nieprawidłowe dane wejściowe.")  
  
  
main()



Zadanie 10

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
import math  
import sys  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def main() -> None:  
 a = float(input("Podaj liczbę zmiennoprzecinkową (a): "))  
 print(f"Math.trunc(a): {math.trunc(a)}")  
 print(f"Math.floor(a): {math.floor(a)}")  
 print(f"Math.ceil(a): {math.ceil(a)}")  
 print(f"Math.fabs(a): {math.fabs(a)}")  
 if sys.version\_info >= (3, 9):  
 b = float(input("Podaj drugą liczbę zmiennoprzecinkową (b): "))  
 print(f"Math.lcm(a, b): {math.lcm(math.ceil(a), math.ceil(b))}")  
 print(f"Math.gcd(a, b): {math.gcd(math.ceil(a), math.ceil(b))}")  
  
  
main()

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Działanie funkcji:

Math.trunc - podaje tylko część całkowitą liczby

Math.floor - podaje liczbę w zaokrągleniu do liczby mniejszej

Math.ceil - podaje liczbę w zaokrągleniu do liczby mniejszej

Math.abs - podaje wartość bezwzględną liczby

Math.lcm - działa tylko na liczbach całkowitych, podaje najmniejszą wspólną wielokrotność

Math.gcd - działa tylko na liczbach całkowitych, podaje największy wspólny dzielnik

Zadanie 11

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
import pytest  
import math  
# stałe i zmienne globalne  
  
# funkcje  
  
  
def calculate\_ladder\_height(length: float, angle\_degrees: float) -> float:  
 angle\_radians = math.radians(angle\_degrees)  
 ladder\_height = length \* math.sin(angle\_radians)  
 return ladder\_height  
  
  
def test\_calculate\_ladder\_height\_positive\_angle():  
 result = calculate\_ladder\_height(5.0, 30.0)  
 assert pytest.approx(result, 0.01) == 2.5  
  
  
def test\_calculate\_ladder\_height\_zero\_length():  
 result = calculate\_ladder\_height(0.0, 45.0)  
 assert "Nieprawidłowa długość drabiny bądź kąt"  
  
  
def test\_calculate\_ladder\_height\_negative\_length():  
 result = calculate\_ladder\_height(-3.0, 60.0)  
 assert "Nieprawidłowa długość drabiny bądź kąt"

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Wnioski

Ćwiczenie pozwoliło mi przetrenować operacje na liczbach w języku Python. Zadania nie sprawiły mi problemu.