Knapik Jakub

**Lab12 - - - >**

Zad.1

import math

def obliczenia():

    a = int(input("Podaj wartość 'a': "))

    b = int(input("Podaj wartość 'b': "))

    x = a - b

    y = a + b

    z = a \* b

    u = a / b

    s = math.sqrt(a)

    ss = math.sqrt(b)

    print("Odejmowanie: ", x)

    print("Dodawanie: ", y)

    print("Mnożenie: ", z)

    print("Dzielenie: ", u)

    print("Pierwiastek z" + str(a) + ":", s)

    print("Pierwiastek z" + str(b) + ":", ss)

obliczenia()

Zad.2

import math

def kula(r):

    pole = math.pi \* r \*\* 2

    objetosc = 4 / 3 \* r \*\* 3

    return pole, objetosc

print(kula(int(input("Podaj r: "))))

def stozek(r, l, h):

    pole\_p = math.pi \* r \*\* 2

    pole\_b = math.pi \* r \* l

    pole\_c = pole\_b + pole\_p

    objetosc = 1 / 3 \* pole\_p \* h

    return pole\_c, objetosc

print(stozek(2 ,4, 5))

def szescian(a):

    pole\_c = 6\*a\*\*2

    objetosc = a\*\*3

    return pole\_c, objetosc

print(szescian(int(input("Podaj a: "))))

Zad.3

def cm(a):

    x = 30.48 \* a

    return str(x) + "cm"

def m(a):

    x = 0.3048 \* a

    return str(x) + "m"

def km(a):

    x = 0.0003048 \* a

    return str(x) + "km"

def mm(a):

    x = 304.8 \* a

    return str(x) + "mm"

print(cm(int(input("Podaj wartośc w stopach: "))))

print(m(int(input("Podaj wartośc w stopach: "))))

print(km(int(input("Podaj wartośc w stopach: "))))

print(mm(int(input("Podaj wartośc w stopach: "))))

Zad.4

c = int(input("Na jakiej wysokości lecimy (w metrach) ... "))

def lot(a):

    b = int(a)\*0.001

    if int(b)<5:

        return str(b) + "km to za nisko"

    else:

        return str(b) + "km jesteś bezpieczny"

print(lot(c))

Zad.5

def bmi(m,h):

    bmi = m/h\*\*2

    if bmi < 18.5:

        print("masz niedowagę")

    elif 18.5 < bmi < 24.99:

        print("waga prawidlowa")

    elif bmi > 25.00:

        print("Masz nadwagę")

bmi(64, 1.70)

**Lab13 - - - >**

Zad.1

class Auta():

    def \_\_init\_\_(self, marka, model, przebieg, moc\_silnika, kolor, cena\_auta, rodzaj):

        self.marka = marka

        self.model = model

        self.przebieg = przebieg

        self.moc\_silnika = moc\_silnika

        self.kolor = kolor

        self.cena\_auta = cena\_auta

        self.rodzaj = rodzaj

    def dragrace(self, predkosc):

        if predkosc > 90:

            wynik = self.moc\_silnika

            return "Wygrałeś!"

        elif predkosc < 90:

            wynik = self.moc\_silnika + 999

            return "Przegrałeś, silnik ci wybuchł", wynik

    def myjnia(self, cena):

        if cena > 100:

            k = self.kolor

            return "Wow jak nowe, nie wiedziałem że miałeś auto koloru "+ k

        elif cena < 100:

            b = "brudny"

            name = self.marka+' '+self.model

            return "Nie starczyło na całe mycie samochodu "+name+" jest "+ b

    def noweauto(self, taknie):

        if taknie == "tak":

            ma = input("Podaj marke: ")

            mo = input("Podaj model "+ma+' ')

            prz = input("Jaki przebieg minimalny: ")

            moc = input("Minimalna moc: ")

            kol = input("Jaki kolor lubisz: ")

            cena = input("Cena: ")

            ro = input("Rodzaj auta: ")

            nc = Auta(ma, mo, prz, moc, kol, cena, ro)

            return "Twoje nowe autko to: ", nc.marka, nc.model, nc.przebieg, nc.moc\_silnika, nc.kolor, nc.cena\_auta, nc.rodzaj

        else:

            return "Biedak..."

seat = Auta("Seat", "Ibiza", 200000, 105, "limonkowy", 14000, "heatchback")

opel\_astra = Auta("Opel", "Astra", 25000, 95, "czerwony", 54000, "combi")

ferrari = Auta("Ferrari", "Portofino", 123000, 600, "czerwony", 60000, "cabriolet")

print(seat.dragrace(22))

print(opel\_astra.myjnia(30))

print(seat.noweauto("tak"))

Zad.2

import random

class Book:

    def \_\_init\_\_(self, tytul, autor, rok\_wydania, liczba\_stron, cena):

        self.tytul= tytul

        self.autor = autor

        self.rok\_wydania = rok\_wydania

        self.liczba\_stron = liczba\_stron

        self.cena = cena

    def czytaj(self):

        return f"Metoda 1: Czytanie książki {self.tytul} autorstwa {self.autor}..."

    def promocja(self, obnizka):

        nowa\_cena = self.cena - obnizka

        return f"Metoda 2: Książka {self.tytul} została przeceniona z {self.cena}zl " \

               f"na {nowa\_cena}zl"

    def zmiana\_autora(self, nowy\_autor):

        self.autor = nowy\_autor

        return f"Metoda 3: Wprowadzono pomyłkę. Kiążkę {self.tytul} napisał {self.autor}."

    def wyswietl\_tytuly(self):

        for book in self:

            print(book.tytul)

    def reklama(self):

        return f"Już są!!! Nowe wydanie {self.tytul} w Empik!!!"

    def srednia\_cena(self):

        suma\_cen = sum(book.cena for book in self)

        sr\_cena = suma\_cen / len(self)

        return sr\_cena

book1 = Book("Harry Potter i Kamień Filozoficzny", "J.K. Rowling", 1997, 223, 35)

book2 = Book("W pustyni i w puszczy", "Henryk Sienkiewicz", 1910, 318, 25)

book3 = Book("Marsjanin", "Andy Weir", 2011, 385, 40)

book4 = Book("Władca Pierścieni: Drużyna Pierścienia", "J.R.R. Tolkien", 1954, 423, 68)

book5 = Book("The Hitchhiker's Guide to the Galaxy", "Douglas Adams", 1979, 215, 25)

books = [book1, book2, book3, book4, book5]

srednia\_cena = Book.srednia\_cena(books)

print(f"Średnia cena książek: {srednia\_cena}")

print(book1.czytaj())

print(book5.promocja(20))

print(book3.zmiana\_autora("Krzysztof Krafczyk"))

Book.wyswietl\_tytuly(books)

print(book1.reklama())

**Lab14 - - - >**

Zad.1

class Pojazd:

    def \_\_init\_\_(self, nazwa, nr\_tablicy, szybkosc):

        self.nazwa = nazwa

        self.nr\_tablicy = nr\_tablicy

        self.szybkosc = szybkosc

    def jazda(self):

        print(f"Pojazd {self.nazwa} jedzie z prędkością {self.szybkosc} km/h.")

class Auto(Pojazd):

    def \_\_init\_\_(self, nr\_tablicy,szybkosc):

        super().\_\_init\_\_('Audi',nr\_tablicy, szybkosc)

    def drift(self):

        print(f'{self.nazwa} leci bokiem przez Auchan z prędkością {self.szybkosc} km/h.')

class Motor(Pojazd):

    def \_\_init\_\_(self,nr\_tablicy, szybkosc):

        super().\_\_init\_\_('Yamaha',nr\_tablicy,szybkosc)

    def wheelie(self):

        print(f'Jednostka policji nr12: Motor {self.nazwa} z nr {self.nr\_tablicy} robi wheelie przez cały most! \n'

              f'Zgarnąć go!')

audi = Auto('DWL4874',50)

audi.drift()

yamaha = Motor('DWL6666', 60)

yamaha.wheelie()

Zad.2

class Animals:

    def \_\_init\_\_(self, gatunek, rasa, imie, typ, region\_wys, predkosc):

        self.gatunek = gatunek

        self.rasa = rasa

        self.imie = imie

        self.typ = typ

        self.region\_wys = region\_wys

        self.predkosc = predkosc

    def wystepowanie(self):

        return f'Gatunek {self.gatunek}, {self.rasa}, region występowania: {self.region\_wys} '

class Cat(Animals):

    def \_\_init\_\_(self, region\_wys):

        super().\_\_init\_\_("Kot", "Munchkin", "Gacek", "miesożerca", region\_wys, 48)

    def drapansko(self):

        return f'{self.gatunek} {self.rasa} ostrzy pazurki na drapaku.'

class Dog(Animals):

    def \_\_init\_\_(self, region\_wys, predkosc):

        super().\_\_init\_\_("Pies", "Border Collie", "Borys", "wszystkożerca", region\_wys, predkosc)

    def poscig(self, pr\_lisa):

        if self.predkosc >= 50:

            print(f'{self.imie} jest w trakcie poscigu...')

            if pr\_lisa > self.predkosc:

                print(f'Nie udało się dogonić lisa.')

            elif pr\_lisa < self.predkosc:

                print(f'Udało się złapać lisa!')

        elif self.predkosc < 50:

            print(f'{self.imie} jest za wolny na ten pościg')

class Bird(Animals):

    def \_\_init\_\_(self, imie, region\_wys):

        super().\_\_init\_\_("Ptak", "Struś", imie, "roślinożerca", region\_wys, 70)

    def ustaw\_imie(self):

        return f'Urodził się nowy struś o imieniu {self.imie}!'

class Fish(Animals):

    def \_\_init\_\_(self, region\_wys):

        super().\_\_init\_\_("Ryba", "Rekin Biały", "Dango", "miesożerca", region\_wys, 40)

kot1 = Cat("Cały Świat")

print(kot1.wystepowanie())

print(kot1.drapansko())

# borys = Dog("Europa", 50)

# borys.poscig(51)

# strus = Bird("Stasiu", "Afryka")

# print(strus.ustaw\_imie())

# ptak1 = Bird("Afryka")

# print(ptak1.wystepowanie())

#

# ryba1 = Fish("Ocean Spokojny")

# print(ryba1.wystepowanie())

Zad.3

import math

class Figura:

    def obwod(self):

        pass

    def pole(self):

        pass

class Kwadrat(Figura):

    def \_\_init\_\_(self, bok):

        self.bok = bok

    def obwod(self):

        return 4 \* self.bok

    def pole(self):

        return self.bok \*\* 2

class Prostokat(Figura):

    def \_\_init\_\_(self, bok\_a, bok\_b):

        self.bok\_a = bok\_a

        self.bok\_b = bok\_b

    def obwod(self):

        return 2 \* self.bok\_a + 2 \* self.bok\_b

    def pole(self):

        return self.bok\_a \* self.bok\_b

class Trojkat(Figura):

    def \_\_init\_\_(self, podstawa, wysokosc):

        self.podstawa = podstawa

        self.wysokosc = wysokosc

    def obwod(self):

        return 3 \* self.podstawa

    def pole(self):

        return 0.5 \* self.podstawa \* self.wysokosc

class Kolo(Figura):

    def \_\_init\_\_(self, promien):

        self.promien = promien

    def obwod(self):

        return 2 \* math.pi \* self.promien

    def pole(self):

        return math.pi \* self.promien \*\* 2

class Romb(Figura):

    def \_\_init\_\_(self, przekatna\_a, przekatna\_b):

        self.przekatna\_a = przekatna\_a

        self.przekatna\_b = przekatna\_b

    def obwod(self):

        return 4 \* math.sqrt(0.25 \* self.przekatna\_a \*\* 2 + 0.25 \* self.przekatna\_b \*\* 2)

    def pole(self):

        return 0.5 \* self.przekatna\_a \* self.przekatna\_b

class Trapez(Figura):

    def \_\_init\_\_(self, podstawa\_a, podstawa\_b, wysokosc):

        self.podstawa\_a = podstawa\_a

        self.podstawa\_b = podstawa\_b

        self.wysokosc = wysokosc

    def obwod(self):

        return self.podstawa\_a + self.podstawa\_b + 2 \* math.sqrt(

            (0.5 \* (self.podstawa\_a - self.podstawa\_b)) \*\* 2 + self.wysokosc \*\* 2)

    def pole(self):

        return 0.5 \* (self.podstawa\_a + self.podstawa\_b) \* self.wysokosc

# Przykład użycia:

print("Obliczanie pola i obwodu figur")

print("1. Kwadrat")

bok\_kwadratu = float(input("Podaj długość boku kwadratu: "))

kwadrat = Kwadrat(bok\_kwadratu)

print("Pole kwadratu:", kwadrat.pole())

print("Obwód kwadratu:", kwadrat.obwod())

**Lab15 - - - >**

Zad.1

import random

class Restaurant:

    def \_\_init\_\_(self, nazwa, rodzaj, ocena, lokalizacja):

        self.nazwa = nazwa

        self.rodzaj = rodzaj

        self.ocena = ocena

        self.lokalizacja = lokalizacja

    def o\_nas(self):

        print(f'{self.nazwa} to {self.rodzaj} z oceną {self.ocena} w miescie {self.lokalizacja}.')

class Stoisko(Restaurant):

    def \_\_init\_\_(self, nazwa, rodzaj, ocena, lokalizacja):

        super().\_\_init\_\_(nazwa, rodzaj, ocena, lokalizacja)

        self.smaki = ['czekolada', 'wanilia', 'truskawka', 'malina']

    def smaki\_lodow(self):

        print(f"Smaki w {self.nazwa} to:")

        for smaki in self.smaki:

            print(f"- {smaki}")

class CoffeShop(Restaurant):

    def \_\_init\_\_(self, nazwa, rodzaj, ocena, lokalizacja):

        super().\_\_init\_\_(nazwa, rodzaj, ocena, lokalizacja)

        self.kawy = {'espresso': 9.0, 'cappuccino': 13.0, 'latte': 15.0, 'americano': 8.0, 'flat white': 13.5}

    def menu(self):

        print(f"Menu w {self.nazwa}:")

        for kawa, cena in self.kawy.items():

            print(f"- {kawa}: {cena} PLN")

    def menu\_1niedostepna(self):

        print(f"Menu w {self.nazwa}:")

        niedostepna = random.choice(list(self.kawy.keys()))

        for kawa, cena in self.kawy.items():

            if kawa == niedostepna:

                print(f'- {kawa} jest aktualnie niedostępna')

            else:

                print(f"- {kawa}: {cena} PLN")

    def tanie(self):

        naj = min(self.kawy, key=self.kawy.get)

        cena = self.kawy[naj]

        print(f"Najtańsza kawa w {self.nazwa} to: {naj}, cena: {cena} PLN")

    def nowa\_kawa(self, nazwa, cena):

        self.kawy[nazwa] = cena

        print(f"Nowa kawa '{nazwa}' została dodana do menu.")

budka1 = Restaurant('Lody na kółkach', 'lodziarnia', '5/5 gwiazdek', 'Brzeg Dolny')

budka2 = Restaurant('Lodex', 'lodziarnia', '4/5', 'Warszawa')

budka1.o\_nas()

budka2.o\_nas()

# budka = Stoisko('Lody na kółkach', 'lodziarnia', '5/5 gwiazdek', 'Brzeg Dolny')

# budka.smaki\_lodow()

# #

# coffie = CoffeShop('Lodówka','kawiarnia','4/5 giazdek','Wołów')

# coffie.menu()

# print(' ')

# coffie = CoffeShop('Lodówka','kawiarnia','4/5 giazdek','Wołów')

# coffie.menu\_1niedostepna()

# coffie = Restaurant('Lodówka','kawiarnia','4/5 giazdek','Wołów')

# coffie.o\_nas()

# coffie = CoffeShop('Lodówka','kawiarnia','4/5 giazdek','Wołów')

# coffie.tanie()

# coffie.nowa\_kawa("double espresso", 10.0)

# coffie.menu()

Zad.2

class Shop:

    def \_\_init\_\_(self, nazwa, lok\_sklepu):

        self.nazwa\_sklepu = nazwa

        self.lok\_sklepu = lok\_sklepu

    def sprzedaj\_piwo(self, piwo):

        print(f"Sprzedano piwo {piwo.nazwa} w sklepie {self.nazwa\_sklepu} w mieście {self.lok\_sklepu}")

class Beer(Shop):

    piwa = []

    def \_\_init\_\_(self, nazwa, rodzaj, cena, procenty, ocena):

        super().\_\_init\_\_("Monopol Kasia","Wrocław")

        self.nazwa = nazwa

        self.rodzaj = rodzaj

        self.cena = cena

        self.procenty = procenty

        self.ocena = ocena

        Beer.piwa.append(self)

    def sprzedaj(self):

        super().sprzedaj\_piwo(self)

    def po\_ocenie(self):

        return sorted(self, key=lambda p: p.ocena, reverse=True)

    def po\_nazwie(self):

        return sorted(self, key=lambda p: p.nazwa)

    @classmethod

    def dodaj\_nowe\_piwo(self):

        nazwa = input("Podaj nazwę piwa: ")

        rodzaj = input("Podaj rodzaj piwa: ")

        cena = float(input("Podaj cenę piwa: "))

        procenty = float(input("Podaj procenty alkoholu w piwie: "))

        ocena = int(input("Podaj ocenę piwa (0-10): "))

        pochodzenie = input("Podaj pochodzenie piwa: ")

        nowe\_piwo = self(nazwa, rodzaj, cena, procenty, ocena, pochodzenie)

        return nowe\_piwo

piwo1 = Beer("Perła Export", "pilzner", 5, 5.2, 5)

piwo2 = Beer("Żubr", "jasny lager", 5, 6.0, 10)

piwo3 = Beer("Lech", "jasne pełne", 6, 5.2, 2)

piwo4 = Beer("Somersby", "oranżada", 7, 4.5, 1)

piwo5 = Beer("Zetecky", "dolna fermentacja", 5, 5.0, 9)

piwo6 = Beer("Romper", "kto to wie", 3, 7.0, 0, )

lista\_piw = [piwo1, piwo2, piwo3, piwo4, piwo5, piwo6]

# sort\_piw = Beer.po\_ocenie(lista\_piw)

# sort\_piw\_po\_nazwie = Beer.po\_nazwie(lista\_piw)

# for piwo in sort\_piw:

#     print(piwo.nazwa, " - ocena: ", piwo.ocena,"/10")

#

# print("\nPosortowane piwa po nazwie:")

# for piwo in sort\_piw\_po\_nazwie:

#     print(piwo.nazwa)

# nowe\_piwo = Piwo.dodaj\_nowe\_piwo()

# print("\nDodano nowe piwo:")

# print("Nazwa: ", nowe\_piwo.nazwa)

# print("Rodzaj: ", nowe\_piwo.rodzaj)

# print("Cena: ", nowe\_piwo.cena)

# print("Procenty: ", nowe\_piwo.procenty)

# print("Ocena: ", nowe\_piwo.ocena)

# print("Pochodzenie: ", nowe\_piwo.pochodzenie)

piwo1.sprzedaj()

Zad.3

class Obywatel:

    def \_\_init\_\_(self, imie, nazwisko, ulica, nr\_domu, kod\_pocztowy, miejscowosc):

        self.imie = imie

        self.nazwisko = nazwisko

        self.ulica = ulica

        self.nr\_domu = nr\_domu

        self.kod\_pocztowy = kod\_pocztowy

        self.miejscowosc = miejscowosc

    @classmethod

    def twoje\_dane(cls):

        imie = input("Podaj imie: ")

        nazwisko = input("Podaj nazwisko: ")

        ulica = input("Podaj ulice: ")

        while True:

            try:

                nr\_domu = int(input("Podaj nr domu: "))

                break

            except ValueError:

                print("Błędne dane. Numer domu musi być liczbą.")

        kod\_pocztowy = input("Podaj swój kod pocztowy: ")

        miejscowosc = input("Podaj miejscowosc: ")

        nowy\_obywatel = cls(imie, nazwisko, ulica, nr\_domu, kod\_pocztowy, miejscowosc)

        return nowy\_obywatel

    def zapis(self, nazwa\_pliku):

        with open(nazwa\_pliku, "w") as f:

            wizytowka = f"------------- \n {self.imie} {self.nazwisko} \n {self.ulica} {self.nr\_domu} \n {self.kod\_pocztowy} {self.miejscowosc} \n--------------"

            f.write(wizytowka)

            print(wizytowka)

obywatel1 = Obywatel.twoje\_dane()

obywatel1.zapis('dane1.txt')

**Lab18 - - - >**

Zad.1

with open("pantadeusz.txt") as plik:

    for linia in plik:

        print (linia.strip().split())

with open("pantadeusz.txt") as plik:

    for linia in plik:

        print (linia)

Zad.2

import linecache

wiersze = [8, 12, 60, 98, 104]

plik = 'pantadeusz.txt'

for numer\_w in wiersze:

    wiersz = linecache.getline(plik, numer\_w)

    print(wiersz)

Zad.2 (z enumerate)

import linecache

wiersze = [8, 12, 60, 98, 104]

plik = 'pantadeusz.txt'

with open(plik, 'r') as file:

    for index, numer\_w in enumerate(wiersze, start=1):

        wiersz = linecache.getline(plik, numer\_w)

        print(f"Line {index}: {wiersz}")

Zad.3

stos1 = []

stos2 = []

stos3 = []

foo1 = open('stosik1.txt', 'w')

for i in range(50):

    stos1.append(i)

    i+=1

    foo1.writelines(str(i)+' ')

foo2 = open('stosik2.txt', 'w')

for i in range(100):

    stos2.append(i)

    i+=1

    foo2.writelines(str(i)+' ')

foo3 = open('stosik3.txt', 'w')

for i in range(150):

    stos3.append(i)

    i+=1

    foo3.writelines(str(i)+' ')

**Lab19 - - - >**

Zad.1

def sprawdz\_kod(kod):

    if not kod[:2].isdigit():

        raise ValueError("Ciąg liczb jest niepoprawny")

    elif kod[2] != '-':

        raise ValueError("Ciąg liczb jest niepoprawny")

    elif not kod[3:].isdigit():

        raise ValueError("Ciąg liczb jest niepoprawny")

with open("kody\_pocztowe.txt", "a") as file:

    while True:

        kod = input("Podaj kod pocztowy (wpisz 'done' aby zakończyć): ")

        if kod == "done":

            print("Do zobaczenia :)")

            break

        try:

            sprawdz\_kod(kod)

            print("Podany kod jest poprawny")

            file.write(kod + "\n")

            print("Kod pocztowy został zapisany do pliku 'kody\_pocztowe.txt'")

        except ValueError as e:

            print("Błąd:", str(e))

**Lab20 - - - >**

Zad.1