**Zad.1**

try:

num = int(input('Enter a number:'))

for x in range(1, num + 1):

if num % x == 0:

print(x)

except ValueError:

print('You entered a wrong value')

**Zad.6**

try:

licznik1 = float(input('Podaj licznik pierwszego ułamka: '))

mianownik1 = float(input('Podaj mianownik pierwszego ułamka: '))

if mianownik1 == 0:

print('Sprzeczność')

licznik2 = float(input('Podaj licznik drugiego ułamka: '))

mianownik2 = float(input('Podaj mianownik drugiego ułamka: '))

if mianownik2 == 0:

print('Sprzeczność')

mianownik3 = int(mianownik1 \* mianownik2)

if mianownik3 == 0:

print('Sprzeczność')

licznik3 = int((mianownik3 / mianownik1) \* licznik1 + (mianownik3 / mianownik2) \* licznik2)

print(licznik3, '/', mianownik3)

except ValueError:

print('Podałeś złą wartość')

**Zad.8**

x = float(input('Podaj liczbę puntów: '))

if x == 40:

print('cel')

elif 34 <= x <= 39.9:

print('bdb')

elif 28 <= x <= 33.9:

print('db')

elif 20 <= x <= 27.9:

print('dst')

elif 16 <= x <= 19.9:

print('dop')

else:

print('ndst')

**Zad.9**

def suma(a, b):

y=0

for x in range(a, b+1):

y=y+x

return y

a = int(input('Podaj pierwszą liczbę: '))

b = int(input('Podaj ostatnią liczbę: '))

print(suma(a, b))

**Zad.12**

def fib\_iteracyjny(x):

pwyrazy = (0, 1)

a, b = pwyrazy

while x > 1:

a, b = b, a + b

x -= 1

return x

x = int(input('Podaj, który element ciągu chcesz policzyć: '))

print(fib\_iteracyjny(x))

def fib\_rekurencyjny(y):

a, b = 0, 1

for i in range(y-1):

a, b = b, a+b

return a

y = int(input('Podaj, który element ciągu chcesz policzyć: '))

print(fib\_rekurencyjny(y))

**Zad. 13**

def newton(n, k):

x = 1

for i in range(1, k + 1):

x = x \* (n - i + 1) / i

return x

n = int(input('Podaj liczbę n: '))

k = int(input('Podaj liczbę k: '))

if n == k or k == 0:

print('1')

elif n > k:

print(newton(n, k))

else:

print('Błąd, n mniejsze niż k')

**Zad.16**

import matplotlib.pyplot as plt

def create\_bar\_chart(data, labels):

num\_bars = len(data)

positions = range(1, num\_bars+1)

plt.barh(positions, data, align='center')

plt.yticks(positions, labels)

plt.xlabel('Amount')

plt.ylabel('Category')

plt.title('Weekly expenditures')

plt.grid()

plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

try:

data = int(input('Podaj ilość kategorii: '))

category = []

amount = []

for i in range(1, data + 1):

labels = str(input('Podaj kategorię: '))

category.append(labels)

a = float(input('Podaj kwotę: '))

amount.append(a)

except ValueError:

print('Podałeś złą wartość')

else:

create\_bar\_chart(amount, category)

**Zad.19**

def day\_month(month):

months = {'styczeń': '31','luty': '29','marzec': '31','kwiecień': '30','maj': '31','czerwiec': '30','lipiec': '31','siepień': '31','wrzesień': '30','październik': '31','listopad': '30','grudzień': '31'}

month = month.strip().lower()

if month not in months.keys():

print('Zła nazwa miesiaca')

else:

print(months[month])

day\_month('styczeń')

**Zad.22**

lst = []

def median(lst):

sortedLst = sorted(lst)

lstLen = len(lst)

index = (lstLen - 1) // 2

if (lstLen % 2):

return sortedLst[index]

else:

return (sortedLst[index] + sortedLst[index + 1])/2.0

median([4,2.5,3,1])

**Zad.31**

import numpy as np

def matrix(n):

return np.random.choice(np.array([0, 255]), (n,n))

n = 10

print(matrix(n))

**Zad.34**

class Kwadrat():

def \_\_init\_\_(self, a):

self.a = a

def bok(self):

return "Kwadrat o boku {}".format(self.a)

def pole(self):

return "polu {}".format(self.a\*\*2)

def obwod(self):

return "i obwodzie {}".format(self.a\*4)

k=Kwadrat(5)

print(k.bok())

print(k.pole())

print(k.obwod())

**Projekt 4**

class Martix:

def \_\_init\_\_(self, wiersze, kolumny):

self.wiersze = wiersze

self.kolumny = kolumny

def użytkownik\_wypełnia(self):

self.body = []

for i in range(self.wiersze):

init\_list = []

for j in range(self.kolumny):

element = float(input("aij = "))

init\_list.append(element)

self.body.append(init\_list)

def macierz\_z\_zerami(self):

self.body = []

for i in range(self.wiersze):

init\_list = []

for j in range(self.kolumny):

element = 0

init\_list.append(element)

self.body.append(init\_list)

def macierz\_z\_jedynkami(self):

self.body = []

for i in range(self.wiersze):

init\_list = []

for j in range(self.kolumny):

element = 1

init\_list.append(element)

self.body.append(init\_list)

def macierz\_jednostkowa(self):

if(self.wiersze == self.kolumny):

self.body = []

for i in range(self.wiersze):

init\_list = []

for j in range(self.kolumny):

if (i==j):

element = 1

else:

element = 0

init\_list.append(element)

self.body.append(init\_list)

else:

print("Liczba wierszy różni się od liczby kolumn.")

def dodawanie(self, other):

if(self.kolumny == other.kolumny and self.wiersze == other.wiersze and len(self.body) != 0 and len(other.body) != 0):

for i in range(self.wiersze):

for j in range(self.kolumny):

self.body[i][j] = self.body[i][j] + other.body[i][j]

else:

print("Liczba wierszy różni się od liczby kolumn.")

def odejmowanie(self, other):

if(self.kolumny == other.kolumny and self.wiersze == other.wiersze and len(self.body) != 0 and len(other.body) != 0):

for i in range(self.wiersze):

for j in range(self.kolumny):

self.body[i][j] = self.body[i][j] - other.body[i][j]

else:

print("Liczba wierszy różni się od liczby kolumn.")

def mnożenie\_przez\_macierz(self, other):

if(self.kolumny == other.kolumny and self.wiersze == other.wiersze and len(self.body) != 0 and len(other.body) != 0):

for i in range(self.wiersze):

for j in range(self.kolumny):

self.body[i][j] = self.body[i][j] \* other.body[i][j]

def mnożenie\_przez\_liczbe(self, a):

for i in range(self.wiersze):

for j in range(self.kolumny):

self.body[i][j] = a\*self.body[i][j]

def wypisz(self):

print(self.body)

**Projekt 5**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

titanic\_data = pd.read\_csv(r"/Users/ewelina/Desktop/CDV/I\_semestr/Programowanie\_w\_data\_science\_podstawy/titanic/train.csv")

titanic\_data.head()

"""

PassengerId Survived Pclass ... Fare Cabin Embarked

0 1 0 3 ... 7.2500 NaN S

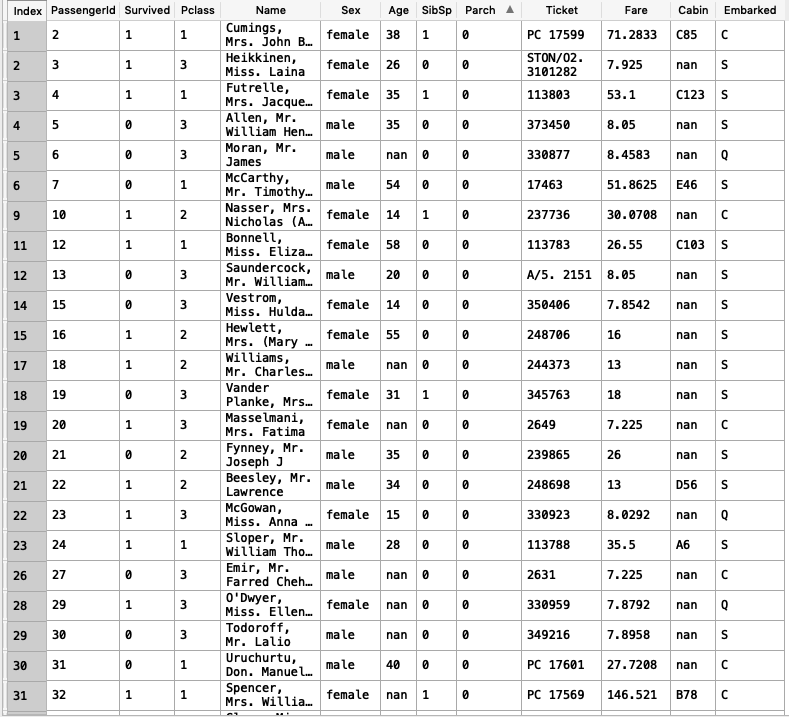
1 2 1 1 ... 71.2833 C85 C

2 3 1 3 ... 7.9250 NaN S

3 4 1 1 ... 53.1000 C123 S

4 5 0 3 ... 8.0500 NaN S

"""



survived = titanic\_data[titanic\_data.Survived ==0]

survived['Sex'].hist()

Obraz zawierający zrzut ekranu

Opis wygenerowany automatycznie

#Prawie pięć razy więcej umarło mężczyzn niż kobiet

survived['Age'].hist()  
  
Obraz zawierający tekst, krzyżówka

Opis wygenerowany automatycznie

#Najwięcej osób zmarło w przedziale wiekowym od 15 do 30 lat

survived['Pclass'].hist()  
  
Obraz zawierający shoji

Opis wygenerowany automatycznie

#Ponad 3,5 razy więcej osób zmarło z niższej klasy społecznej niż z średniej lub wyższej