* Scrie un program care cere două numere de la utilizator și încearcă să le împartă. Prinde orice excepție care ar putea apărea (ex: împărțire la zero sau introducere nevalidă).
* Definește o listă de 5 elemente și cere un index de la utilizator. Încearcă să afișezi elementul de la acel index, dar tratează excepția dacă indexul este în afara listei.
* \* Cere două numere și o operație (+, -, \*, /). Încearcă să efectuezi calculul și să tratezi erorile: împărțire la 0, operare invalidă, date nevalide.
* Scrie o funcție safe\_divide(a, b) care returnează rezultatul împărțirii, dar:
* dacă b == 0, returnează "Impartire la zero interzisa!"
* dacă a sau b nu sunt numere, returnează "Valori invalide"

Folosește try, except, else.

* Cere în mod repetat numere de la utilizator și adună-le până când se introduce „stop”. Folosește try-except pentru a prinde dacă nu se introduce un număr valid.

Incercati sa faceti rezolvarile in cadrul unor functii, unde e cazul, si organizati rularea codului in acel ‘ if \_\_name\_\_ == “\_\_main\_\_”: ‘. Obisnuiti-va sa va pastrati munca organizata in acest format.

EX 1

def numere\_2():  
 try:  
 numar1 = float(input("Introdu primul numar:"))  
  
 while True:  
 try:  
 numar2 = float(input("Introdu al doile numar:"))  
 if numar2 == 0:  
 print("Al doilea numar nu poate sa fie 0.Incercati din nou.")  
 else:  
 break  
 except ValueError:  
 print("Introduceti un numar valid")  
 rezultat = numar1 / numar2  
 print(f"Rezultat: {rezultat}")  
 except ZeroDivisionError:  
 print("Al doilea numar nu poate sa fie 0.")  
 except Exception as e:  
 print(f"A aparut o eroare neasteptata:{e}")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 numere\_2()

EX 2

def elemente\_5():  
 lista = ["mere", "banane", "mure", "pere", "struguri"]  
  
  
 try:  
 index = int(input("Introdu un index:"))  
 print(f"Elementul de la indexul {index} este: {lista[index]}")  
 except IndexError:  
 print("Indexul introdus este in afara limitelor listei.")  
 except ValueError:  
 print("Eroare: Introdu un numar intreg valid.")  
  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 elemente\_5()

EX 3

def operatie\_2\_nr():  
 try:  
 numar1 = float(input("Scrie primul numar: "))  
 numar2 = float(input("Scrie al doilea numar: "))  
 operatie = str(input("Scrie operatie: "))  
 if operatie == "+":  
 print(numar1 + numar2)  
 elif operatie == "-":  
 print(numar1 - numar2)  
 elif operatie == "\*":  
 print(numar1 \* numar2)  
 elif operatie == "/":  
 print(numar1 / numar2)  
 else:  
 print("Operatie invalida")  
 except ValueError:  
 print("Numar invalid")  
 except ZeroDivisionError:  
 print("Al doilea numar nu poate fi 0 pentru impartire")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 operatie\_2\_nr()

EX 4

def safe\_divide(a, b):  
 try:  
 rezultat = a / b  
 return rezultat  
 except ZeroDivisionError:  
 print("Impartire la zero interzisa!")  
 except TypeError:  
 print("Valori invalide!")  
 except Exception as e:  
 print(f"A aparut o eroare neasteptata:{e}")  
 return None  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print(safe\_divide(100, 0))  
 print(safe\_divide("100", "1"))  
 print(safe\_divide(100, 5))

EX 5

def adunare():  
 try:  
 sir\_de\_numere = 0  
 while True:  
 numar\_nou = str(input("Intrudu un numar intreg sau introdu 'stop' pentru a opri aplicatia: "))  
 if numar\_nou.lower() == "stop":  
 break  
 sir\_de\_numere += int(numar\_nou)  
 print(sir\_de\_numere)  
 except ValueError:  
 print("Valoare invalida")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 adunare()