Tema 13.
Excepțiile în Python

Ce ne așteaptă?

1. Captarea excepțiilor
2. Blocul finally
3. Structuri try-except-finally imbricate
4. Excepții definite de utilizator

1. Care sunt tipurile de erori în Python?
2. Câte tipuri de fluxuri de execuție sunt în Python?
3. Ce reprezintă o excepție?
4. Cum se captează o excepție?
5. Care sunt particularitățile structurii try-except?
6. Care este rolul blocului finally?
7. Care sunt particularitățile structurii try-except-finally imbricate
8. Care este rolul blocului else la captarea excepțiilor?
9. Care sunt combinațiile posibile ale blocurile de captare?
10. Cum se creează și se generează excepțiile definite de utilizator?

1. Captarea excepțiilor

Tipuri de erori

- **Erori de sintaxă – erori apărute în sintaxa codului**

```
x=123
if x==123
    print("Hello")
```

- **Erori de execuție (excepții) – erori apărute la executarea programului fiind cauzate de**
  - intrările utilizatorului
  - logica programului
  - probleme cu memoria

```
print(10/0)

print(10/”doi”)

x=int(input(“Introduceti un numar:"))
print(x)
```

Fluxul de execuție a programului

- Fluxul normal de execuție - se execută cu succes toate declarațiile

```
print('Unu')
print('Doi')
print('Trei')
print('Patru')
print('Cinci')
```

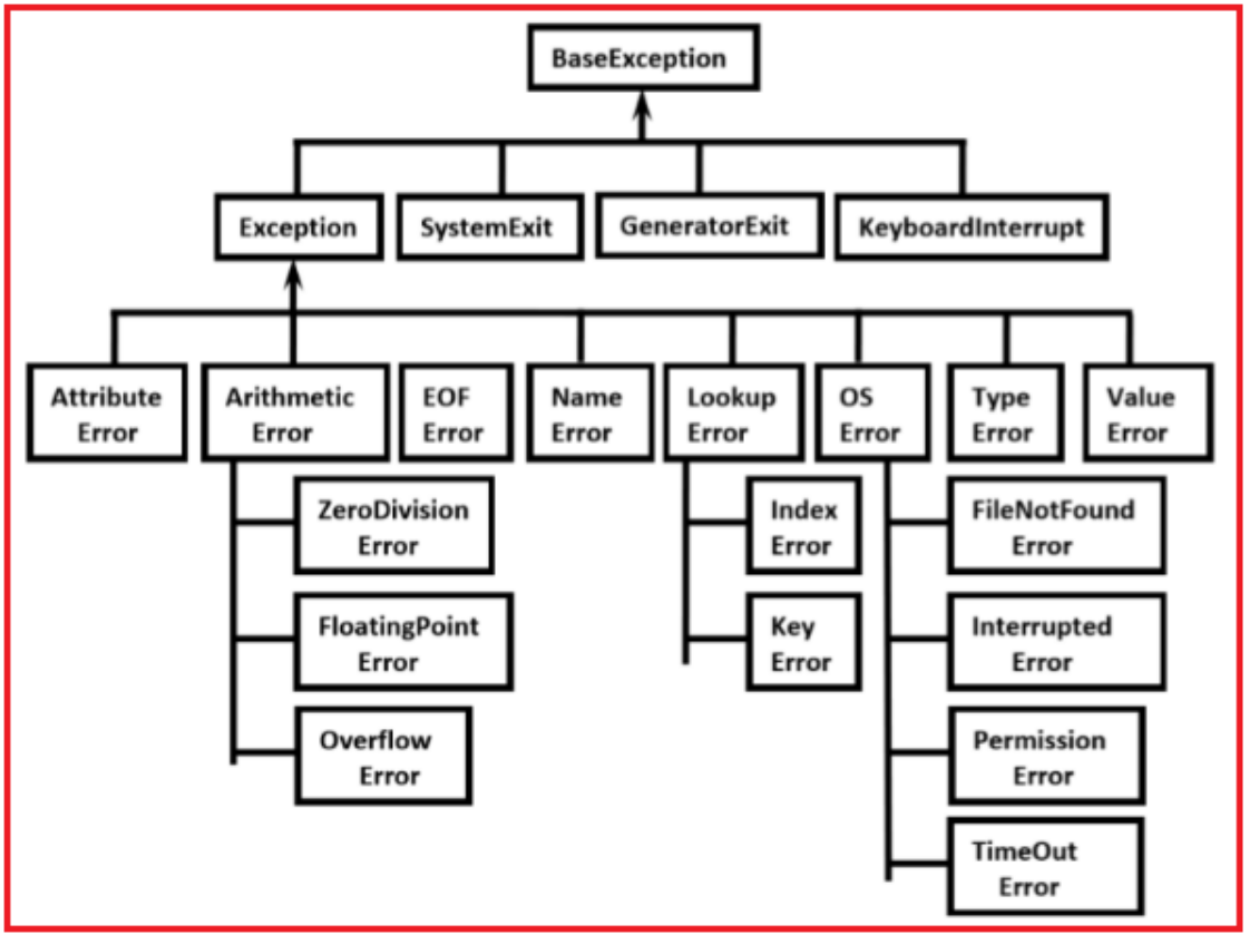
- Fluxul anormal de execuție – apare o eroare și se întrerupe execuția programului

```
print('Unu')
print('Doi')
print(10/0)
print('Patru')
print('Cinci')
```

Noțiune de excepție

- Excepție – orice eveniment nedorit și neașteptat ce întrerupe fluxul normal de execuție
- Captarea excepțiilor – procesul de corectare a erorilor și de evitare a fluxului anormal
- Fiecărui tip de eroare îi corespunde o clasă, iar excepția este un obiect al clasei respective
- La apariția unei erori, Python (PVM) creează obiectul corespunzător și caută codul de captare a acestuia
- Dacă codul de captare lipsește, se afișează informația erorii și se întrerupe execuția

Ierarhia claselor erorilor



Captarea excepțiilor

- Captarea excepțiilor se realizează cu structura *try-except*
- Sintaxa try-except :

```
try:
    codul in care poate apărea exceptie
except NumeExceptie:
    codul care se va executa cand va aparea exceptia
```

- try și except sunt cuvinte cheie în Python
- Exemplu de utilizare:

```
num1=10
num2=0
try:
    num3=num1/num2
    print(num3)
except ZeroDivisionError:
    print("Divizarea la 0 nu există")
print('Finisare executie')
```

Particularități ale structurii try-except

- Odată apărută excepția în blocul try, se trece la executarea blocul except, restul codului rămas în blocul try nu se mai execută
- Blocul except se execută doar dacă apare eroarea cărei denumire este specificată
- Dacă în declarația except nu se trece nici o denumire a excepției, atunci blocul acesteia se va executa la apariție oricărei excepții
- Dacă în blocul except apare o excepție, atunci aceasta nu va fi captată și se ajunce la un flux anormal de execuție

```
num1=10
try:
    print("Inceputul blocului try")
    num2=num1/"zero"
    print(num2)
    print("Sfarsitul blocului try")
except:
    print("A aparut o exceptie")
print('Finisare executie')
```

Informații despre excepție

- Pentru captarea tuturor excepțiilor se poate trece în sintaxa except denumire clasei superioare a excepțiilor - Exception
- În sintaxa except se poate utiliza aliasing-ul pentru a putea afla informații despre excepție
- În blocul except se poate cere afisarea informațiilor despre eroare prin printarea aliasing-ului excepției

```
num1=10
try:
    print("Inceputul blocului try")
    num2=num1/"zero"
    print(num2)
    print("Sfarsitul blocului try")
except Exception as exc:
    print(exc)
print('Finisare executie')
```

Structura try-except cu blocuri except multiple

- **Sintaxa try-except cu blocuri except multiple:**

```
try:
    codul in care poate apărea exceptie
except NumeExceptie1:
    codul care se va executa cand va aparea exceptia NumeExceptie1
except NumeExceptie2:
    codul care se va executa cand va aparea exceptia NumeExceptie2
...
```

- **Exemplu de utilizarea**

```
try:
    num1=int(input("Introduceti primul numar: "))
    num2=int(input("Introduceti al doilea numar: "))
    print(num1/num2)
except ZeroDivisionError:
    print("Divizare la zero nu exista")
except ValueError:
    print("Nu ati introdus valoarea numerica")
```

Ordinea blocuri except multiple

- În cazul structurii try-except cu blocuri except multiple se vor scrie mai întâi blocurile mai specifice și apoi a celor mai generale (conform arborelui ierarhiei):

- Dacă excepțiile sunt independente ordinea blocuri except nu are importanță dar se va executa blocul a cărei excepții apare prima:

- Exemplu de utilizarea

```
try:
    num1=int(input("Introduceti primul numar: "))
    num2=int(input("Introduceti al doilea numar: "))
    print(num1/num2)
    print(num1[1])
except ZeroDivisionError:
    print("Divizare la zero nu exista")
except ValueError:
    print("Nu ati introdus valoarea numerica")
except Exception as exc:
    print("A aparut o alta exceptie: ", exc)
```

Structura try-except cu un bloc except cu exceptii multiple

- Sintaxa try-except cu un bloc except cu exceptii multiple:

```
try:
    codul in care poate apărea exceptie
except (NumeExceptie1, NumeExceptie2 ,...):
    codul care se va executa cand va aparea o excepție din cele enumerate
sau
except (NumeExceptie1, NumeExceptie2 ,...) as exc:
    codul care se va executa cand va aparea o excepție din cele enumerate
```

- Exemplu de utilizarea

```
try:
    num1=int(input("Introduceti primul numar: "))
    num2=int(input("Introduceti al doilea numar: "))
    print(num1/num2)
except (ZeroDivisionError,ValueError) as exc:
    print("A aparut o exceptie: ", exc)
```

2. Blocul finally

Necesitatea blocului finally

- Blocul finally permite definirea unui cod ce se va executa indiferent de execuția blocurile try sau except
- Sintaxa try-except cu un bloc except cu exceptii multiple:

```
try:
    codul in care poate apărea exceptie
except NumeExceptie:
    codul care se va executa cand va aparea exceptia
finally:
    codul care se va executa întotdeauna
```

- Exemplu de utilizarea

```
try:
    num1=int(input("Introduceti primul numar: "))
    num2=int(input("Introduceti al doilea numar: "))
    print(num1/num2)
except ZeroDivisionError:
    print("Divizare la zero nu exista")
finally:
    print("Primul numar introdus este: ", num1)
```

Excepție în blocul finally

- Dacă apare o excepți în blocul finally acesta nu va fi captată

- Exemplu de utilizare

```
try:
    num1=int(input("Introduceti primul numar: "))
    num2=int(input("Introduceti al doilea numar: "))
    print(num1/num2)
except ZeroDivisionError:
    print("Divizare la zero nu exista")
finally:
    print(num1[1])
    print("Primul numar introdus este: ", num1)
```

3. Structuri try-except-finally imbricate

Particularitățile structurii try-except-finally imbricate

- Structură try-except-finally imbricat – bloc try-except-finally situat în interiorul unuia dintre oricare blocurile try, except, finally
- Structura imbricată se mai numește bloc intern, iar cea in care se află – bloc extern
- Excepțiile apărute în blocul intern sunt captate de blocul inter și cele apărute în blocul extern, de blocul exter

```
try:
    num1=int(input("Introduceti primul numar: "))
    num2=int(input("Introduceti al doilea numar: "))
    print(num1/num2)
    try:
        print(num1[1])
    except TypeError:
        print("Tipul int nu utilizeaza indecsi")
    finally:
        print("Blocul intern s-a executat")
except ZeroDivisionError:
    print("Divizare la zero nu exista")
finally:
    print("Blocul extern s-a executat")
```

Blocul else

- Blocul else se va executa doar atunci când nu vor apărea excepții
- Blocul else se situează după blocul except dar înaintea blocului finally

```
try:
    num1=int(input("Introduceti primul numar: "))
    num2=int(input("Introduceti al doilea numar: "))
    print(num1/num2)
except ZeroDivisionError:
    print("Divizare la zero nu exista")
else:
    print("Al doilea numar este diferit de zero")
finally:
    print("Primul numar introdus este: ", num1)
```

Combinații posibile cu structura try-except-else-finally (1)

- **Blocuri singulare – sintaxă nevalidă**
  - doar blocul try:

```
try:
    num1=int(input("Introduceti primul numar: "))
    num2=int(input("Introduceti al doilea numar: "))
    print(num1/num2)
```

  - doar blocul except:

```
except ZeroDivisionError:
    print("Divizare la zero nu exista")
```

  - doar blocul else:

```
else:
    print("Al doilea numar este diferit de zero")
```

  - doar blocul finally:

```
finally:
    print("Primul numar introdus este: ", num1)
```

Combinații posibile cu structura try-except-else-finally (2)

- **Blocul try cu bloc except și bloc finally – sintaxă validă**

– blocul try doar cu except

```
try:
    num1=int(input("Introduceti primul numar: "))
    num2=int(input("Introduceti al doilea numar: "))
    print(num1/num2)
except ZeroDivisionError:
    print("Divizare la zero nu exista")
```

– blocul try doar cu finally

```
try:
    num1=int(input("Introduceti primul numar: "))
    num2=int(input("Introduceti al doilea numar: "))
    print(num1/num2)
finally:
    print("Primul numar introdus este: ", num1)
```

Combinații posibile cu structura try-except-else-finally (3)

- **Blocul try cu bloc else fără bloc except – sintaxă nevalidă**

```
try:
    num1=int(input("Introduceti primul numar: "))
    num2=int(input("Introduceti al doilea numar: "))
    print(num1/num2)
else:
    print("Al doilea numar este diferit de zero")
```

- **Blocul try cu bloc except și bloc else – sintaxă validă**

```
try:
    num1=int(input("Introduceti primul numar: "))
    num2=int(input("Introduceti al doilea numar: "))
    print(num1/num2)
except ZeroDivisionError:
    print("Divizare la zero nu exista")
else:
    print("Al doilea numar este diferit de zero")
```

4. Excepții definite de utilizator

Crearea și generarea excepțiilor de utilizator

- **Excepții predefinite – excepții a căror clasă este incorporată în Python (ex ValueError, etc)**
- **Excepții definite de utilizator – excepții create de utilizator și generate în mod explicit în funcție de necesitate**
- **Sintaxa creării clasei excepției**

```
class  NumeExceptie(Exception):
    def__init__(self, argument)
        self.argument = argument
```

- **Sintaxa generării (aruncării) excepției**

```
raise  NumeExceptie("mesaj"):
```

Exemplu de excepții definite de utilizator

- Excepția NegativeError va apărea atunci când se va introduce un număr negativ

```
class NegativeError(Exception):
    def __init__(self, data):
        self.data = data

try:
    x = int(input("Introdu un numar pozitiv: "))
    if x < 0:
        raise NegativeError(x)
    print(f"Ati introdus numarul {x}")

except NegativeError as e:
    print(f"Ati introdus {e}. Introduceti, va rog, un numar pozitiv")
```