Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Python
Kivétel kezelés

Dr. Schuster György

2017. november 13.

Kivétel kezelés

Miért jó a kivétel kezelés?

Kivétel kezelés

Miért jó a kivétel kezelés?

Egy keletkezett hiba nem állítja meg a programot.

Kivétel kezelés

Miért jó a kivétel kezelés?

Egy keletkezett hiba nem állítja meg a programot.

Ha hiba keletkezett lehetőségünk van korrigálni.

Kivétel kezelés

Miért jó a kivétel kezelés?

Egy keletkezett hiba nem állítja meg a programot.

Ha hiba keletkezett lehetőségünk van korrigálni.

Több féle hibát is tudunk szisztematikusan kezelni.

Kivétel kezelés

Miért jó a kivétel kezelés?

Egy keletkezett hiba nem állítja meg a programot.

Ha hiba keletkezett lehetőségünk van korrigálni.

Több féle hibát is tudunk szisztematikusan kezelni.

Lehetőségünk van saját hiba kezelésre is.

Kivétel kezelés

A legegyszerűbb hibakezelés a **`try-except`** módszer.

Kivétel kezelés

A legegyszerűbb hibakezelés a **`try-except`** módszer.

Vegyük a következő példát!

Kivétel kezelés

A legegyszerűbb hibakezelés a **try-except** módszer.

Vegyük a következő példát!

```
#!/usr/bin/python3
x=1
for y in [-2,-1,0,1,2]:
    z=x/y
    print(y)
```

Képernyő

Kivétel kezelés

A legegyszerűbb hibakezelés a **try-except** módszer.

Vegyük a következő példát!

```
#!/usr/bin/python3
x=1
for y in [-2,-1,0,1,2]:
    z=x/y
    print(y)
```

Képernyő

```
-2
-1
```

A program fut, a ciklustörzs gond nélkül végrahajtásra kerül.

Kivétel kezelés

A legegyszerűbb hibakezelés a **try-except** módszer.

Vegyük a következő példát!

```
#!/usr/bin/python3
x=1
for y in [-2,-1,0,1,2]:
    z=x/y
    print(y)
```

Képernyő

```
-2
-1
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
ZeroDivisionError:  division by zero
```

A program fut, a ciklustörzs gond nélkül végrahajtásra kerül.
A program futása megaszakad és a program kilép, mert nullával osztottunk.

Kivétel kezelés

A legegyszerűbb hibakezelés a **try-except** módszer.

Vegyük a következő példát!

```
#!/usr/bin/python3
x=1
for y in [-2,-1,0,1,2]:
    z=x/y
    print(y)
```

Képernyő

```
-2
-1
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
ZeroDivisionError:  division by zero
```

A program fut, a ciklustörzs gond nélkül végrahajtásra kerül.
A program futása megaszakad és a program kilép, mert nullával osztottunk.
Jó lenne, ha nem lépne ki a program.

Kivétel kezelés

A legegyszerűbb hibakezelés a **try-except** módszer.

Vegyük a következő példát!

```
#!/usr/bin/python3
x=1
for y in [-2,-1,0,1,2]:
    z=x/y
    print(y)
```

Képernyő

```
-2
-1
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
ZeroDivisionError:  division by zero
```

A program fut, a ciklustörzs gond nélkül végrahajtásra kerül.
A program futása megaszakad és a program kilép, mert nullával osztottunk.
Jó lenne, ha nem lépne ki a program.
Akkor kapjuk el a hibát!

Kivétel kezelés

A legegyszerűbb hibakezelés a **`try-except`** módszer.

Az átalakított program:

Kivétel kezelés

A legegyszerűbb hibakezelés a **try-except** módszer.

Az átalakított program:

```
#!/usr/bin/python3
x=1
for y in [-2,-1,0,1,2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    print(y)
```

Kivétel kezelés

A legegyszerűbb hibakezelés a **try-except** módszer.

Az átalakított program:

```
#!/usr/bin/python3
x=1
for y in [-2,-1,0,1,2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    print(y)
```

A **try**-blokk, ide kerül a kritikus programrészlet.

Kivétel kezelés

A legegyszerűbb hibakezelés a **try-except** módszer.

Az átalakított program:

```
#!/usr/bin/python3
x=1
for y in [-2,-1,0,1,2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    print(y)
```

A **try**-blokk, ide kerül a kritikus programrészlet.
A kérdéses **except**-blokk, ami hiba esetén kerül végrehajtásra.
A hibát mi kezeljük!

Kivétel kezelés

A legegyszerűbb hibakezelés a **try-except** módszer.

Az átalakított program:

```
#!/usr/bin/python3
x=1
for y in [-2,-1,0,1,2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    print(y)
```

Képernyő

```
-2
-1
```

Nincs probléma.

Kivétel kezelés

A legegyszerűbb hibakezelés a **try-except** módszer.

Az átalakított program:

```
#!/usr/bin/python3
x=1
for y in [-2,-1,0,1,2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    print(y)
```

Képernyő

```
-2
-1
Auuu!
0
```

Nincs probléma.
Nullával osztás történt.

Kivétel kezelés

A legegyszerűbb hibakezelés a **try-except** módszer.

Az átalakított program:

```
#!/usr/bin/python3
x=1
for y in [-2,-1,0,1,2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    print(y)
```

Képernyő

```
-2
-1
Auuu!
0
1
2
```

Nincs probléma.
Nullával osztás történt.
A program megy tovább.

Kivétel kezelés

A legegyszerűbb hibakezelés a **try-except** módszer.

Az átalakított program:

```
#!/usr/bin/python3
x=1
for y in [-2,-1,0,1,2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    print(y)
```

Több hiba kezelése

Hogyan kezelünk több hibát?

Több hiba kezelése

Hogyan kezelünk több hibát?
Több `except`-blokkot használunk.

Több hiba kezelése

Hogyan kezelünk több hibát?
Több **except**-blokkot használunk.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    except TypeError:
        print("RTFM!")
    print(y)
```

Nullával való osztás hiba kezelése.

Több hiba kezelése

Hogyan kezelünk több hibát?
Több **except**-blokkot használunk.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    except TypeError:
        print("RTFM!")
    print(y)
```

Nullával való osztás hiba kezelése.

Rossz típus használat hiba kezelése,

Több hiba kezelése

Hogyan kezelünk több hibát?
Több **except**-blokkot használunk.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    except TypeError:
        print("RTFM!")
    print(y)
```

Képernyő

```
-1
```

Nincs hiba, a program fut.

Több hiba kezelése

Hogyan kezelünk több hibát?
Több **except**-blokkot használunk.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    except TypeError:
        print("RTFM!")
    print(y)
```

Képernyő

```
-1
Auuu!
0
```

Nincs hiba, a program fut.
Nullával való osztás hiba, elkaptuk.

Több hiba kezelése

Hogyan kezelünk több hibát?
Több **except**-blokkot használunk.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    except TypeError:
        print("RTFM!")
    print(y)
```

Képernyő

```
-1
Auuu!
0
1
```

Nincs hiba, a program fut.
Nullával való osztás hiba, elkaptuk.
Nincs hiba, a program fut.

Több hiba kezelése

Hogyan kezelünk több hibát?
Több **except**-blokkot használunk.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    except TypeError:
        print("RTFM!")
    print(y)
```

Képernyő

```
-1
Auuu!
0
1
RTFM!
Hello
```

Nincs hiba, a program fut.
Nullával való osztás hiba, elkaptuk.
Nincs hiba, a program fut.
Típus hiba, elkaptuk.

Több hiba kezelése

Hogyan kezelünk több hibát?
Több **except**-blokkot használunk.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    except TypeError:
        print("RTFM!")
    print(y)
```

Képernyő

```
-1
Auuu!
0
1
RTFM!
Hello
2
```

Nincs hiba, a program fut.
Nullával való osztás hiba, elkaptuk.
Nincs hiba, a program fut.
Típus hiba, elkaptuk.
Nincs hiba, a program fut.

Több hiba kezelése

Hogyan kezelünk több hibát?
Több **except**-blokkot használunk.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except ZeroDivisionError:
        print("Auuu!")
    except TypeError:
        print("RTFM!")
    print(y)
```

Honnan tudhatjuk a hiba típusát?

Honnan tudhatjuk a hiba típusát?

1. Kiolvassuk a dokumentációból (RTFM[1])

[1] Read The *Friendly* Manual

Honnan tudhatjuk a hiba típusát?

1. Kiolvassuk a dokumentációból (RTFM [1])
2. Elkövetünk egy keresett hibát és a hibaüzenetből egyszerűen kiolvassuk. Pl.:

[1]

Honnan tudhatjuk a hiba típusát?

1. Kiolvassuk a dokumentációból (RTFM [1])
2. Elkövetünk egy keresett hibát és a hibaüzenetből egyszerűen kiolvassuk. Pl.:

```
>>> x=1
>>> y=0
>>> z=x/y
```

Képernyő

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError:  division by zero
```

[1]

Honnan tudhatjuk a hiba típusát?

1. Kiolvassuk a dokumentációból (RTFM [1])
2. Elkövetünk egy keresett hibát és a hibaüzenetből egyszerűen kiolvassuk. Pl.:

```
>>> x=1
>>> y=0
>>> z=x/y
```

Képernyő

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError:  division by zero
```

[1]

Honnan tudhatjuk a hiba típusát?

1. Kiolvassuk a dokumentációból (RTFM [1])
2. Elkövetünk egy keresett hibát és a hibaüzenetből egyszerűen kiolvassuk. Pl.:

```
>>> x=1
>>> y=0
>>> z=x/y
```

Képernyő

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError:  division by zero
```

Ezt másoljuk a megfelelő **except** mögé!

[1]

Lehetőség van bármilyen hiba elkapására is.

Lehetőség van bármilyen hiba elkapására is.

Példa:

Lehetőség van bármilyen hiba elkapására is.

Példa:

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    print(y)
```

Üres **except**-et adunk meg. Ekkor a program minden hibát elkap, de a hibát nekünk kell azonosítani.

Lehetőség van bármilyen hiba elkapására is.

Példa:

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    print(y)
```

Képernyő

```
-1
```

Lehetőség van bármilyen hiba elkapására is.

Példa:

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    print(y)
```

Képernyő

```
-1
Error!
0
```

Lehetőség van bármilyen hiba elkapására is.

Példa:

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    print(y)
```

Képernyő

```
-1
Error!
0
1
```

Lehetőség van bármilyen hiba elkapására is.

Példa:

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    print(y)
```

Képernyő

```
-1
Error!
0
1
Error!
Hello!
```

Lehetőség van bármilyen hiba elkapására is.

Példa:

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    print(y)
```

Képernyő

```
-1
Error!
0
1
Error!
Hello!
2
```

Lehetőség van bármilyen hiba elkapására is.

Példa:

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    print(y)
```

Üres **except**-et adunk meg. Ekkor a program minden hibát elkap, de a hibát nekünk kell azonosítani.

Kivétel kezelés

Az előző példákban kiírtuk az `y` változó értékeit, akár volt hiba, akár nem.

Kivétel kezelés

Az előző példákban kiírtuk az `y` változó értékeit, akár volt hiba, akár nem.

De szeretnénk, hogy hiba esetén ne legyen kiírva az érték.

Kivétel kezelés

Az előző példákban kiírtuk az `y` változó értékeit, akár volt hiba, akár nem.

De szeretnénk, hogy hiba esetén ne legyen kiírva az érték.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    else:
        print(y)
```

Ha hiba történik, elkapjuk
Ha nem történik hiba, akkor az `else` lefut és kiírjuk az értéket.

Kivétel kezelés

Az előző példákban kiírtuk az `y` változó értékeit, akár volt hiba, akár nem.

De szeretnénk, hogy hiba esetén ne legyen kiírva az érték.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    else:
        print(y)
```

Képernyő

```
-1
```

Kivétel kezelés

Az előző példákban kiírtuk az `y` változó értékeit, akár volt hiba, akár nem.

De szeretnénk, hogy hiba esetén ne legyen kiírva az érték.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    else:
        print(y)
```

Képernyő

```
-1
Error!
```

Kivétel kezelés

Az előző példákban kiírtuk az `y` változó értékeit, akár volt hiba, akár nem.

De szeretnénk, hogy hiba esetén ne legyen kiírva az érték.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    else:
        print(y)
```

Képernyő

```
-1
Error!
1
```

Kivétel kezelés

Az előző példákban kiírtuk az `y` változó értékeit, akár volt hiba, akár nem.

De szeretnénk, hogy hiba esetén ne legyen kiírva az érték.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    else:
        print(y)
```

Képernyő

```
-1
Error!
1
Error!
```

Kivétel kezelés

Az előző példákban kiírtuk az `y` változó értékeit, akár volt hiba, akár nem.

De szeretnénk, hogy hiba esetén ne legyen kiírva az érték.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    else:
        print(y)
```

Képernyő

```
-1
Error!
1
Error!
2
```

Kivétel kezelés

Az előző példákban kiírtuk az `y` változó értékeit, akár volt hiba, akár nem.

De szeretnénk, hogy hiba esetén ne legyen kiírva az érték.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    else:
        print(y)
```

Ha hiba történik, elkapjuk
Ha nem történik hiba, akkor az `else` lefut és kiírjuk az értéket.

Kivétel kezelés

A **`finally`** utasítás, ha megadjuk mindenképpen a hibakezelő blokk végén lefut.

Kivétel kezelés

A **`finally`** utasítás, ha megadjuk mindenképpen a hibakezelő blokk végén lefut.

Példa:

Kivétel kezelés

A **finally** utasítás, ha megadjuk mindenképpen a hibakezelő blokk végén lefut.

Példa:

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    finally:
        print(y)
```

Kivétel kezelés

A **finally** utasítás, ha megadjuk mindenképpen a hibakezelő blokk végén lefut.

Példa:

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    finally:
        print(y)
```

Képernyő

```
-1
Error!
0
1
Error!
Hello!
2
```

Kivétel kezelés

A **finally** utasítás, ha megadjuk mindenképpen a hibakezelő blokk végén lefut.

Példa:

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    finally:
        print(y)
```

Ennek is van értelme, de még nem látszik.

Kivétel kezelés

A **finally** utasítás, ha megadjuk mindenképpen a hibakezelő blokk végén lefut.

Példa:

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,'Hello!',2]:
    try:
        z=x/y
    except:
        print("Error!")
    finally:
        print(y)
```

Kivétel kezelés

Előfordulhat olyan eset, hogy a kivételt nem akarjuk teljesen kezelni, de még a kilépés előtt valamit el akarunk végezni.

Kivétel kezelés

Előfordulhat olyan eset, hogy a kivételt nem akarjuk teljesen kezelni, de még a kilépés előtt valamit el akarunk végezni.

Ekkor a **`raise`** utasítás a megoldás.

Kivétel kezelés

Előfordulhat olyan eset, hogy a kivételt nem akarjuk teljesen kezelni, de még a kilépés előtt valamit el akarunk végezni.

Ekkor a **raise** utasítás a megoldás.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,2]:
    try:
        z=x/y
    except:
        print("Error!")
        raise
    else:
        print(y)
```

Ezt mi kezeljük.

Átadjuk a rendszernek a vezérlést.

Kivétel kezelés

Előfordulhat olyan eset, hogy a kivételt nem akarjuk teljesen kezelni, de még a kilépés előtt valamit el akarunk végezni.

Ekkor a **raise** utasítás a megoldás.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,2]:
    try:
        z=x/y
    except:
        print("Error!")
        raise
    else:
        print(y)
```

Képernyő

```
-1
```

Kivétel kezelés

Előfordulhat olyan eset, hogy a kivételt nem akarjuk teljesen kezelni, de még a kilépés előtt valamit el akarunk végezni.

Ekkor a **raise** utasítás a megoldás.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,2]:
    try:
        z=x/y
    except:
        print("Error!")
        raise
    else:
        print(y)
```

Képernyő

```
-1
Error!
```

Kivétel kezelés

Előfordulhat olyan eset, hogy a kivételt nem akarjuk teljesen kezelni, de még a kilépés előtt valamit el akarunk végezni.

Ekkor a **raise** utasítás a megoldás.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,2]:
    try:
        z=x/y
    except:
        print("Error!")
        raise
    else:
        print(y)
```

Képernyő

```
-1
Error!
Traceback (most recent call last):
  File "./ex4.py", line 7, in <module>
    z=x/y
ZeroDivisionError:  division by zero
```

Kivétel kezelés

Előfordulhat olyan eset, hogy a kivételt nem akarjuk teljesen kezelni, de még a kilépés előtt valamit el akarunk végezni.

Ekkor a **raise** utasítás a megoldás.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,2]:
    try:
        z=x/y
    except:
        print("Error!")
        raise
    else:
        print(y)
```

Ezt mi kezeljük.

Átadjuk a rendszernek a vezérlést.

A program a **raise** után már nem fut.

Kivétel kezelés

Előfordulhat olyan eset, hogy a kivételt nem akarjuk teljesen kezelni, de még a kilépés előtt valamit

Ekkor a **raise** utasítá

A **raise** ennél sokkal többet tud, de ez most korai.

```
#!/usr/bin/python3
x=1
for y in [-1,0,1,2]:
    try:
        z=x/y
    except:
        print("Error!")
        raise
    else:
        print(y)
```

Ezt mi kezeljük.

Átadjuk a rendszernek a vezérlést.

A program a **raise** után már nem fut.

Kivétel kezelés

A kivételekről könnyen lehet információt szerezni a **`sys`** modul segítségével.

Kivétel kezelés

A kivételekről könnyen lehet információt szerezni a **`sys`** modul segítségével.
Mintapélda:

Kivétel kezelés

A kivételekről könnyen lehet információt szerezni a **sys** modul segítségével.
Mintapélda:

```
#!/usr/bin/python3
import sys
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except:
        e=sys.exc_info()[0]
        print(e)
    else:
        print(y)
```

A **sys** modul segítségével szerezhető információ a bekövetkezett hibáról. Ezután rajtunk múlik, hogyan kezeljük.

Kivétel kezelés

A kivételekről könnyen lehet információt szerezni a **sys** modul segítségével.
Mintapélda:

```
#!/usr/bin/python3
import sys
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except:
        e=sys.exc_info()[0]
        print(e)
    else:
        print(y)
```

Képernyő

```
-1
```

Kivétel kezelés

A kivételekről könnyen lehet információt szerezni a **sys** modul segítségével.
Mintapélda:

```
#!/usr/bin/python3
import sys
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except:
        e=sys.exc_info()[0]
        print(e)
    else:
        print(y)
```

Képernyő

```
-1
<class 'ZeroDivisionError'>
```

Kivétel kezelés

A kivételekről könnyen lehet információt szerezni a **sys** modul segítségével.
Mintapélda:

```
#!/usr/bin/python3
import sys
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except:
        e=sys.exc_info()[0]
        print(e)
    else:
        print(y)
```

Képernyő

```
-1
<class 'ZeroDivisionError'>
1
```

Kivétel kezelés

A kivételekről könnyen lehet információt szerezni a **sys** modul segítségével.
Mintapélda:

```
#!/usr/bin/python3
import sys
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except:
        e=sys.exc_info()[0]
        print(e)
    else:
        print(y)
```

Képernyő

```
-1
<class 'ZeroDivisionError'>
1
<class 'TypeError'>
```

Kivétel kezelés

A kivételekről könnyen lehet információt szerezni a **sys** modul segítségével.
Mintapélda:

```
#!/usr/bin/python3
import sys
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except:
        e=sys.exc_info()[0]
        print(e)
    else:
        print(y)
```

Képernyő

```
-1
<class 'ZeroDivisionError'>
1
<class 'TypeError'>
2
```

Kivétel kezelés

A kivételekről könnyen lehet információt szerezni a **sys** modul segítségével.
Mintapélda:

```
#!/usr/bin/python3
import sys
x=1
for y in [-1,0,1,'Hello',2]:
    try:
        z=x/y
    except:
        e=sys.exc_info()[0]
        print(e)
    else:
        print(y)
```

A **sys** modul segítségével szerezhető információ a bekövetkezett hibáról. Ezután rajtunk múlik, hogyan kezeljük.

Kivétel kezelés

A **`sys.exc_info()`** függvény egy három elemű kupacot ad vissza. Ennek elemei:

Kivétel kezelés

A **`sys.exc_info()`** függvény egy három elemű kupacot ad vissza. Ennek elemei:

0. a hiba megnevezése,

Kivétel kezelés

A **`sys.exc_info()`** függvény egy három elemű kupacot ad vissza. Ennek elemei:

0. a hiba megnevezése,
1. egy rövid magyarázatot tartalmaz a hibával kapcsolatban,

Kivétel kezelés

A **`sys.exc_info()`** függvény egy három elemű kupacot ad vissza. Ennek elemei:

0. a hiba megnevezése,
1. egy rövid magyarázatot tartalmaz a hibával kapcsolatban,
2. a hibakereséshez egy úgynevezett ”traceback” információt közöl.

Kivétel kezelés

A **`sys.exc_info()`** függvény egy három elemű kupacot ad vissza. Ennek elemei:

0. a hiba m
1. egy rövid
2. a hibakere közöl.

Most abbahagyjuk!

Kivétel kezelés

A **`sys.exc_info()`** függvény egy három elemű kupacot ad vissza. Ennek elemei:

0. a hiba m
1. egy rövid
2. a hibakere közöl.

Most abbahagyjuk!
Befejezni nem is lehet.