# Exceptions

# Exceptions

When Python detects an error, it *raises* an *exception* (also called “*throws an* exception”)

If the exception is not handled in the program, the program crashes and the error is displayed in a Traceback message

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

To prevent program from crashing when an error occurs, include a *handler* for the error

# Types of Error Exceptions

- **ValueError**: invalid value for function
- **IOError**: failed to open, read, or write file
- **TypeError**: invalid operation for variable
- **IndexError**: index out of range for list
- **KeyError**: key not found in dictionary
- **ZeroDivisionError**: attempt to divide by 0
- …

# Writing Exception Handlers

- Use **try** and **except** statements together

**try:**

    <code that could raise error exception>

**except** <type of error>:

    <code to handle error>

# ZeroDivisionError

```
num1 = int(input('Enter a number'))
num2 = int(input('Enter another number'))

result =num1/num2
print(num1, 'divided by', num2, 'is', result)
```

Try dividing by zero - PAUSE VIDEO HERE.

```python
num1 = int(input('Enter a number'))
num2 = int(input('Enter another number'))

result =num1/num2
print(num1, 'divided by', num2, 'is', result)

```

```
beste@Beste-Ubu: ~/cs110example/in_class/week8
beste@Beste-Ubu:~/cs110example/in_class/week8$ python3 zero.py
Enter a number2
Enter another number0
Traceback (most recent call last):
  File "zero.py", line 4, in <module>
    result =num1/num2
ZeroDivisionError: division by zero
beste@Beste-Ubu:~/cs110example/in_class/week8$
```

# ZeroDivisionError handle exception

```
num1 = int(input('Enter a number'))
num2 = int(input('Enter another number'))
try:
    result =num1/num2
    print(num1, 'divided by', num2, 'is', result)
except ZeroDivisionError:
    print('Cannot divide by zero')
```

Try dividing by zero - PAUSE VIDEO HERE.

```python
num1 = int(input('Enter a number'))
num2 = int(input('Enter another number'))

# result =num1/num2
# print(num1, 'divided by', num2, 'is', result)

try:
    result =num1/num2
    print(num1, 'divided by', num2, 'is', result)
except ZeroDivisionError:
    print('you cannot divide by zero')

```

```
beste@Beste-Ubu: ~/cs110example/in_class/week8
beste@Beste-Ubu:~/cs110example/in_class/week8$ python3 zero.py
Enter a number2
Enter another number0
you cannot divide by zero
beste@Beste-Ubu:~/cs110example/in_class/week8$
```

# ValueError

```
hours_worked = int(input('Enter the number of hours worked'))
#Enter a non-integer to break it
print(hours_worked)
```

Try code - PAUSE VIDEO HERE.

# ValueError handle exception

```
try:
    hours_worked = int(input('Enter the number of hours worked: '))
    print(hours_worked)
except ValueError:
    print('Input must be an integer')
```

Try code - PAUSE VIDEO HERE.

# Writing Exception Handlers

**try** can have multiple **except** statements

```
try:
    <code that could raise error exception>
except <error type 1>:
    <code to handle error type 1>
except <error type 2>:
    <code to handle error type 2>
```

# Writing Exception Handlers

- If you are lazy, can write one generic **except** statement

```
try:
    <code that could raise error exception>
except:
    <code to handle any type of error>
    print(“An error occurred”)
```

# When Will a Program Crash?

- When an error exception occurs in code that isn't in a **try** block

or

- When an error exception occurs in a **try** block, but no exception handler is provided for it

# *Else* Clause

- Place after all **except** blocks
- Executed after **try** block if *no* exceptions
  - Not executed if any exceptions occur

**try:**

<code that could raise error exception>

**except** <type of error>:

<code to handle error>

**else**:

<code that executes if no exceptions match>

# *Finally* Clause

- Place after all **except** blocks
- Executed after **try** block & any exceptions
- Useful for clean-up, e.g. closing files

```
try:
   <code that could raise error exception>
except <type of error>:
   <code to handle error>
finally:
   <code that executes after try block & except block>
```

# Example

```
try:
    bankBalance = int(input("Enter the starting bank balance: "))
except ValueError:
    print("Bank Balance must be a numerical value!")
else:
    print("This is the else block!")
finally:
    print("This is the finally block!")
```

What gets printed if user enters 20?
What gets printed if user enters ‘hello’?

Try code - PAUSE VIDEO HERE.

# Validating User Inputs (general)

```
again = True
while again:
    n = int(input("Please enter integer between 1 and 5: "))
    if n > 0 and n < 6:
        again = False
    else:
        print("Oops!  That isn't between 1 and 5.")
```

# Using Exceptions for Input Validation

- Validating input using exceptions is very similar
- Put exception-handler (**try-except** statement) in **while** loop
- Keep looping until user enters valid value

# Using Exceptions for Input Validation - no While loop

```
try:
    userChoice = int(input("Please enter an integer: "))

except ValueError:
    print("Oops!  That isn't an integer.")
```

Try code - PAUSE VIDEO HERE.

# Using Exceptions for Input Validation with While loop

```
again = True
while again:
    try:
        userChoice = int(input("Please enter an integer: "))
        again = False    # If we get here, input is valid
    except ValueError:
        print("Oops!  That isn't an integer.")
```

Try code - PAUSE VIDEO HERE.