## Exception Handling in Python

Exception handling in Python is very similar to Java. The code, which harbours the risk of an exception, is embedded in a try block. While in Java exceptions are caught by catch clauses, in Python we have statements introduced by an "except" keyword. It's possible to create "custom-made" exceptions: With the raise statement it's possible to force a specified exception to occur.

Let's look at a simple example. Assuming we want to ask the user to enter an integer number. If we use a input(), the input will be a string, which we have to cast into an integer. If the input isn't a valid integer, we will generate (raise) a ValueError. We show this in the following interactive session:

n = **int**(**input**("Please enter a number: "))

With the aid of exception handling, we can write robust code for reading an integer from input:

**while** True:

**try**:

n = **input**("Please enter an integer: ")

n = **int**(n)

**break**

**except** **ValueError**:

**print**("No valid integer! Please try again ...")

**print**("Great, you successfully entered an integer!")

It's a loop, which breaks only if a valid integer has been given. The while loop is entered. The code within the try clause will be executed statement by statement. If no exception occurs during the execution, the execution will reach the break statement and the while loop will be left. If an exception occurs, i.e. in the casting of n, the rest of the try block will be skipped and the except clause will be executed. The raised error, in our case a ValueError, has to match one of the names after except. In our example only one, i.e. "ValueError:". After having printed the text of the print statement, the execution does another loop. It starts with a new input().

We could turn the code above into a function, which can be used to have a foolproof input.

**def** int\_input(prompt):

**while** True:

**try**:

age = **int**(**input**(prompt))

**return** age

**except** **ValueError** **as** e:

**print**("Not a proper integer! Try it again")

We use this with our dog age example from the chapter [Conditional Statements](https://python-course.eu/python-tutorial/conditional-statements.php).

**def** dog2human\_age(dog\_age):

human\_age = -1

**if** dog\_age < 0:

human\_age = -1

**elif** dog\_age == 0:

human\_age = 0

**elif** dog\_age == 1:

human\_age = 14

**elif** dog\_age == 2:

human\_age = 22

**else**:

human\_age = 22 + (dog\_age -2) \* 5

**return** human\_age

age = int\_input("Age of your dog? ")

**print**("Age of the dog: ", dog2human\_age(age))

#### **OUTPUT:**

Not a proper integer! Try it again

Not a proper integer! Try it again

Age of the dog: 37