Rex
Jones II

How To Handle Exceptions Using try-except

Python Video = https://youtu.be/G4l53igtbX8

Exception Handling (Try / Except)

In this tutorial session, let's talk about exception handling. The purpose of handling an exception is to locate an error, handle the error, then continue running our program. Up to this point, our program stops running when it encounters an error.

In the IDE, let's start with a variable called number = then prompt input from the user input('(100/Number) Enter A Number: '). We want a value = after dividing 100/ by a number. Next, let's print(value). Also, tell the user print('Thanks For Your Input'). One more thing, let's also take the number and convert the number into an integer by writing the int() function.

```
number = int(input('(100/Number) Enter A Number: '))
value = 100/number
print(value)
print('Thanks For Your Input')
```

When I run, Enter A Number: How about 10? The console returns 10.0 and Thanks For Your Input

```
(100/Number) Enter A Number: 10
10.0
Thanks For Your Input
```

This time, watch what happens when I enter Ten. We get a Traceback.

```
(100/Number) Enter A Number: Ten
Traceback (most recent call last):
  File "C:\Users\RexJo\PycharmProjects\pythonProject\try_
    number = int(input('(100/Number) Enter A Number: '))
ValueError: invalid literal for int() with base 10: 'Ten'
```

A Traceback includes information about the exception. The exception is a ValueError: I entered Ten which is a String but the program expects an integer. That's why the console shows invalid literal for int(). Notice 2 things. #1 Both print statements did not show up and #2 the exit code shows 1.

```
Process finished with exit code 1
```

1 means there is a problem. We do not see the print statements because the program stopped running after the input() function Enter A Number. In the program, the print statements on lines 4 and 5 should print the value then print Thanks For Your Input. However, the program returned an exception because we did not handle the exception.

We handle exceptions in Python by using a try – except block. A try – except block has 2 concepts. First, it ask Python to do something then it tells Python to do something. Ask Then Tell. First, it asks Python to try and execute our code then tell Python to raise an exception if there is a problem with our code. For example, we want to try and execute the 1st 3 lines: Get input from the user, Divide 100 by a Number, then print the value. To perform the 1st concept, we write try: and a colon before the 3 lines then indent

3 the lines. For the 2nd concept, we use keyword except: and don't forget the colon after the 3 lines. Now, we want to perform a command. Let's just add a print() statement that says "Invalid Number:"

```
try:
  number = int(input('(100/Number) Enter A Number: '))
  value = 100/number
  print(value)
except:
  print('Invalid Number')
print('Thanks For Your Input')
```

Notice, there are 3 print statements. 2 of the print statements are indented and 1 of the print statements is not indented. For the 1st indented print statement is part of the try clause. The 2nd indented print statement is part of the except clause. The 3rd print statement is not indented and it's not part of no clause but it will execute each time. It shows how execution continues running after coming across an exception for an Invalid Number and it shows when not coming across an Invalid Number.

Try-except blocks are good for recovering from an exception that we anticipate to happen. If there is not an error, then Python skips the except clause. So when I run and enter 10 again. We see 10.0 and Thanks For Your Input! But when I run and enter the same string Ten.

```
(100/Number) Enter A Number: Ten
Invalid Number
Thanks For Your Input
```

The console let's us know Invalid Number. Also, we see the print statement Thanks For Your Input although there was an exception. We cannot divide a number like 100 by Ten. Exit code is 0.

```
Process finished with exit code 0
```

Do you see the line under except? That's too broad so the message says "Too broad exception clause". Python prefers a detailed exception handler like the ValueError and the line goes away when I enter ValueError. Now after the exception, we see the lines goes away but also, make the print statement more detail by adding "Invalid Number: Enter A Numerical Value". Now when I run, enter a string like One. The console shows Invalid Number: Enter A Numerical Value.

```
(100/Number) Enter A Number: One
Invalid Number: Enter A Numerical Value
Thanks For Your Input
```

The message provides more detail and let's the user know what to do. What if by mistake, a user wants to enter 20 but forget the 2 so they only have 0 as their number. We see a different Traceback. ZeroDivisionError is the exception. It shows up because a number cannot be divided by 0.

```
(100/Number) Enter A Number: 0
Traceback (most recent call last):
  File "C:\Users\RexJo\PycharmProjects\pythonProject\try_
    value = 100/number
ZeroDivisionError: division by zero
```

The previous exception for ValueError did not match ZeroDivisionError so the program stopped running. If I hover except we see "The except clause(s) specify one or more exception handlers". Therefore, we can add one more except clause for ZeroDivisionError. In fact, we can add as many exceptions as we need to. So before the last print statement, we write except ZeroDivisionError. It's optional but we can also add a variable after the exception name: as ex. It can also be exc. So it doesn't matter what we name this exc. Don't forget the colon. Let's print(exc). exc will return print division by zero if there is an error. Let's also add another print() statement so it's more detail print("Invalid Number: Number Cannot Be 0"). If there is not an error with the code inside the try block then Python skips over each except block. Now, if there is an error then Python look for an except block that matches the error then execute that code.

```
try:
  number = int(input('(100/Number) Enter A Number: '))
  value = 100/number
  print(value)
except ValueError:
  print('Invalid Number: Enter A Numerical Value')
except ZeroDivisionError as exc:
  print(exc)
  print('Invalid Number: Number Cannot Be 0')
print('Thanks For Your Input')
```

Let's Run and Number is 0. The console shows division by zero which comes from the variable exc – Invalid Number: Number Cannot Be 0 is our customized message.

```
(100/Number) Enter A Number: 0
division by zero
Invalid Number: Number Cannot Be 0
Thanks For Your Input
```

Enter another number like Two. We see a different message Invalid Number: Enter A Numerical Value.

```
(100/Number) Enter A Number: Two
Invalid Number: Enter A Numerical Value
Thanks For Your Input
```

We see how the try-except blocks handle an exception. If we enter a good number like 5 then we see 20.0 and print statement Thanks For Your Input which prints every time when handling an exception.

```
(100/Number) Enter A Number: 5
20.0
Thanks For Your Input
```

Contact Info

- ✔ Email Rex.Jones@Test4Success.org
- ✔ YouTube https://www.youtube.com/c/RexJonesII/videos
- ✔ Facebook https://facebook.com/JonesRexII
- ✔ Twitter https://twitter.com/RexJonesII
- ✔ GitHub https://github.com/RexJonesII/Free-Videos
- ✔ LinkedIn https://www.linkedin.com/in/rexjones34/