VIDEO 6 : Exception Handling & Accurate Floats

Nobody wants their programs to crash. We should write code that anticipates the bad things a user may enter to cause a crash and eliminate them. However some times the user may try to access a database that doesn't exist, or they simply refuse to follow instructions. That is where exception handling comes in.

Through exception handling we can catch an error that would normally crash our program and give the user the opportunity to do the right thing.

In this 1st example we'll ask the user to enter a number, but if they refuse we will ask again by catching and solving the error.

We will surround our problematic code with a try block. We will then try to catch the expected error in an except block. If the user does something completely unexpected we will use an expect block without defining a specific exception to catch all other exceptions.

CODE

```
# By giving the while a value of True it will cycle until a break is reached
while True:

    # If we expect an error can occur surround potential error with try
    try:
        number = int(input("Please enter a number : "))
        break

    # The code in the except block provides an error message to set things right
    # We can either target a specific error like ValueError
    except ValueError:
        print("You didn't enter a number")

    # We can target all other errors with a default
    except:
        print("An unknown error occurred")

print("Thank you for entering a number")
```

As we continue we will cover other common exceptions you should know how to catch and solve.

Python Problem for you to Solve

I showed you a new trick you can use with a while loop above. Now I want you to implement a Do While loop in Python using that trick along with break, which you learned about in the last video.

The rules for a Do While loop is that they always execute all of the code at least once. After that 1st loop if the condition is true they will run that code again.

In other programming languages they have the format of

```
do {
```

```
... Bunch of code ...
} while(condition)
```

I want you to create a guessing game in which the user must chose a number between 1 and 10 with the following format.

```
Guess a number between 1 and 10 : 1
Guess a number between 1 and 10 : 3
Guess a number between 1 and 10 : 5
Guess a number between 1 and 10 : 7
You guessed it
```

Try your best. The goal is to get you to think in new ways and to understand the final solution. Hint : You'll need a while and break.

Solution

CODE

```
secret_number = 7

while True:
    guess = int(input("Guess a number between 1 and 10 : "))

    if guess == secret_number:
        print("You guessed it")
        break
```

More Accurate Floats

If you are not satisfied with 15 decimals of procession in your floating post calculations then you're in luck.

The decimal module provides for more accurate floating point calculations. With from you can reference methods without the need to reference the module like we had to do with math in a previous video. 28 points of precision by default.

We can also create an alias being D here to avoid conflicts with methods with the same name.

```
from decimal import Decimal as D
```

Here is some example code you can use to work with accurate floats.

CODE

```
sum = D(0)
sum += D("0.01")
sum += D("0.01")
sum += D("0.01")
sum -= D("0.03")

print("Sum = ", sum)
```

That’s all for this video. In the next video I’ll cover strings in detail and of course I’ll have more problems for you to solve.