Exceptions V

Raising your own exception and processing an exception multiple times

So far all the exceptions we have seen have occurred because the program does something that is not acceptable to Python. For example, trying to convert a string to an int, opening a file that does not exist, using an index value that is outside the bounds, etc. However, sometimes, we may want to define as an error something that is legal from the standpoint of Python but not legal for our program. In this case, we can 'raise' our own exceptions.
In this notebook we look at how we can explictly generate exceptions. We will also study how to handle the same exception multiple times.

Write a function that will ask the user to provides two integers and returns the exponent of the first number raised to the second. That is, given two numbers x and y, the function will return x raised to y. Suppose, we want to do this only for positive x values. We can define our functions so that it raises an exception when the user enters a negative value for x.

The syntax to do that is: `raise` Name of Exception

In [13]:

```
'''
  Compute Exponent function
'''
import sys
def compute_exponent():
    try:
        n1 = int(input('Enter the first number: '))
        if n1 <= 0:
            raise ValueError # This forces an execption whenever the user enters a negative value
for n1
        n2 = int(input('Enter the second number: '))
        return n1**n2
    except ValueError:
        print('Only positive numbers are valid!!!')
        # sys.exit()

compute_exponent()
```

```
Enter the first number: -5
Only positive numbers are valid!!!
```

You can process exceptions multiple times regardless of whether they were automatically raised or explicitly raised by you. That is once an exception occurs (automatically or because you raised it), you can process it immediately (in the same function) and then also pass it to the calling function for more exception handling.

Write a function that calls the `compute_exponent()` function that we wrote above. However, this time, if the user enters a negative value, the `compute_exponent()` function should print a message and then raise the exception to the calling function to further process it.

In [14]:

```
'''
  Raise Exception and pass to calling function
'''
def compute_exponent():
    try:
        n1 = int(input('Enter the first number: '))
        if n1 <= 0:
            raise ValueError
        print('no error')
        n2 = int(input('Enter the second number: '))
        return n1**n2
    except ValueError:
        print('\nOnly positive numbers are valid!!!\nPrinting from compute_exponent\n')
        raise ValueError #We raise the exception again and pass it on to the calling function for
further handling
```

```
def call_compute():
    try:
        print(compute_exponent())
    except ValueError:
        print('Error in data entry. Cant process further.\nPrinting from call_compute')

call_compute()
```

```
Enter the first number: -2

Only positive numbers are valid!!!
Printing from compute_exponent

Error in data entry. Cant process further.
Printing from call_compute
```

What will happen upon execution of this code?

In [ ]:

```
'''
  Hint: Look at except block
'''
def compute_exponent():
    try:
        n1 = int(input('Enter the first number: '))
        if n1 <= 0:
            raise ValueError
        print('no error')
        n2 = int(input('Enter the second number: '))
        return n1**n2
    except ValueError:
        print('\nOnly positive numbers are valid!!!\nPrinting from compute_exponent\n')
        raise ValueError #We raise the exception again and pass it on to the calling function for
further handling

def call_compute():
    try:
        print(compute_exponent())
    except ZeroDivisionError:
        print('Error in data entry. Cant process further.\nPrinting from call_compute')

call_compute()
```