**Syntax Error:** As the name suggests this error is caused by the wrong syntax in the code. It leads to the termination of the program.

1]Initialize the amount variable=10000 check that You are eligible to

purchase Dsa Self Paced or not using if condition.

# initialize the amount variable

amount **=** 10000

# check that You are eligible to

#  purchase Dsa Self Paced or not

**if**(amount > 2999)

print("You are eligible to purchase Dsa Self Paced")

**Exceptions:** Exceptions are raised when the program is syntactically correct, but the code resulted in an error. This error does not stop the execution of the program, however, it changes the normal flow of the program.

# initialize the amount variable

marks **=** 10000

# perform division with 0

a **=** marks **/** 0

print(a)

**Try and Except Statement – Catching Exceptions**

|  |
| --- |
| # Python program to handle simple runtime error  #Python 3    a **=** [1, 2, 3]  **try**:  **print** ("Second element = %d" **%**(a[1]))        # Throws error since there are only 3 elements in array  **print** ("Fourth element = %d" **%**(a[3]))    **except**:      print ("An error occurred") |

In the above example, the statements that can cause the error are placed inside the try statement (second print statement in our case). The second print statement tries to access the fourth element of the list which is not there and this throws an exception. This exception is then caught by the except statement.

The try block will generate an exception, because x is not defined:

try:  
  print(x)  
except:  
  print("An exception occurred")

Since the try block raises an error, the except block will be executed.

Without the try block, the program will crash and raise an error:

## Many Exceptions

You can define as many exception blocks as you want, e.g. if you want to execute a special block of code for a special kind of error:

### **Example**

Print one message if the try block raises a NameError and another for other errors:

try:  
  print(x)  
except NameError:  
  print("Variable x is not defined")  
except:  
  print("Something else went wrong")

## Else

You can use the else keyword to define a block of code to be executed if no errors were raised:

### **Example**

In this example, the try block does not generate any error:

try:  
  print("Hello")  
except:  
  print("Something went wrong")  
else:  
  print("Nothing went wrong")

## Finally

The finally block, if specified, will be executed regardless if the try block raises an error or not.

### **Example**

try:  
  print(x)  
except:  
  print("Something went wrong")  
finally:  
  print("The 'try except' is finished")

### **Example**

Try to open and write to a file that is not writable:

try:  
  f = open("demofile.txt")  
  try:  
    f.write("Lorum Ipsum")  
  except:  
    print("Something went wrong when writing to the file")  
  finally:  
    f.close()  
except:  
  print("Something went wrong when opening the file")

## Raise an exception

As a Python developer you can choose to throw an exception if a condition occurs.

To throw (or raise) an exception, use the raise keyword.

### **Example**

Raise an error and stop the program if x is lower than 0:

x = -1  
  
if x < 0:  
  raise Exception("Sorry, no numbers below zero")

### **Example**

Raise a TypeError if x is not an integer:

x = "hello"  
  
if not type(x) is int:  
  raise TypeError("Only integers are allowed")