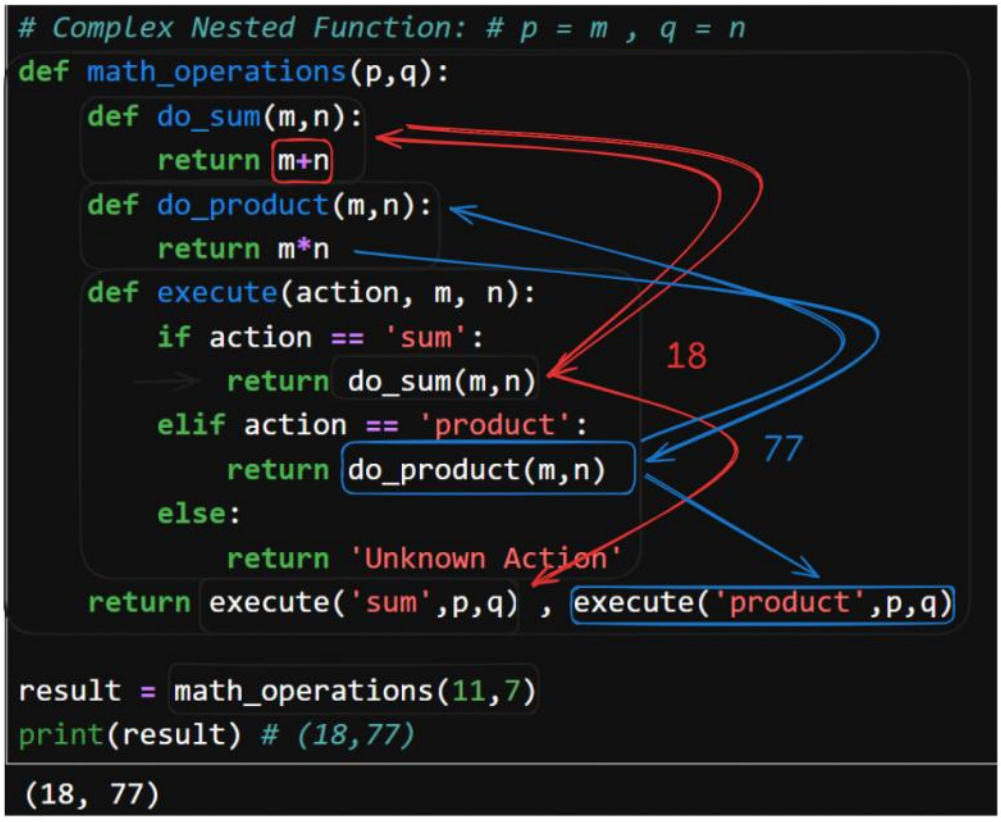
Functions & Exception Handling

Session Objectives:

- Understand recursion
- Use lambda (anonymous) functions
- Understand Python's exception handling model
- Apply try, except, else, finally, and raise statements effectively

```python
# Complex Nested Function: # p = m , q = n
def math_operations(p,q):
    def do_sum(m,n):
        return m+n
    def do_product(m,n):
        return m*n
    def execute(action, m, n):
        if action == 'sum':
            return do_sum(m,n)
        elif action == 'product':
            return do_product(m,n)
        else:
            return 'Unknown Action'
    return execute('sum',p,q) , execute('product',p,q)

result = math_operations(11,7)
print(result) # (18,77)
```

```
(18, 77)
```



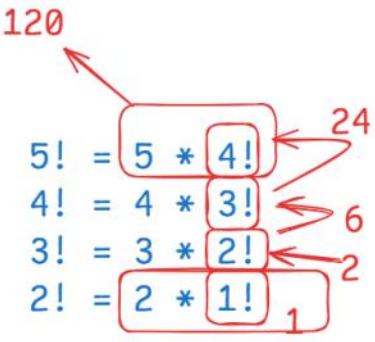
Memory

```
p,m = 11
q,n = 7
action = 'sum'
         'product'
```

What is Recursion?

Recursion is when a function calls itself to solve a smaller instance of the same problem. It consists of:

1. Base Case : Stops the recursion.
2. Recursive Case : The function calls itself with smaller inputs.



Factorial(n) → fact(n) = n * fact(n-1)

fact(n-1) = n-1 * fact(n-2)

fact(n-2) = n-2 * fact(n-3)

fact(n-3) = n-3 * fact(n-4)

fact(n-4) = n-4 * fact(n-5)

Factorial

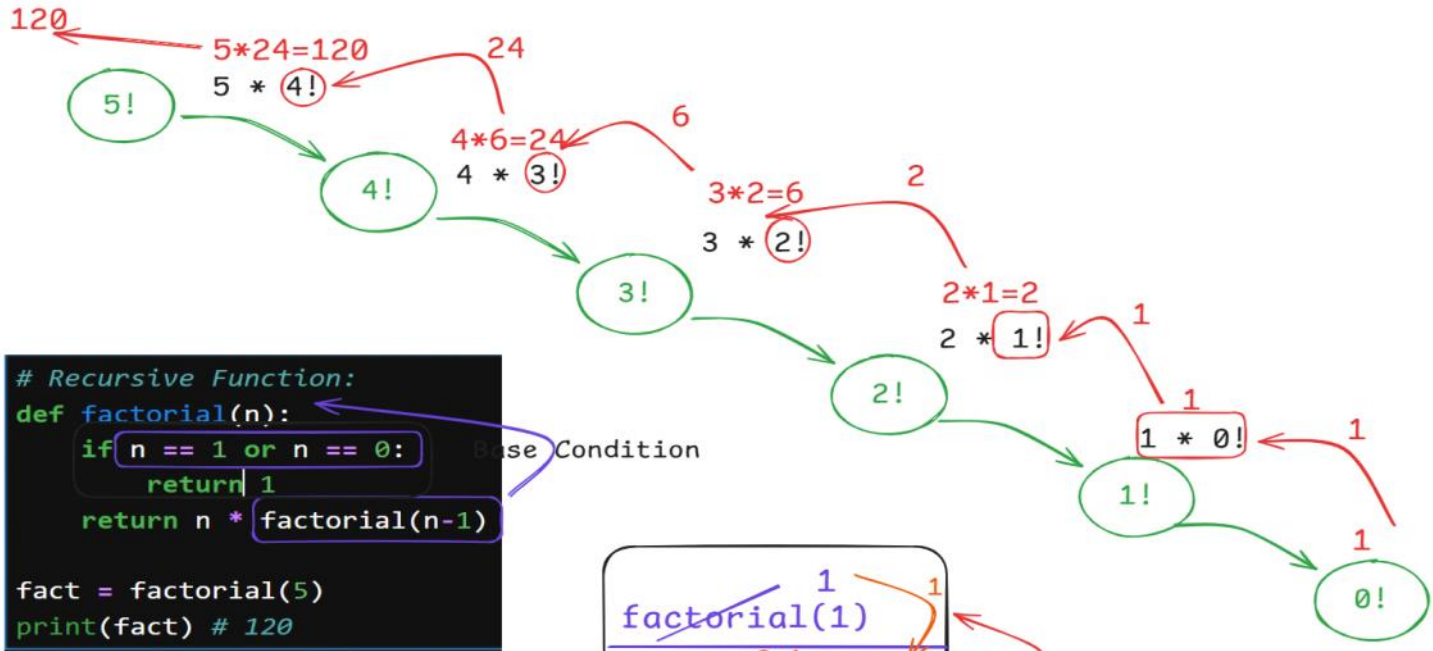
$n! = n * (n-1) * (n-2) * (n-3) * \cdots * 3 * 2 * 1$

0! = 1
1! = 1
2! = 2 x 1 = 2
3! = 3 x 2 x 1 = 6
4! = 4 x 3 x 2 x 1 = 24

Base Case

0! = 1
1! = 1

Faith + Expectation



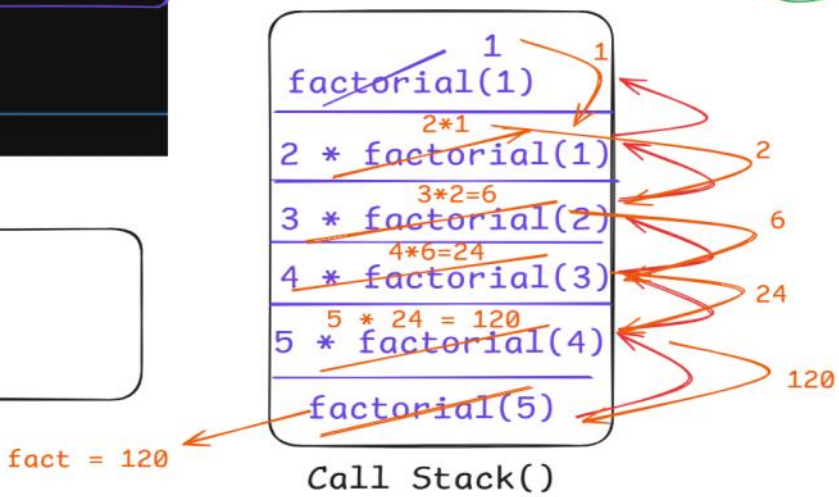

```
# Recursive Function:
def factorial(n):
    if n == 1 or n == 0:
        return 1
    return n * factorial(n-1)

fact = factorial(5)
print(fact) # 120
```

Base Condition

120

Memory

n = 5 ~~4~~ ~~3~~ ~~2~~ 1



fact = 120

Call Stack()

```python
# Iterative Approach:
def factorial(n): # n = 7
    result = 1
    for i in range(2,n+1): # [2,3,4,5,6,7]
        result *= i # 1*2*3*4*5*6*7 = 7!
    return result

factorial(5)
```

```
120
```

```python
factorial(7)
```

```
5040
```

What is a Lambda Function?

A lambda is a short, one-line anonymous function used for small operations without using def.

Syntax:

```
lambda arguments: expression
```

What is a Lambda Function?

A lambda is a short, one-line anonymous function used for small operations without using def.

Syntax:

```
lambda arguments: expression
```

```python
square = lambda val : val ** 2
square
```

```
<function __main__.<lambda>(val)>
```

```python
square(11)
```

```
121
```

```python
square(10)
```

```
100
```

```python
square(21)
```

```
441
```

```
division = lambda a,b : a/b
division
```

```
<function __main__.<lambda>(a, b)>
```

```
division(10,2)
```

```
5.0
```

```
division(7,2)
```

```
3.5
```

```
division(10,3)
```

```
3.3333333333333335
```

```
remainder = lambda p,q : p%q
remainder
```

```
<function __main__.<lambda>(p, q)>
```

```
remainder(17,3) # 2
```

```
2
```

```
add_10 = lambda val : val + 10
add_10
```

```
<function __main__.<lambda>(val)>
```

```
add_10(111)
```

```
121
```

```
# Lambda with Higher Order Functions
def multiplier_factory(factor):
    return lambda val : val * factor

triple_value = multiplier_factory(3)
# triple_value = lambda val : val * 3
triple_value(10)
```

```
30
```

```
triple_value
```

```
<function __main__.multiplier_factory.<locals>.<lambda>(val)>
```

```
# Lambda with Higher Order Functions
def multiplier_factory(factor):
    return lambda val : val * factor

double_value = multiplier_factory(2)
# double_value = lambda val : val * 2
double_value
```

```
<function __main__.multiplier_factory.<locals>.<lambda>(val)>
```

```
double_value(11)
```

```
22
```

What are Exceptions?

Exceptions are errors that disrupt the flow of your program. Common ones include:

Error	Description
`SyntaxError`	Invalid code structure
`IndentationError`	Wrong indentation
`TypeError`	Wrong data type
`NameError`	Using undefined variables
`ValueError`	Invalid value
`IndexError`	Out-of-range index
`KeyError`	Missing dictionary key
`AttributeError`	Missing object method/attr
`ZeroDivisionError`	Division by 0

try-except-else-finally: Error Handling Structure

Block	Purpose
`try`	Code that might raise error
`except`	Handle specific error types
`else`	Runs if `try` succeeds
`finally`	Always runs (cleanup etc.)

```
try:
    num = int(input("Please Enter a Divisor: ")
    result = 100/num
    print(f"The Calculated Value is {result}.")
except SyntaxError:
    print("Fix the Syntax.")
```

```
  Cell In[23], line 2
    num = int(input("Please Enter a Divisor: ")
              ^
SyntaxError: '(' was never closed
```

Fix Code

SyntaxError can't be handled with Except Block

```python
try:
    num = int(input("Please Enter a Divisor: "))
    result = 100/num
    print(f"The Calculated Value is {result}.")
except SyntaxError:
    print("Fix the Syntax.")
```

```
Please Enter a Divisor:  10
The Calculated Value is 10.0.
```

```python
try:
    num = int(input("Please Enter a Divisor: "))
    result = 100/num
    print(f"The Calculated Value is {result}.")
except SyntaxError:
    print("Fix the Syntax.")
```

```
Please Enter a Divisor:  a
```

Fix Code

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[25], line 2
      1 try:
----> 2     num = int(input("Please Enter a Divisor: "))
      3     result = 100/num
      4     print(f"The Calculated Value is {result}.")

ValueError: invalid literal for int() with base 10: 'a'
```

```python
try:
    num = int(input("Please Enter a Divisor: "))
    result = 100/num
    print(f"The Calculated Value is {result}.")
except ValueError:
    print("This is not a Valid Number, Please Enter the Digit Only...")
```

```
Please Enter a Divisor:  a
This is not a Valid Number, Please Enter the Digit Only...
```

```
try:
    num = int(input("Please Enter a Divisor: "))
    result = 100/num
    print(f"The Calculated Value is {result}.")
except ValueError:
    print("This is not a Valid Number, Please Enter the Digit Only...")
```

```
Please Enter a Divisor:  0
```

Fix Code

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[27], line 3
      1 try:
      2     num = int(input("Please Enter a Divisor: "))
----> 3     result = 100/num
      4     print(f"The Calculated Value is {result}.")
      5 except ValueError:

ZeroDivisionError: division by zero
```

```
try:
    num = int(input("Please Enter a Divisor: "))
    result = 100/num
    print(f"The Calculated Value is {result}.")
except ValueError:
    print("This is not a Valid Number, Please Enter the Digit Only...")
except ZeroDivisionError:
    print("Division by zero is not allowed....")
```

```
Please Enter a Divisor:  0
Division by zero is not allowed....
```

```
# Multiple Exception in One Block:
try:
    num = int(input("Please Enter a Divisor: "))
    result = 100/num
    print(f"The Calculated Value is {result}.")
except (ValueError,ZeroDivisionError):
    print("Please Enter a Valid Number and Avoid Dividing by Zero...")
```

```
Please Enter a Divisor:  a
Please Enter a Valid Number and Avoid Dividing by Zero...
```

```
# Multiple Exception in One Block:
try:
    num = int(input("Please Enter a Divisor: "))
    result = 100/num
    print(f"The Calculated Value is {result}.")
except (ValueError,ZeroDivisionError):
    print("Please Enter a Valid Number and Avoid Dividing by Zero...")
```

```
Please Enter a Divisor:  0
Please Enter a Valid Number and Avoid Dividing by Zero...
```

```python
# Nested Try-Except:
try:
    num = int(input("Please Enter a Divisor: "))
    try:
        result = 100/num
        print(f"The Calculated Value is {result}.")
    except ZeroDivisionError:
        print("Division by zero is not allowed....")
except ValueError:
    print("This is not a Valid Number, Please Enter the Digit Only...")
```

```
Please Enter a Divisor:  z
This is not a Valid Number, Please Enter the Digit Only...
```

```python
# Nested Try-Except:
try:
    num = int(input("Please Enter a Divisor: "))
    try:
        result = 100/num
        print(f"The Calculated Value is {result}.")
    except ZeroDivisionError:
        print("Division by zero is not allowed....")
except ValueError:
    print("This is not a Valid Number, Please Enter the Digit Only...")
```

```
Please Enter a Divisor:  0
Division by zero is not allowed....
```

```python
# Avoding error type is not recommended. Use Specific Exception Names for clarity and debugging.
try:
    num = int(input("Please Enter a Divisor: "))
    result = 100/num
    print(f"The Calculated Value is {result}.")
except :
    print("Please Enter a Valid Number and Avoid Dividing by Zero....")
```

```
Please Enter a Divisor:  50
The Calculated Value is 2.0.
```

try-except-else Statement

What is the else block?

The else block runs only if no exceptions are raised in the try block.

Syntax:

```
try:
    # Code that might raise an exception
except Exception1:
    # Handle Exception1
except Exception2:
    # Handle Exception2
else:
    # Runs ONLY if no exception occurs
```

```
try:
    num = int(input("Please Enter a Divisor: "))
    result = 100/num
    print(f"The Calculated Value is {result}.")
except ValueError:
    print("This is not a Valid Number, Please Enter the Digit Only...")
except ZeroDivisionError:
    print("Division by zero is not allowed....")
else:
    print(f"Operations Completed Successfully. The result is {result}.")
```

```
Please Enter a Divisor:  5
The Calculated Value is 20.0.
Operations Completed Successfully. The result is 20.0.
```

```
try:
    num = int(input("Please Enter a Divisor: "))
    result = 100/num
    print(f"The Calculated Value is {result}.")
except ValueError:
    print("This is not a Valid Number, Please Enter the Digit Only...")
except ZeroDivisionError:
    print("Division by zero is not allowed....")
else:
    print(f"Operations Completed Successfully. The result is {result}.")
```

```
Please Enter a Divisor:  abc
This is not a Valid Number, Please Enter the Digit Only...
```

```
# Index Error:
car_list = ['Taigun','Creata','Safari','Innova','Thar']
try:
    val = int(input("Enter a valid index: "))
    print(car_list[val]) # Index Error
else:
    print("Code Run Successfully")
```

```
  Cell In[36], line 6
    else:
    ^
SyntaxError: expected 'except' or 'finally' block
```

Fix Code

```
# Index Error:
car_list = ['Taigun','Creata','Safari','Innova','Thar']

val = int(input("Enter a valid index: "))
print(car_list[val]) # Index Error
```

```
Enter a valid index:  7
```

Fix Code

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[37], line 5
      2 car_list = ['Taigun','Creata','Safari','Innova','Thar']
      4 val = int(input("Enter a valid index: "))
----> 5 print(car_list[val])

IndexError: list index out of range
```

```
# Index Error:
car_list = ['Taigun','Creata','Safari','Innova','Thar']
try:
    val = int(input("Enter a valid index: "))
    print(car_list[val]) # Index Error
except IndexError:
    print("IndexError: The Position you are trying to access doesn't exist.")
else:
    print("Code Run Successfully")
```

```
Enter a valid index:  7
IndexError: The Position you are trying to access doesn't exist.
```

```python
# Index Error:
car_list = ['Taigun','Creata','Safari','Innova','Thar']
try:
    val = int(input("Enter a valid index: ")) # 2
    print(car_list[val]) # 'Safari'
except IndexError:
    print("IndexError: The Position you are trying to access doesn't exist.")
else:
    print("Code Run Successfully")
```

```
Enter a valid index:  2
Safari
Code Run Successfully
```

```python
# Attribute Error: # Calling wrong methods
person = {
    'name' : 'Shyam Sundar',
    'age' : 29,
    'city' : 'Mumbai'
}
try:
    person.add('state', 'Maharashtra') # Attribute Error
except AttributeError:
    print("Attribute Error: `dict` object has no method name `.add()`.")
else:
    print("No Attribute Error Occurred, Your Try Block Run Successfully.")
```

```
Attribute Error: `dict` object has no method name `.add()`.
```

```python
# Attribute Error: # Calling wrong methods
person = {
    'name' : 'Shyam Sundar',
    'age' : 29,
    'city' : 'Mumbai'
}
try:
    person['state'] = 'Maharashtra'
    print(person)
    print("Key-Value Pair Added Successfully.")
except AttributeError:
    print("Attribute Error: `dict` object has no method name `.add()`.")
else:
    print("No Attribute Error Occurred, Your Try Block Run Successfully.")
```

```
{'name': 'Shyam Sundar', 'age': 29, 'city': 'Mumbai', 'state': 'Maharashtra'}
Key-Value Pair Added Successfully.
No Attribute Error Occurred, Your Try Block Run Successfully.
```

```python
# Attribute Error:
# Even and Odd Number
_list = [11,21,23,44,11,221,12,19,22,29,77,10,100,-50,500,51,False,True]
# False = 0
# True = 1
# List -> .append() , .insert() , .extend()
# Set -> .add() , .update()
# Dictionary -> .setdefault(), .update()
try:
    even_set = set()
    odd_set = set()
    for val in _list:
        if val % 2 == 1: # Odd
            odd_set.append(val) # Attribute Error
        elif val % 2 == 0: # 'Even'
            even_set.add(val)
    print(even_set)
    print(odd_set) # No 11 as duplicates allowed, Also final set will be shuffled
except AttributeError:
    print("Attribute Error: `set` object has no method name `.append()`.")
else:
    print("No Attribute Error Occurred, Your Try Block Run Successfully.")
```

```
Attribute Error: `set` object has no method name `.append()`.
```

```python
# Attribute Error:
# Even and Odd Number
_list = [11,21,23,44,11,221,12,19,22,29,77,10,100,-50,500,51,False,True]
# False = 0
# True = 1
# List -> .append() , .insert() , .extend()
# Set -> .add() , .update()
# Dictionary -> .setdefault(), .update()
try:
    even_set = set()
    odd_set = set()
    for val in _list:
        if val % 2 == 1: # Odd
            odd_set.add(val) # No Attribute Error
        elif val % 2 == 0: # 'Even'
            even_set.add(val)
    print(even_set)
    print(odd_set) # No 11 as duplicates allowed, Also final set will be shuffled
except AttributeError:
    print("Attribute Error: `set` object has no method name `.append()`.")
else:
    print("No Attribute Error Occurred, Your Try Block Run Successfully.")
```

```
{False, 100, 10, 44, 12, -50, 500, 22}
{True, 11, 77, 29, 19, 51, 21, 23, 221}
No Attribute Error Occurred, Your Try Block Run Successfully.
```

```python
# Attribute Error: # Calling wrong methods
person = {
    'name' : 'Shyam Sundar',
    'age' : 29,
    'city' : 'Mumbai'
}
try:
    person['state'] = 'Maharashtra'
    print(Person) # 'NameError'
    print("Key-Value Pair Added Successfully.")
except AttributeError:
    print("Attribute Error: `dict` object has no method name `.add()`.")
else:
    print("No Attribute Error Occurred, Your Try Block Run Successfully.")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[44], line 9
      7 try:
      8     person['state'] = 'Maharashtra'
----> 9     print(Person) # 'NameError'
     10     print("Key-Value Pair Added Successfully.")
     11 except AttributeError:

NameError: name 'Person' is not defined
```

Fix Code

```python
# Name Error: # Name is not defined in memory
person = {
    'name' : 'Shyam Sundar',
    'age' : 29,
    'city' : 'Mumbai'
}
try:
    person['state'] = 'Maharashtra'
    print(Person) # 'NameError'
    print("Key-Value Pair Added Successfully.")
except AttributeError:
    print("Attribute Error: `dict` object has no method name `.add()`.")
except NameError:
    print("The variable you are calling doesn't exist.")
else:
    print("No Attribute Error Occurred, Your Try Block Run Successfully.")
```

```
The variable you are calling doesn't exist.
```

```python
# Key Error
person = {
    'name' : 'Shyam Sundar',
    'age' : 29,
    'city' : 'Mumbai'
}
try:
    person['state'] = 'Maharashtra'
    print(person['City']) # 'KeyError'
    print("Key-Value Pair Added Successfully.")
except AttributeError:
    print("Attribute Error: `dict` object has no method name `.add()`.")
except NameError:
    print("The variable you are calling doesn't exist.")
else:
    print("No Attribute Error Occurred, Your Try Block Run Successfully.")
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[47], line 9
      7 try:
      8     person['state'] = 'Maharashtra'
----> 9     print(person['City']) # 'KeyError'
     10     print("Key-Value Pair Added Successfully.")
     11 except AttributeError:

KeyError: 'City'
```

Fix Code

```python
# Key Error
person = {
    'name' : 'Shyam Sundar',
    'age' : 29,
    'city' : 'Mumbai'
}
try:
    person['state'] = 'Maharashtra'
    print(person['City']) # 'KeyError'
    print("Key-Value Pair Added Successfully.")
except AttributeError:
    print("Attribute Error: `dict` object has no method name `.add()`.")
except NameError:
    print("The variable you are calling doesn't exist.")
except KeyError:
    print("The Key, you are looking for, doesn't exist in your person dictionary.")
else:
    print("No Attribute Error Occurred, Your Try Block Run Successfully.")
```

```
The Key, you are looking for, doesn't exist in your person dictionary.
```

```python
# Indentation Error:
# Index Error:
car_list = ['Taigun','Creata','Safari','Innova','Thar']
try:
    for car in carlist:
    print(car)
except IndentationError:
    print("Expected an indedtated block after the 'for' statement.")
```

```
  Cell In[50], line 6
    print(car)
    ^
IndentationError: expected an indented block after 'for' statement on line 5
```

Fix Code

```python
# Indentation Error:
car_list = ['Taigun','Creata','Safari','Innova','Thar']
try:
    for car in car_list:
        print(car)
except:
    print("A Regular Except Block")
else:
    print("Code Run Successfully ✅")
# except IndentationError: # Avoid
#     print("Expected an indedtated block after the 'for' statement.")
```

```
Taigun
Creata
Safari
Innova
Thar
Code Run Successfully ✅
```

```python
# Type Error:
x = '3'
y = 5
print(x + y)
```

```
---------------------------------------------------------
TypeError                                 Traceback (most
Cell In[54], line 4
      2 x = '3'
      3 y = 5
----> 4 print(x + y)

TypeError: can only concatenate str (not "int") to str
```

Fix Code

```python
# Type Error:
try:
    x = '3'
    y = 5
    print(x + y)
except TypeError:
    print("Type Error")
```

```
Type Error
```

try-except-finally Statement

What is the finally block?

The finally block always runs, no matter what.

Even if:

- An exception occurs
- No exception occurs
- The program is interrupted with return, break, or raise

Syntax:

```
try:
    # Risky code
except ExceptionType:
    # Handle error
finally:
    # Always run this cleanup code
```

```python
# Key Error
person = {
    'name' : 'Shyam Sundar',
    'age' : 29,
    'city' : 'Mumbai'
}
try:
    person['state'] = 'Maharashtra'
    print(person['City']) # 'KeyError'
    print("Key-Value Pair Added Successfully.")
except AttributeError:
    print("Attribute Error: `dict` object has no method name `.add()`.")
except NameError:
    print("The variable you are calling doesn't exist.")
except KeyError:
    print("The Key, you are looking for, doesn't exist in your person dictionary.")
else:
    print("No Attribute Error Occurred, Your Try Block Run Successfully.")
finally:
    print("Finally will always run, No Matter what? 😎")
```

```
The Key, you are looking for, doesn't exist in your person dictionary.
Finally will always run, No Matter what? 😎
```

```python
# Key Error
person = {
    'name' : 'Shyam Sundar',
    'age' : 29,
    'city' : 'Mumbai'
}
try:
    person['state'] = 'Maharashtra'
    print(person['city']) # 'Mumbai'
    print("Key-Value Pair Added Successfully.")
except AttributeError:
    print("Attribute Error: `dict` object has no method name `.add()`.")
except NameError:
    print("The variable you are calling doesn't exist.")
except KeyError:
    print("The Key, you are looking for, doesn't exist in your person dictionary.")
else:
    print("No Attribute Error Occurred, Your Try Block Run Successfully.")
finally:
    print("Finally will always run, No Matter what? 😎")
```

```
Mumbai
Key-Value Pair Added Successfully.
No Attribute Error Occurred, Your Try Block Run Successfully.
Finally will always run, No Matter what? 😎
```

raise Keyword

What is raise?

The raise keyword lets you intentionally trigger an exception.

Syntax:

```
raise ExceptionType("Error message")
```

```python
# Raise Keyword
marks = int(input("Enter the valid marks: "))
if marks < 0:
    raise ValueError("Marks can't be negative.")
```

```
Enter the valid marks:  -11
```

Fix Code

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[58], line 4
      2 marks = int(input("Enter the valid marks: "))
      3 if marks < 0:
----> 4     raise ValueError("Marks can't be negative.")

ValueError: Marks can't be negative.
```

```python
# Raise Keyword
marks = int(input("Enter the valid marks: "))
if marks < 0:
    raise ValueError("Marks can't be negative.")
```

```
Enter the valid marks:  91
```

```python
# Raise Keyword
marks = int(input("Enter the valid marks: "))
if marks > 100:
    raise ValueError("Score Exceeds the allowed limit.")
```

```
Enter the valid marks:  121
```

Fix Code

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[60], line 4
      2 marks = int(input("Enter the valid marks: "))
      3 if marks > 100:
----> 4     raise ValueError("Score Exceeds the allowed limit.")

ValueError: Score Exceeds the allowed limit.
```

```python
temperature = int(input("Enter the Temperature in degree Celcius...")) # AQI for Temperature Measuring
if temperature > 40:
    raise ValueError("Temperature is above the safety threshold.")
```

```
Enter the Temperature in degree Celcius... 50
```

Fix Code

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[63], line 3
      1 temperature = int(input("Enter the Temperature in degree Celcius... )) # AQI for Temperature Measuring
      2 if temperature > 40:
----> 3     raise ValueError("Temperature is above the safety threshold.")

ValueError: Temperature is above the safety threshold.
```