# CN-Lec19-Exception Handling

## Exception Handling

### Session Objectives:

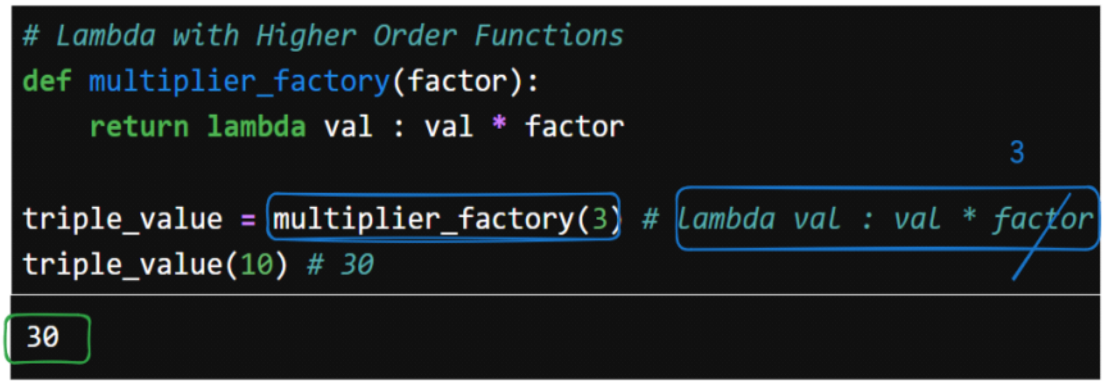
- ✓ Understand recursion
- ✓ Use lambda (anonymous) functions
- ✓ Understand Python's exception handling model
- ✓ Apply try, except, else, finally, and raise statements effectively

### What is a Lambda Function?

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

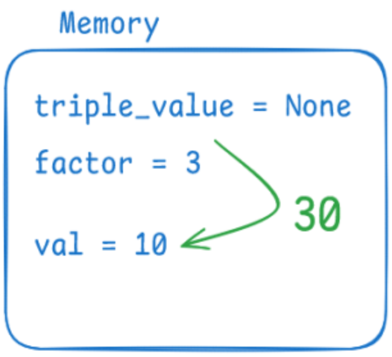
Syntax:

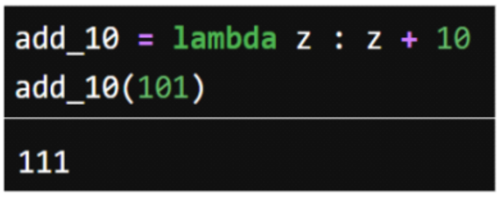
```
lambda arguments: expression
```



```
# Lambda with Higher Order Functions
def multiplier_factory(factor):
    return lambda val : val * factor
                                                      3
triple_value = multiplier_factory(3) # lambda val : val * factor
triple_value(10) # 30
```

```
30
```



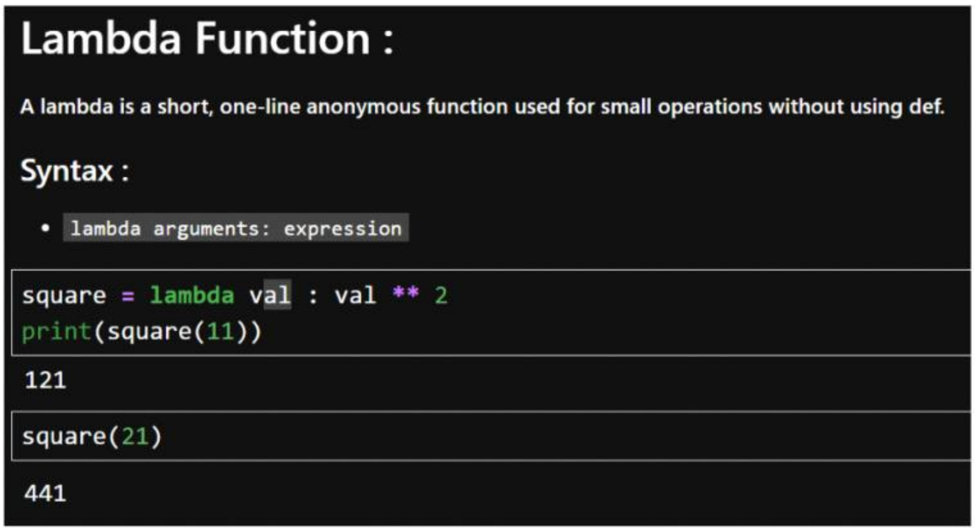


```
add_10 = lambda z : z + 10
add_10(101)
```

```
111
```

```
triple_value = lambda val : val * 3

triple_value(10)
```



# Lambda Function :

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

## Syntax :

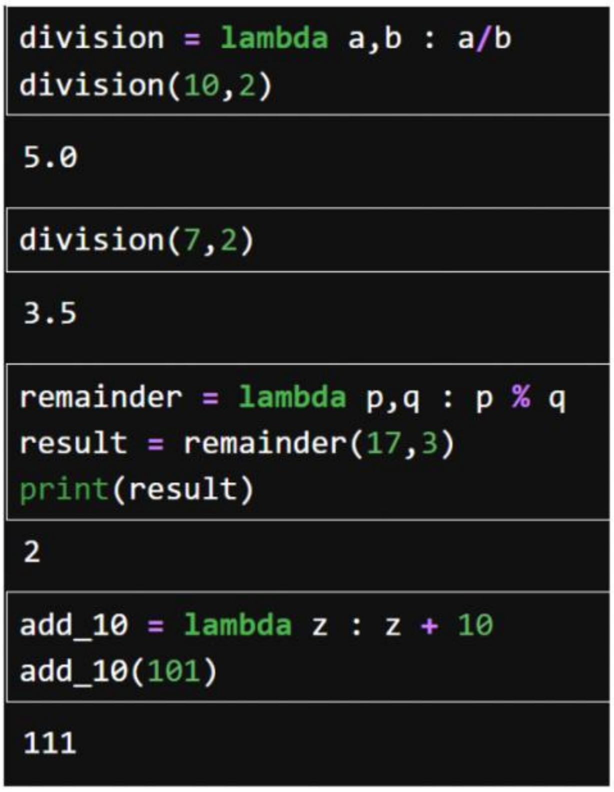
- `lambda arguments: expression`

```python
square = lambda val : val ** 2
print(square(11))
```

```
121
```

```python
square(21)
```

```
441
```



```python
division = lambda a,b : a/b
division(10,2)
```

```
5.0
```

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

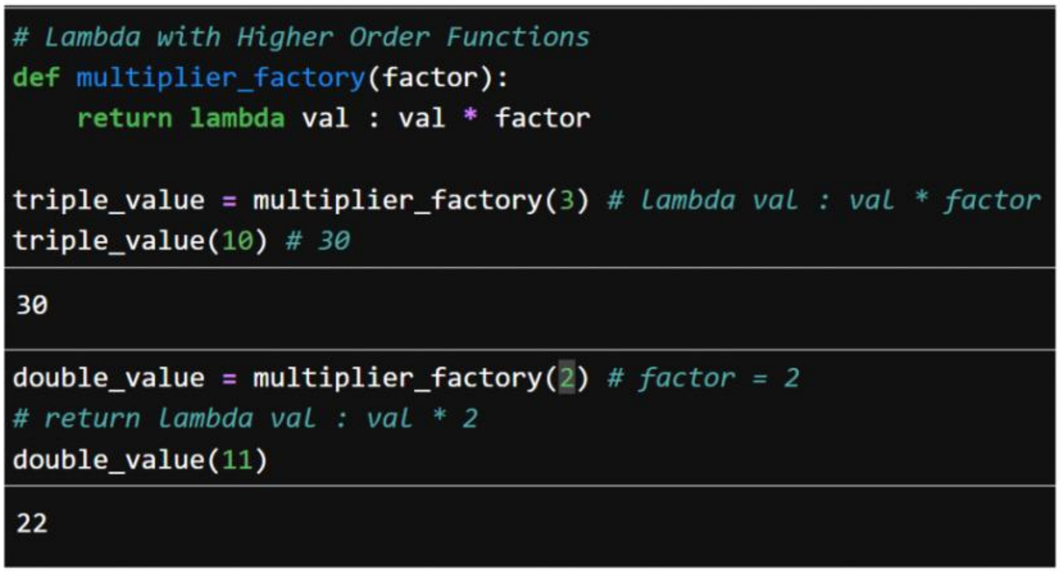
```
3.5
```

```python
remainder = lambda p,q : p % q
result = remainder(17,3)
print(result)
```

```
2
```

```python
add_10 = lambda z : z + 10
add_10(101)
```

```
111
```



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

triple_value = multiplier_factory(3) # lambda val : val * factor
triple_value(10) # 30
```

```
30
```

```python
double_value = multiplier_factory(2) # factor = 2
# return lambda val : val * 2
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 |

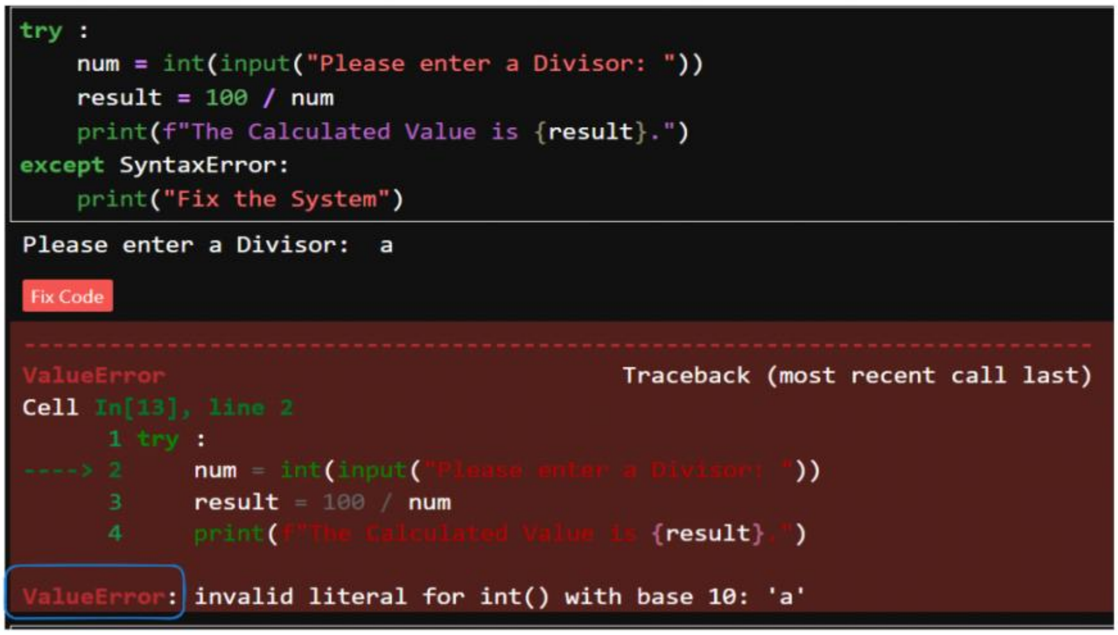
## try-except-else-finally: Error Handling Structure

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

```
try:
    pass
except <ErrorName> :
    pass
```

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

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

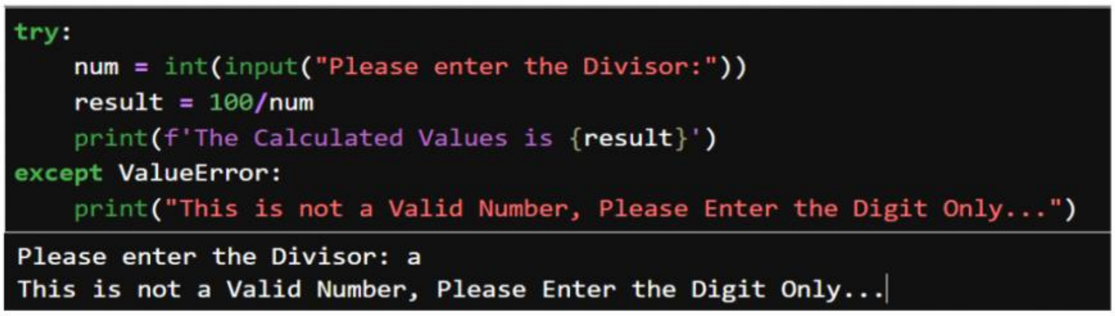


```
try :
    num = int(input("Please enter a Divisor: "))
    result = 100 / num
    print(f"The Calculated Value is {result}.")
except SyntaxError:
    print("Fix the System")
```

```
Please enter a Divisor:  a
```

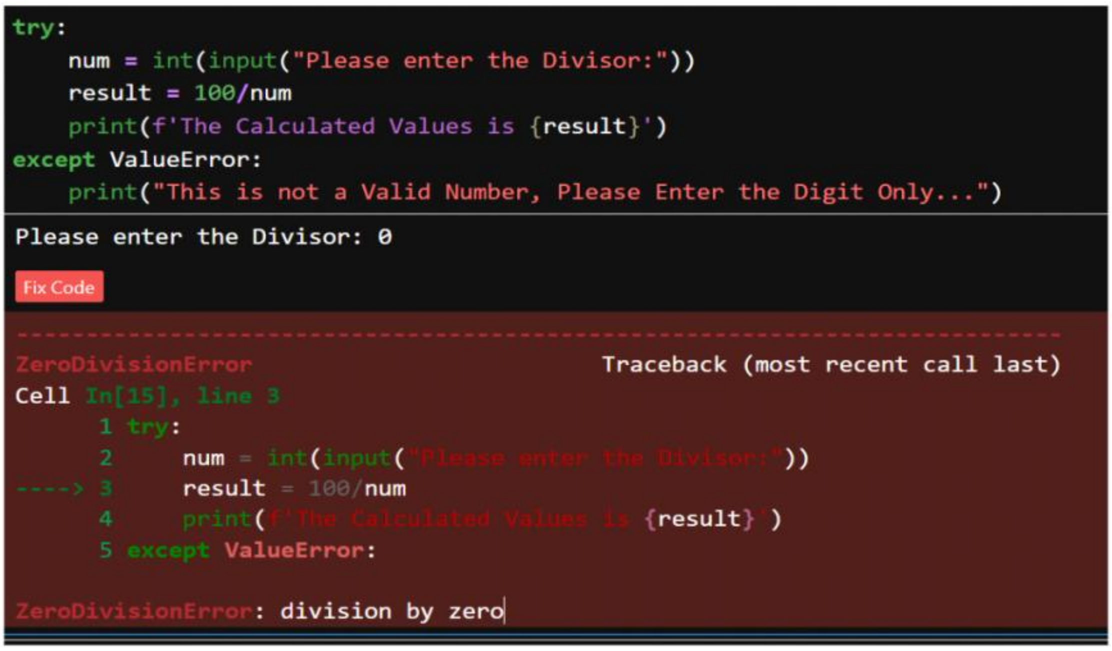
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[13], line 2
      1 try :
----> 2     num = int(input("Please enter a Divisor: "))
      3     result = 100 / num
      4     print(f"The Calculated Value is {result}.")

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



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

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

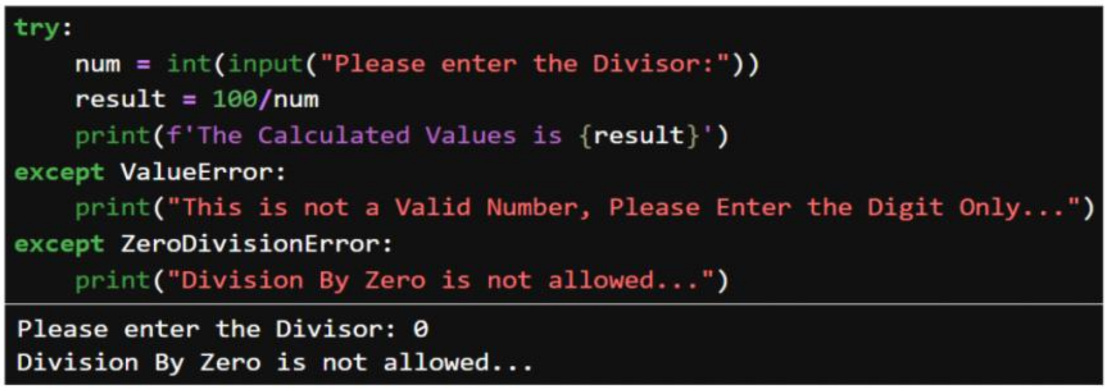


```
try:
    num = int(input("Please enter the Divisor:"))
    result = 100/num
    print(f'The Calculated Values is {result}')
except ValueError:
    print("This is not a Valid Number, Please Enter the Digit Only...")
```

```
Please enter the Divisor: 0
```

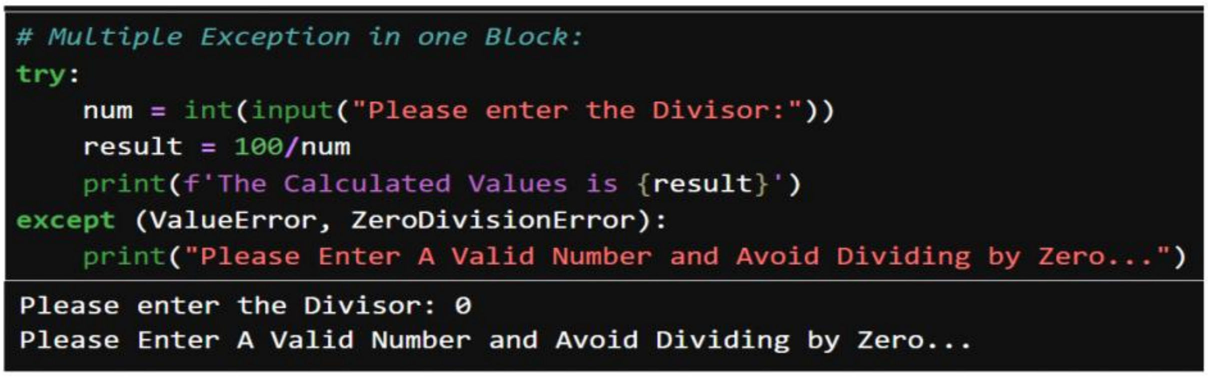
```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[15], line 3
      1 try:
      2     num = int(input("Please enter the Divisor:"))
----> 3     result = 100/num
      4     print(f'The Calculated Values is {result}')
      5 except ValueError:

ZeroDivisionError: division by zero
```



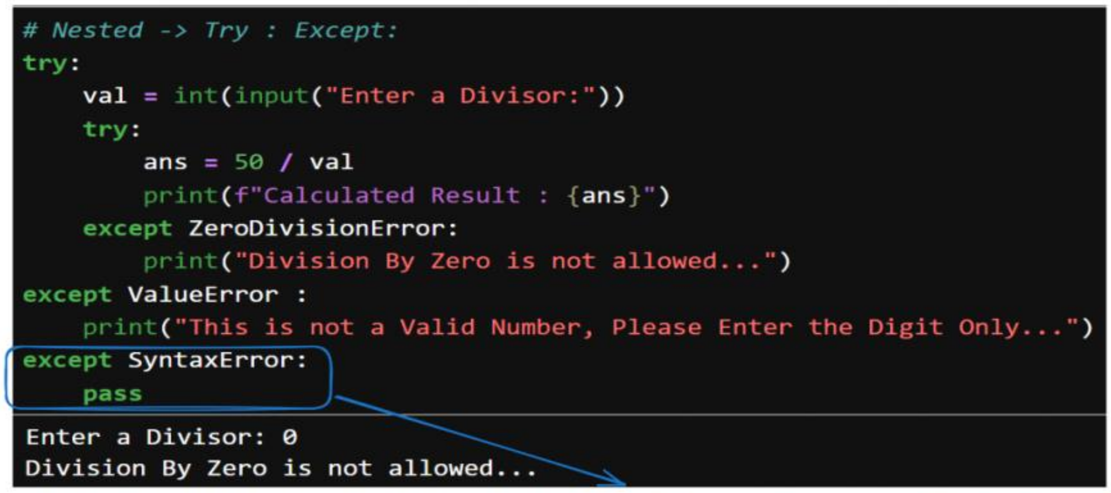
```
try:
    num = int(input("Please enter the Divisor:"))
    result = 100/num
    print(f'The Calculated Values is {result}')
except ValueError:
    print("This is not a Valid Number, Please Enter the Digit Only...")
except ZeroDivisionError:
    print("Division By Zero is not allowed...")
```

```
Please enter the Divisor: 0
Division By Zero is not allowed...
```



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

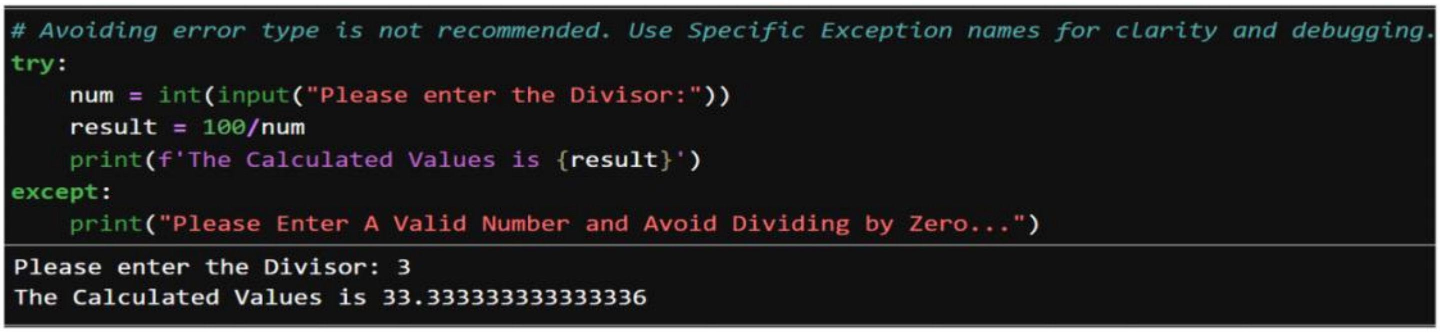
```
Please enter the Divisor: 0
Please Enter A Valid Number and Avoid Dividing by Zero...
```



```
# Nested -> Try : Except:
try:
    val = int(input("Enter a Divisor:"))
    try:
        ans = 50 / val
        print(f"Calculated Result : {ans}")
    except ZeroDivisionError:
        print("Division By Zero is not allowed...")
except ValueError :
    print("This is not a Valid Number, Please Enter the Digit Only...")
except SyntaxError:
    pass
```

```
Enter a Divisor: 0
Division By Zero is not allowed...
```

Syntax Error won't be handled
by try:except block.



```
# Avoiding error type is not recommended. Use Specific Exception names for clarity and debugging.
try:
    num = int(input("Please enter the Divisor:"))
    result = 100/num
    print(f'The Calculated Values is {result}')
except:
    print("Please Enter A Valid Number and Avoid Dividing by Zero...")
```

```
Please enter the Divisor: 3
The Calculated Values is 33.333333333333336
```

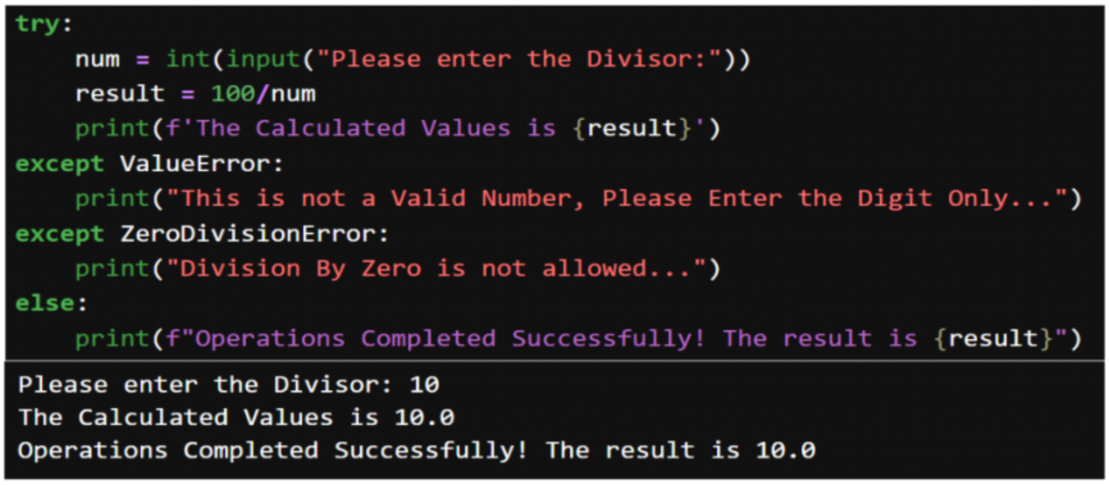
## try-except-else Statement

### What is the else block?

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

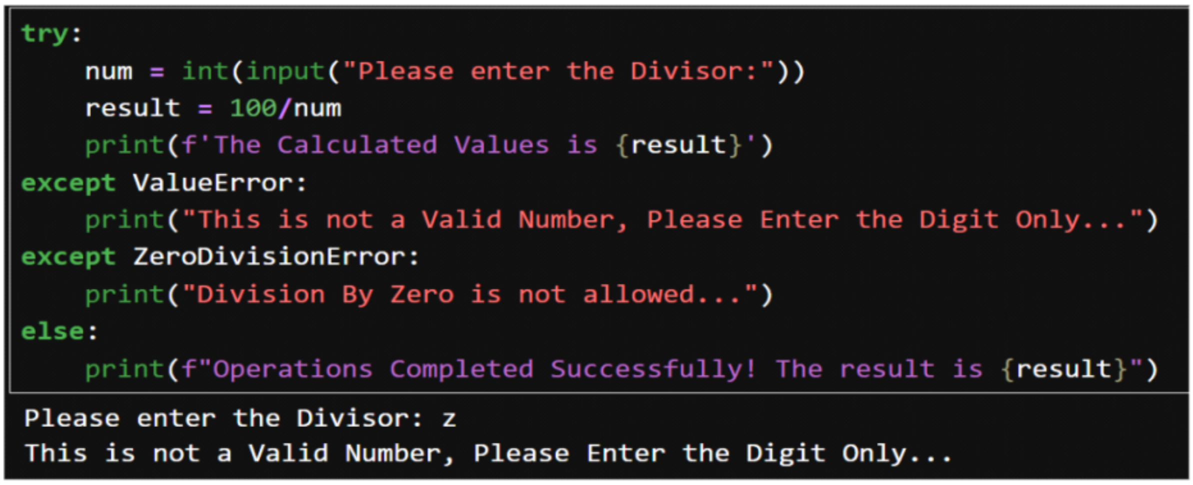
Syntax:

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



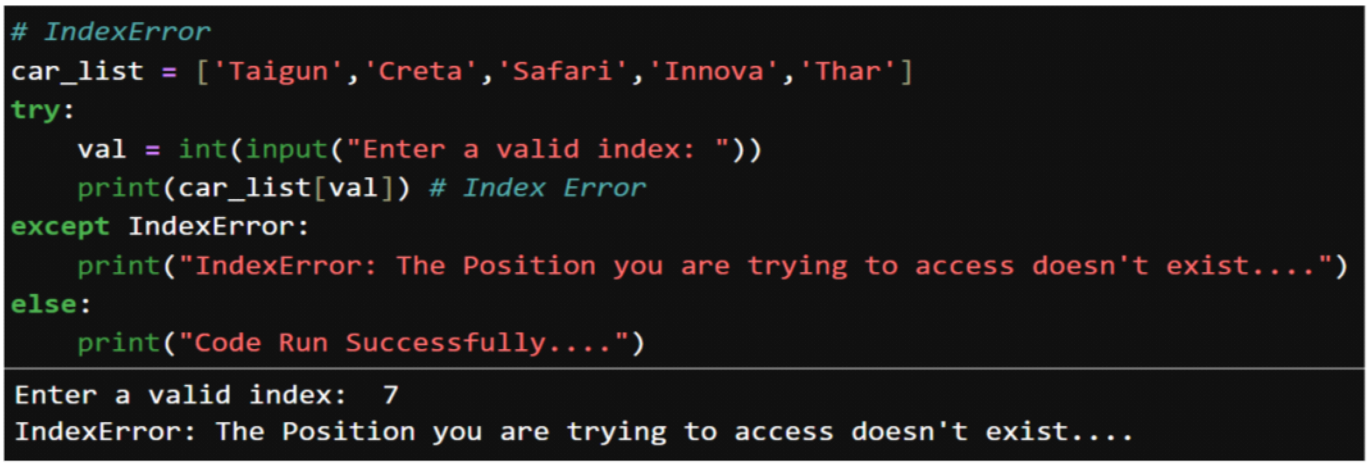
```
try:
    num = int(input("Please enter the Divisor:"))
    result = 100/num
    print(f'The Calculated Values is {result}')
except ValueError:
    print("This is not a Valid Number, Please Enter the Digit Only...")
except ZeroDivisionError:
    print("Division By Zero is not allowed...")
else:
 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

```
Please enter the Divisor: 10
The Calculated Values is 10.0
Operations Completed Successfully! The result is 10.0
```



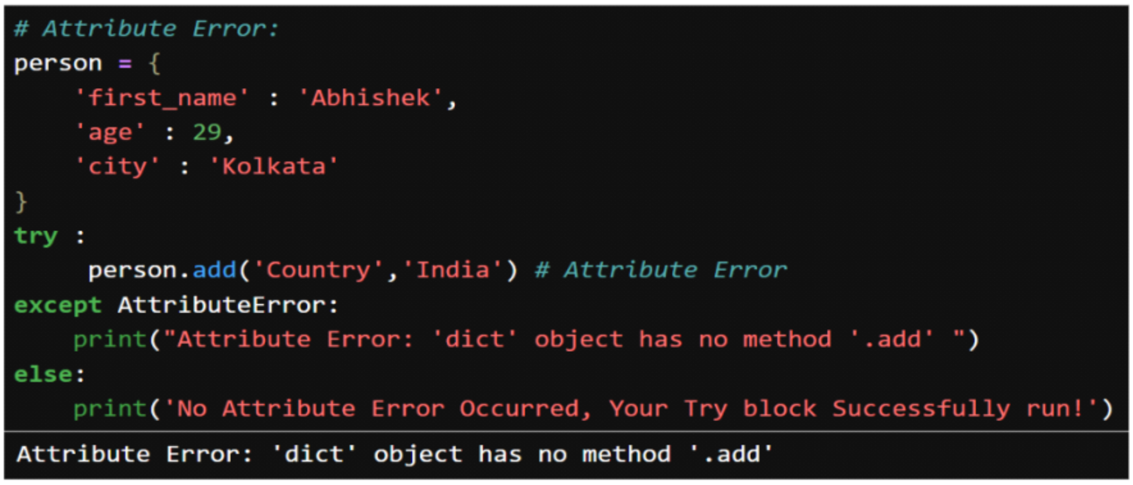
```
try:
    num = int(input("Please enter the Divisor:"))
    result = 100/num
    print(f'The Calculated Values is {result}')
except ValueError:
    print("This is not a Valid Number, Please Enter the Digit Only...")
except ZeroDivisionError:
    print("Division By Zero is not allowed...")
else:
 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

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



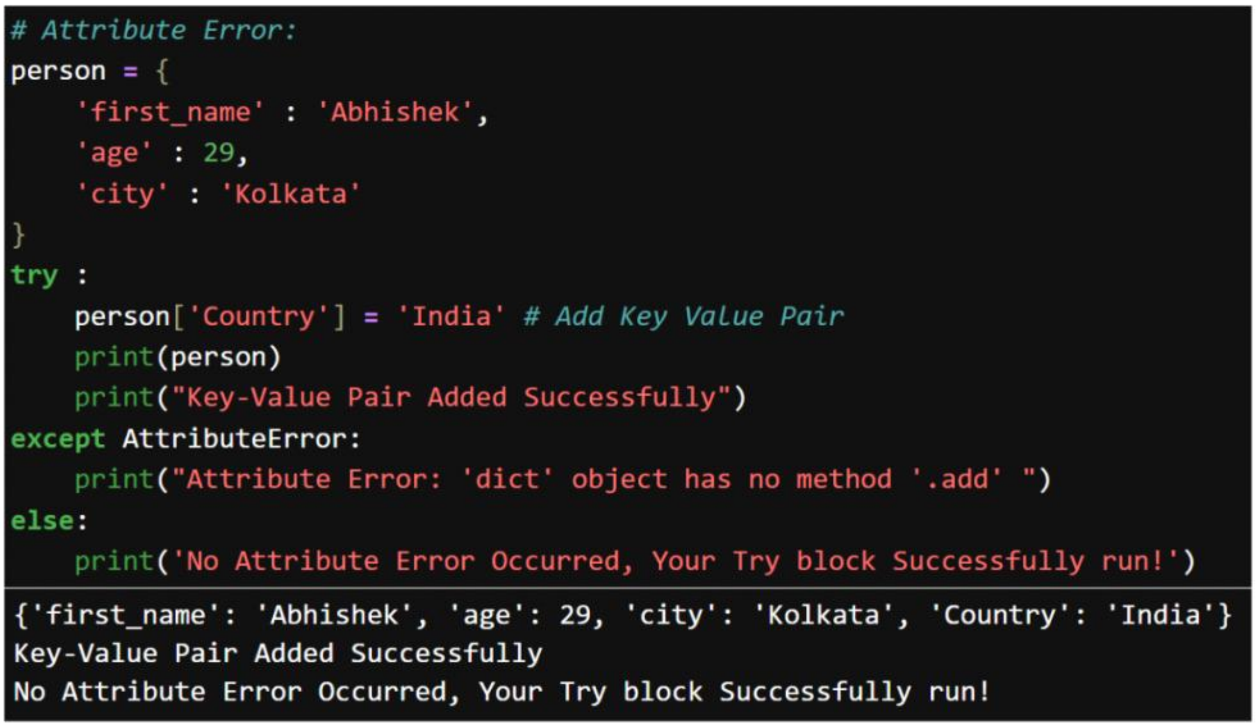
```
# IndexError
car_list = ['Taigun','Creta','Safari','Innova','Thar']
try:
    val = int(input("Enter a valid index: "))
    print(car_list[val]) # Index Error
except IndexError:
    print("IndexError: The Position you are trying to access doesn't exist....")
else:
    print("Code Run Successfully....")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

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



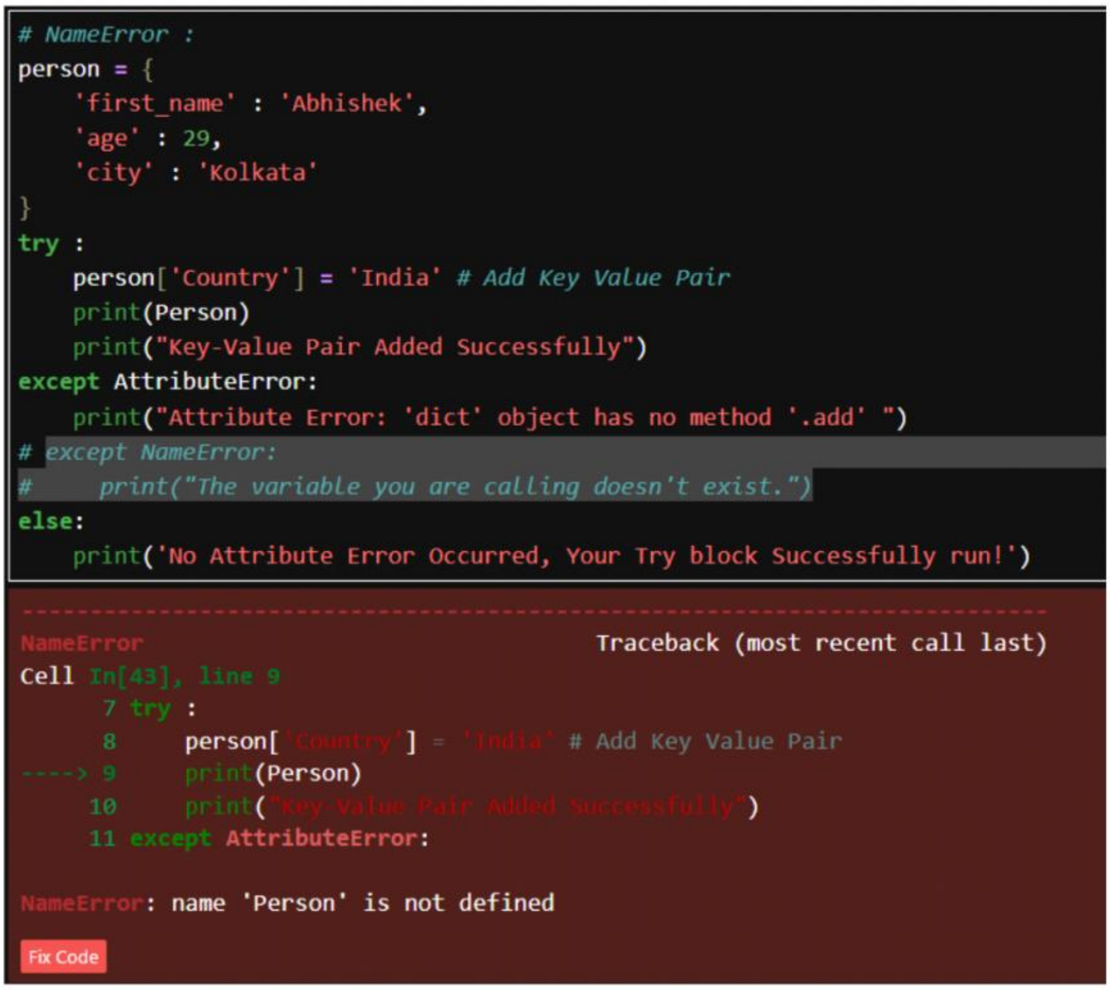
```
# Attribute Error:
person = {
    'first_name' : 'Abhishek',
    'age' : 29,
    'city' : 'Kolkata'
}
try :
     person.add('Country','India') # Attribute Error
except AttributeError:
    print("Attribute Error: 'dict' object has no method '.add' ")
else:
    print('No Attribute Error Occurred, Your Try block Successfully run!')
```

```
Attribute Error: 'dict' object has no method '.add'
```



```python
# Attribute Error:
person = {
    'first_name' : 'Abhishek',
    'age' : 29,
    'city' : 'Kolkata'
}
try :
    person['Country'] = 'India' # Add Key Value Pair
    print(person)
    print("Key-Value Pair Added Successfully")
except AttributeError:
    print("Attribute Error: 'dict' object has no method '.add' ")
else:
    print('No Attribute Error Occurred, Your Try block Successfully run!')
```

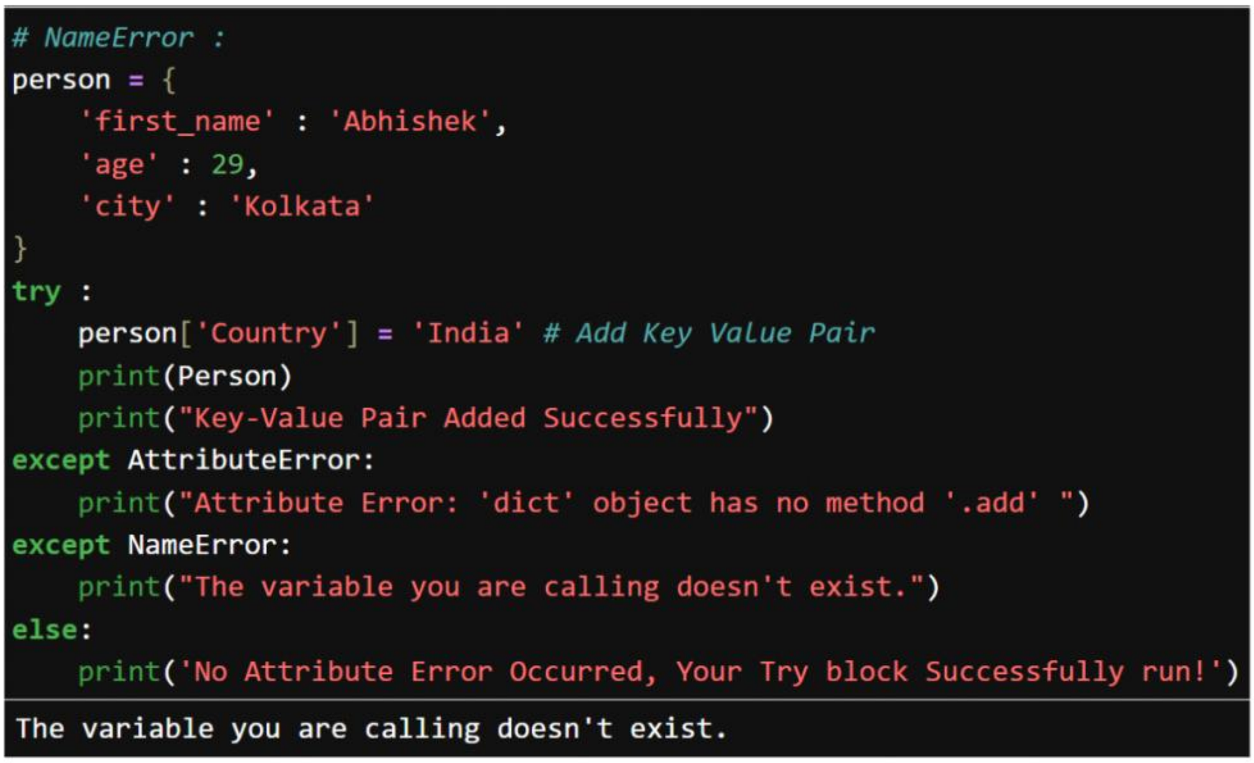
```
{'first_name': 'Abhishek', 'age': 29, 'city': 'Kolkata', 'Country': 'India'}
Key-Value Pair Added Successfully
No Attribute Error Occurred, Your Try block Successfully run!
```



```python
# NameError :
person = {
    'first_name' : 'Abhishek',
    'age' : 29,
    'city' : 'Kolkata'
}
try :
    person['Country'] = 'India' # Add Key Value Pair
    print(Person)
    print("Key-Value Pair Added Successfully")
except AttributeError:
    print("Attribute Error: 'dict' object has no method '.add' ")
# except NameError:
#     print("The variable you are calling doesn't exist.")
else:
    print('No Attribute Error Occurred, Your Try block Successfully run!')
```

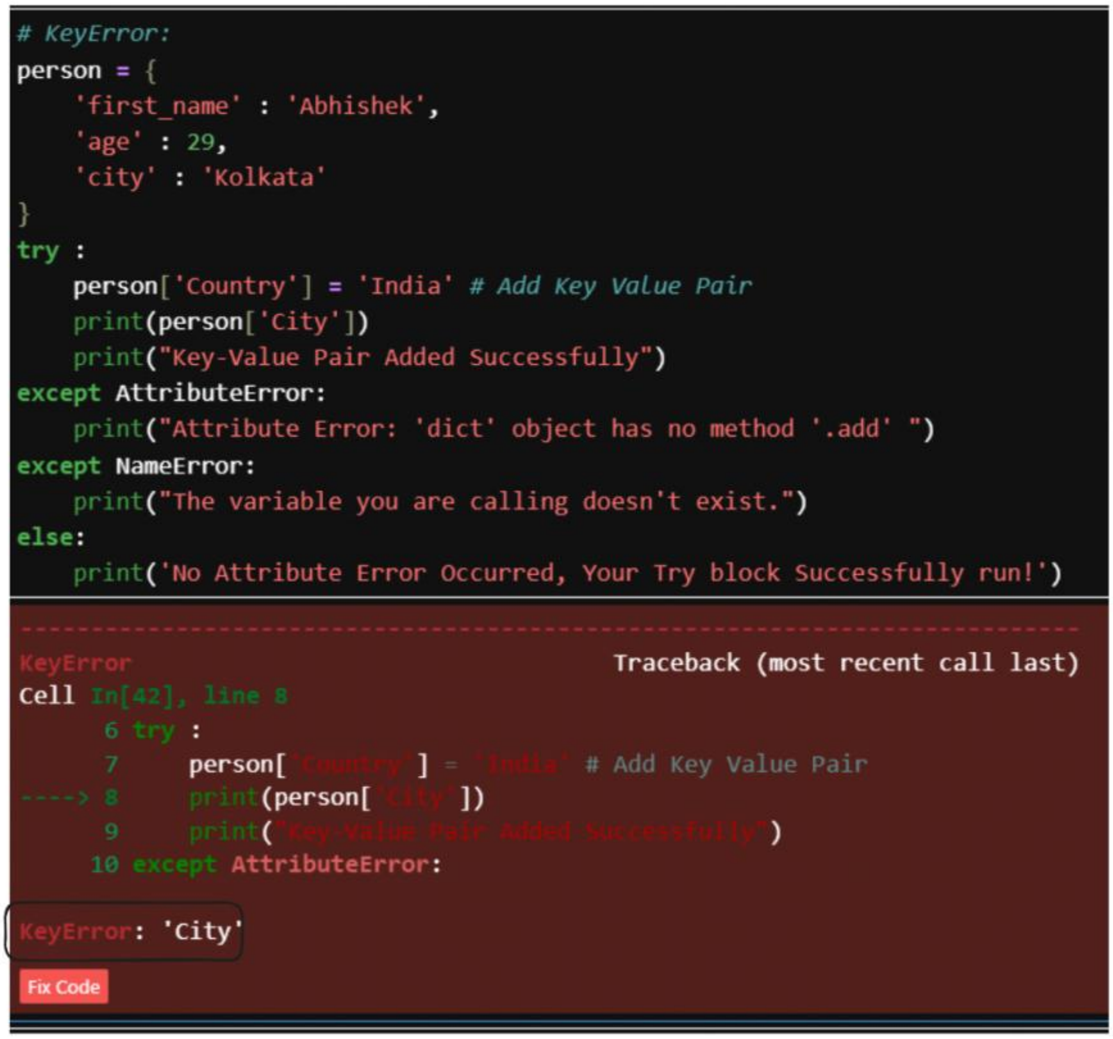
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[43], line 9
      7 try :
      8     person['Country'] = 'India' # Add Key Value Pair
----> 9     print(Person)
     10     print("Key-Value Pair Added Successfully")
     11 except AttributeError:

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



```
# NameError :
person = {
    'first_name' : 'Abhishek',
    'age' : 29,
    'city' : 'Kolkata'
}
try :
    person['Country'] = 'India' # Add Key Value Pair
    print(Person)
    print("Key-Value Pair Added Successfully")
except AttributeError:
    print("Attribute Error: 'dict' object has no method '.add' ")
except NameError:
    print("The variable you are calling doesn't exist.")
else:
    print('No Attribute Error Occurred, Your Try block Successfully run!')
```

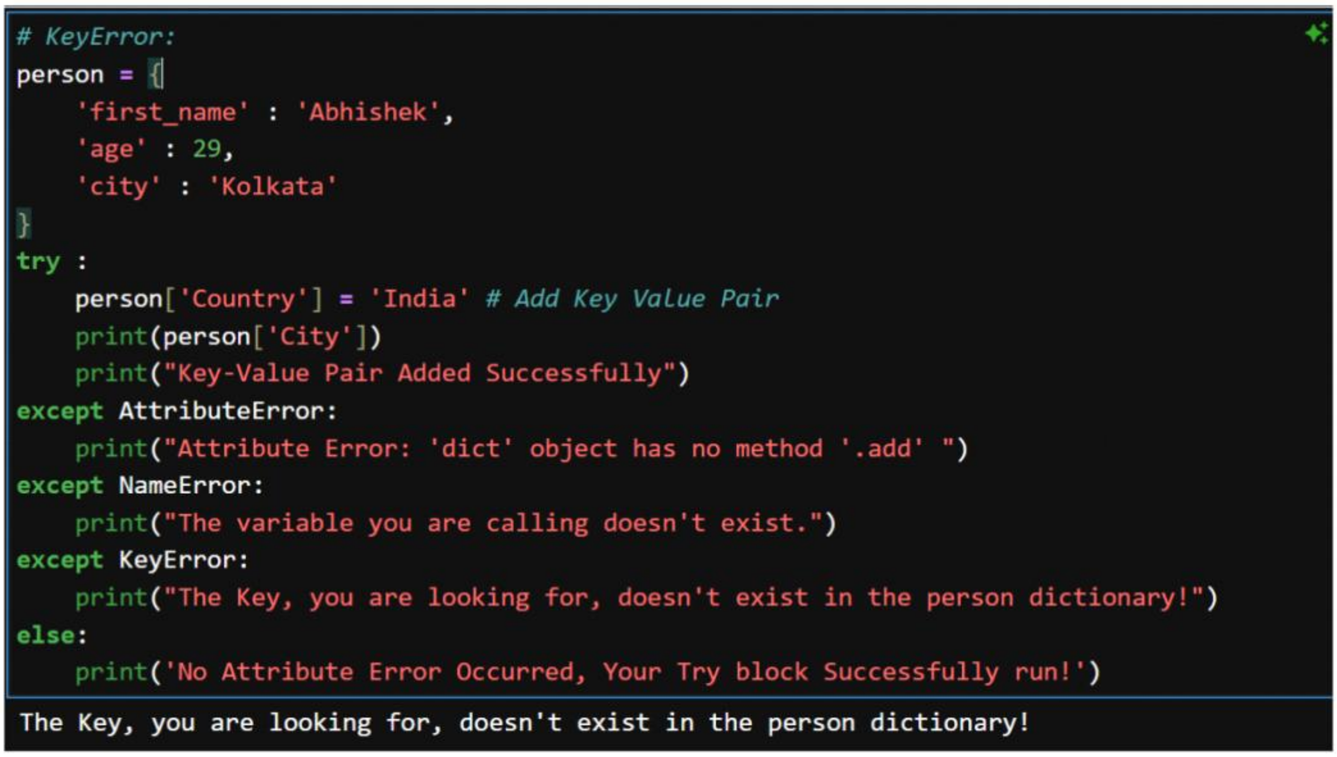
```
The variable you are calling doesn't exist.
```



```
# KeyError:
person = {
    'first_name' : 'Abhishek',
    'age' : 29,
    'city' : 'Kolkata'
}
try :
    person['Country'] = 'India' # Add Key Value Pair
    print(person['City'])
    print("Key-Value Pair Added Successfully")
except AttributeError:
    print("Attribute Error: 'dict' object has no method '.add' ")
except NameError:
    print("The variable you are calling doesn't exist.")
else:
    print('No Attribute Error Occurred, Your Try block Successfully run!')
```

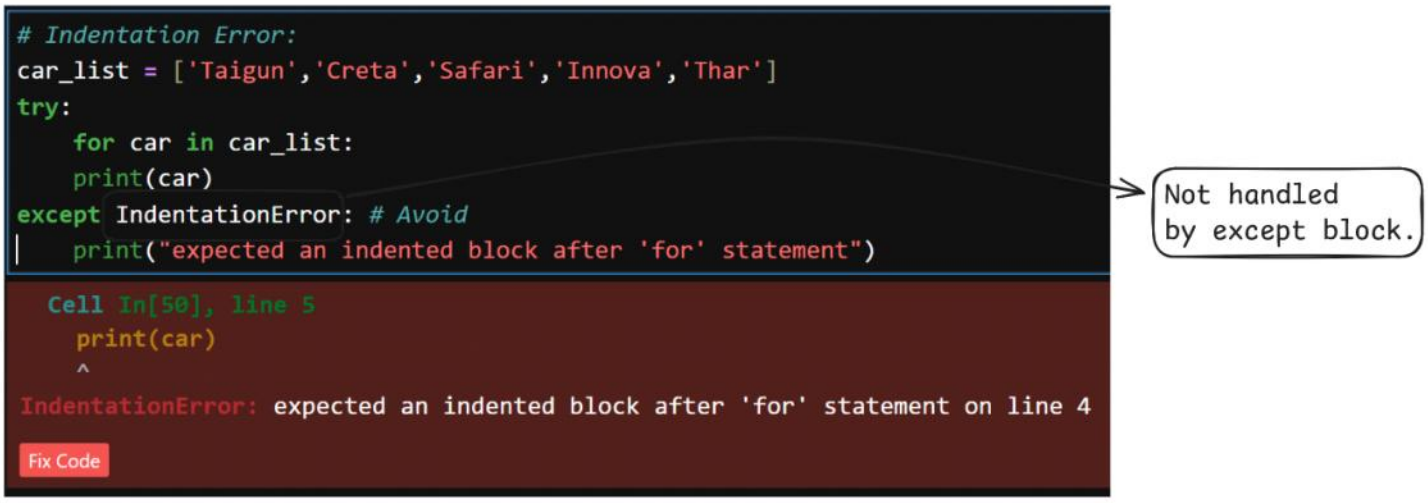
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[42], line 8
      6 try :
      7     person['Country'] = 'India' # Add Key Value Pair
----> 8     print(person['City'])
      9     print("Key-Value Pair Added Successfully")
     10 except AttributeError:

KeyError: 'City'
```



```python
# KeyError:
person = {
    'first_name' : 'Abhishek',
    'age' : 29,
    'city' : 'Kolkata'
}
try :
    person['Country'] = 'India' # Add Key Value Pair
    print(person['City'])
    print("Key-Value Pair Added Successfully")
except AttributeError:
    print("Attribute Error: 'dict' object has no method '.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 the person dictionary!")
else:
    print('No Attribute Error Occurred, Your Try block Successfully run!')
```

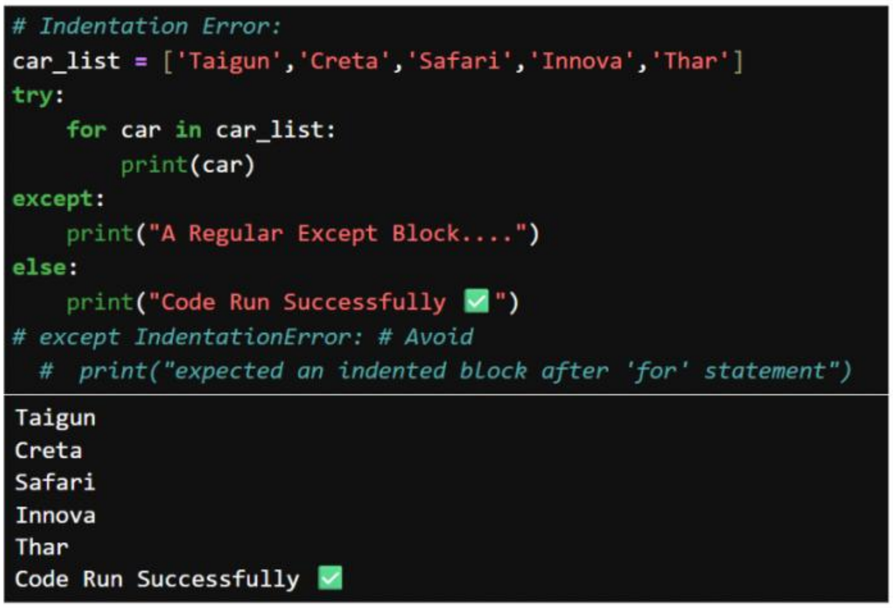
```
The Key, you are looking for, doesn't exist in the person dictionary!
```



```python
# Indentation Error:
car_list = ['Taigun','Creta','Safari','Innova','Thar']
try:
    for car in car_list:
    print(car)
except IndentationError: # Avoid
    print("expected an indented block after 'for' statement")
```

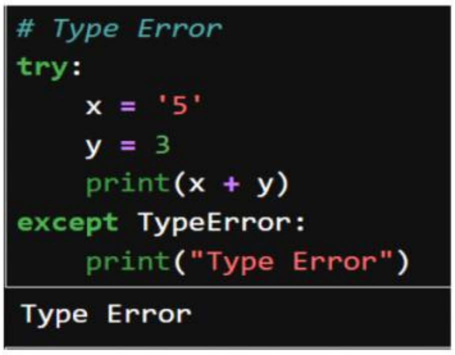
Not handled by except block.

```
  Cell In[50], line 5
    print(car)
    ^
IndentationError: expected an indented block after 'for' statement on line 4
```



```python
# Indentation Error:
car_list = ['Taigun','Creta','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 indented block after 'for' statement")
```

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



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

```
Type Error
```

## try-except-finally Statement

### What is the finally block?

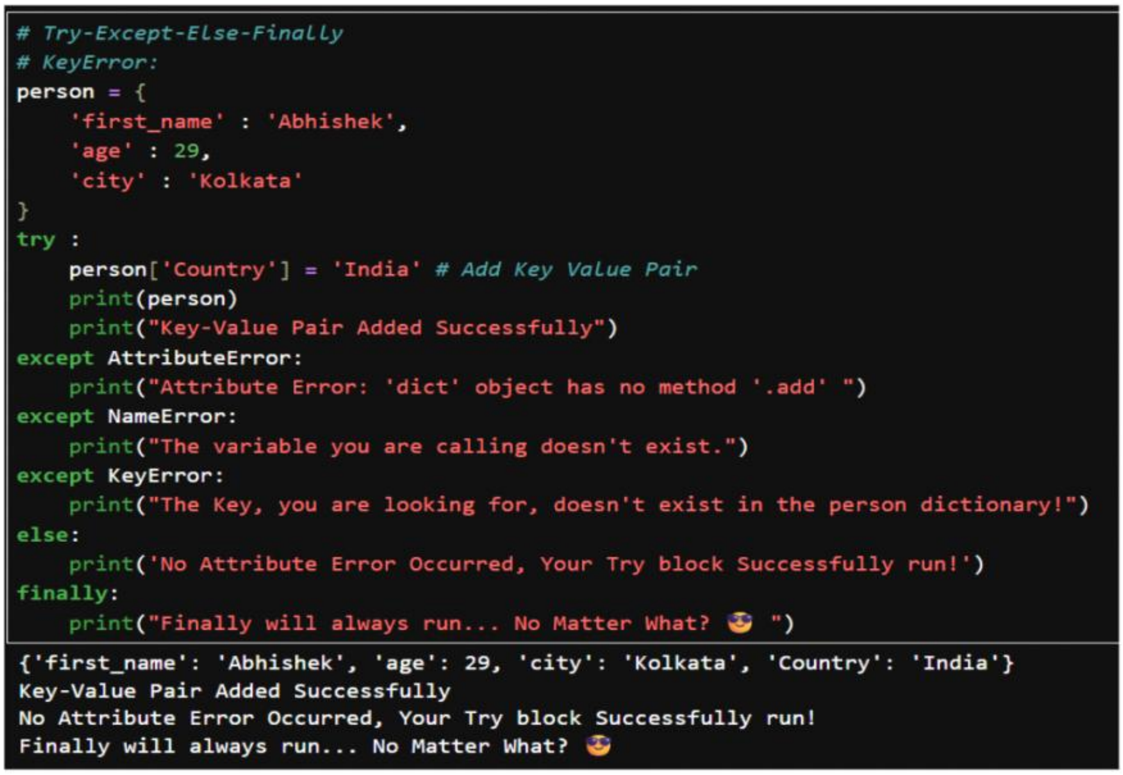
The finally block always runs, no matter what.

Even if:

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

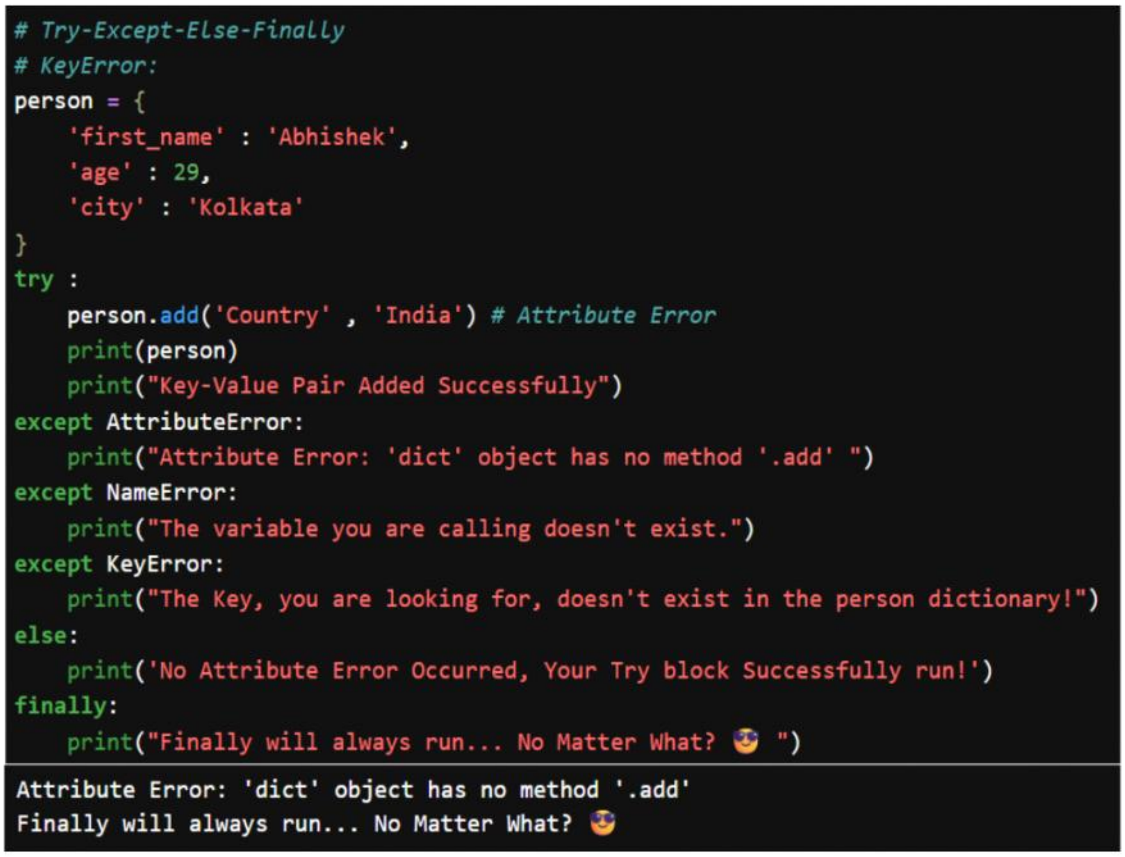
Syntax:

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



```python
# Try-Except-Else-Finally
# KeyError:
person = {
    'first_name' : 'Abhishek',
    'age' : 29,
    'city' : 'Kolkata'
}
try :
    person['Country'] = 'India' # Add Key Value Pair
    print(person)
    print("Key-Value Pair Added Successfully")
except AttributeError:
    print("Attribute Error: 'dict' object has no method '.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 the person dictionary!")
else:
    print('No Attribute Error Occurred, Your Try block Successfully run!')
finally:
    print("Finally will always run... No Matter What? 😎 ")
```

```
{'first_name': 'Abhishek', 'age': 29, 'city': 'Kolkata', 'Country': 'India'}
Key-Value Pair Added Successfully
No Attribute Error Occurred, Your Try block Successfully run!
Finally will always run... No Matter What? 😎
```



```python
# Try-Except-Else-Finally
# KeyError:
person = {
    'first_name' : 'Abhishek',
    'age' : 29,
    'city' : 'Kolkata'
}
try :
    person.add('Country' , 'India') # Attribute Error
    print(person)
    print("Key-Value Pair Added Successfully")
except AttributeError:
    print("Attribute Error: 'dict' object has no method '.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 the person dictionary!")
else:
    print('No Attribute Error Occurred, Your Try block Successfully run!')
finally:
    print("Finally will always run... No Matter What? 😎 ")
```

```
Attribute Error: 'dict' object has no method '.add'
Finally will always run... No Matter What? 😎
```

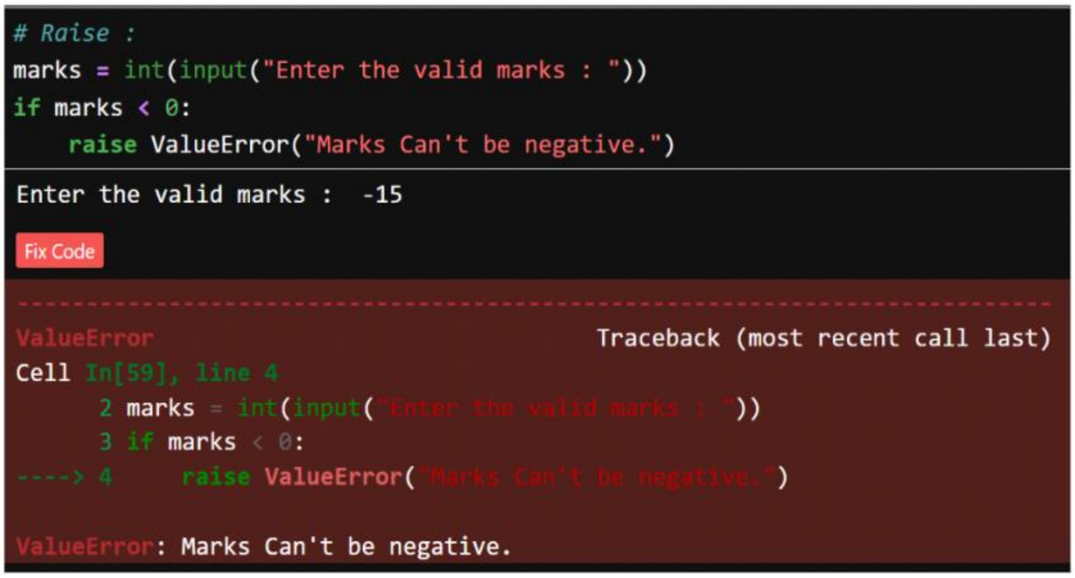
## raise Keyword

### What is raise?

The raise keyword lets you intentionally trigger an exception.

Syntax:

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

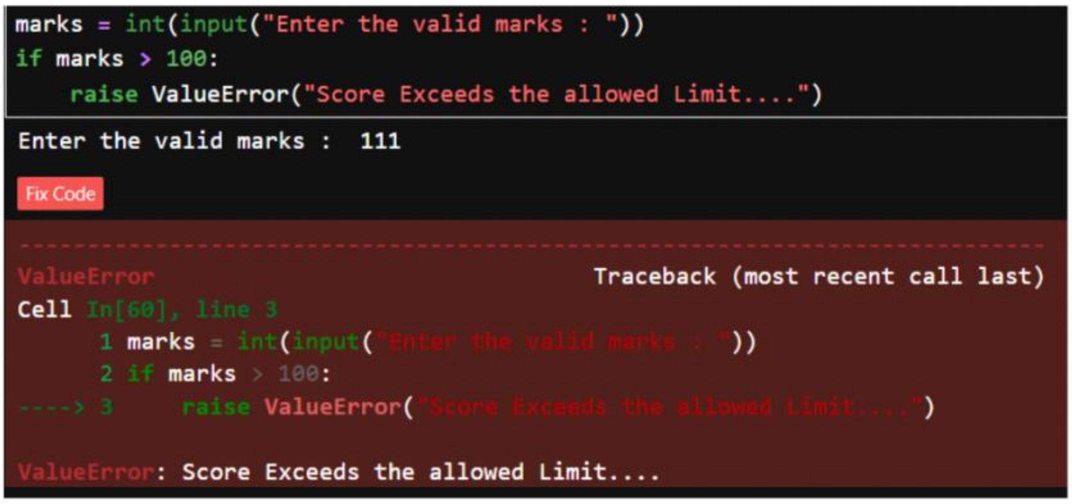


```python
# Raise :
marks = int(input("Enter the valid marks : "))
if marks < 0:
    raise ValueError("Marks Can't be negative.")
```

```
Enter the valid marks :  -15
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[59], 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.
```

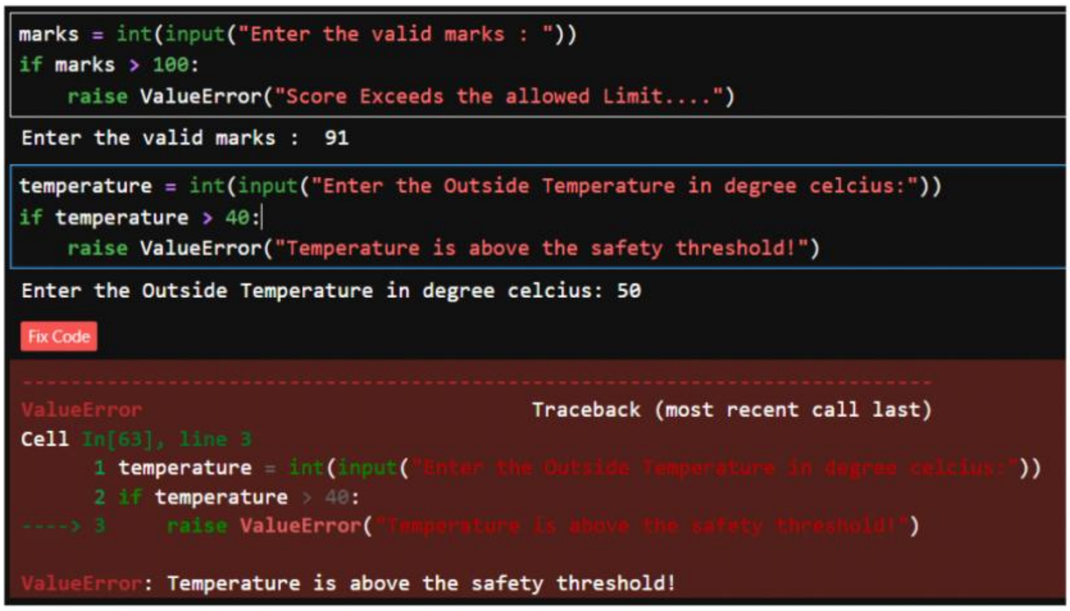


```
marks = int(input("Enter the valid marks : "))
if marks > 100:
    raise ValueError("Score Exceeds the allowed Limit....")
```

```
Enter the valid marks :  111
```

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

ValueError: Score Exceeds the allowed Limit....
```



```
marks = int(input("Enter the valid marks : "))
if marks > 100:
    raise ValueError("Score Exceeds the allowed Limit....")
```

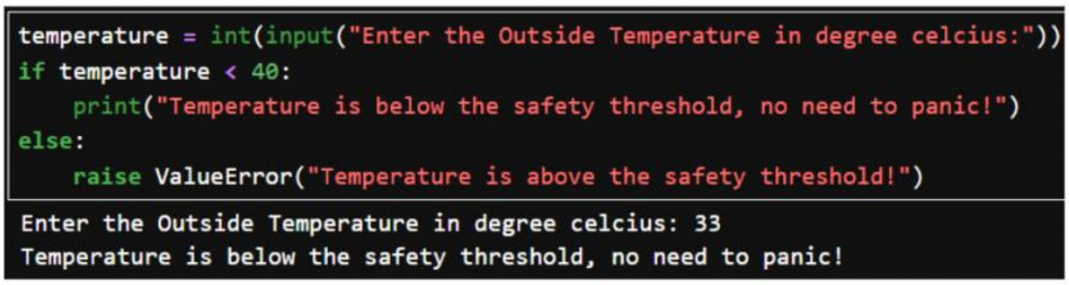
```
Enter the valid marks :  91
```

```
temperature = int(input("Enter the Outside Temperature in degree celcius:"))
if temperature > 40:
    raise ValueError("Temperature is above the safety threshold!")
```

```
Enter the Outside Temperature in degree celcius: 50
```

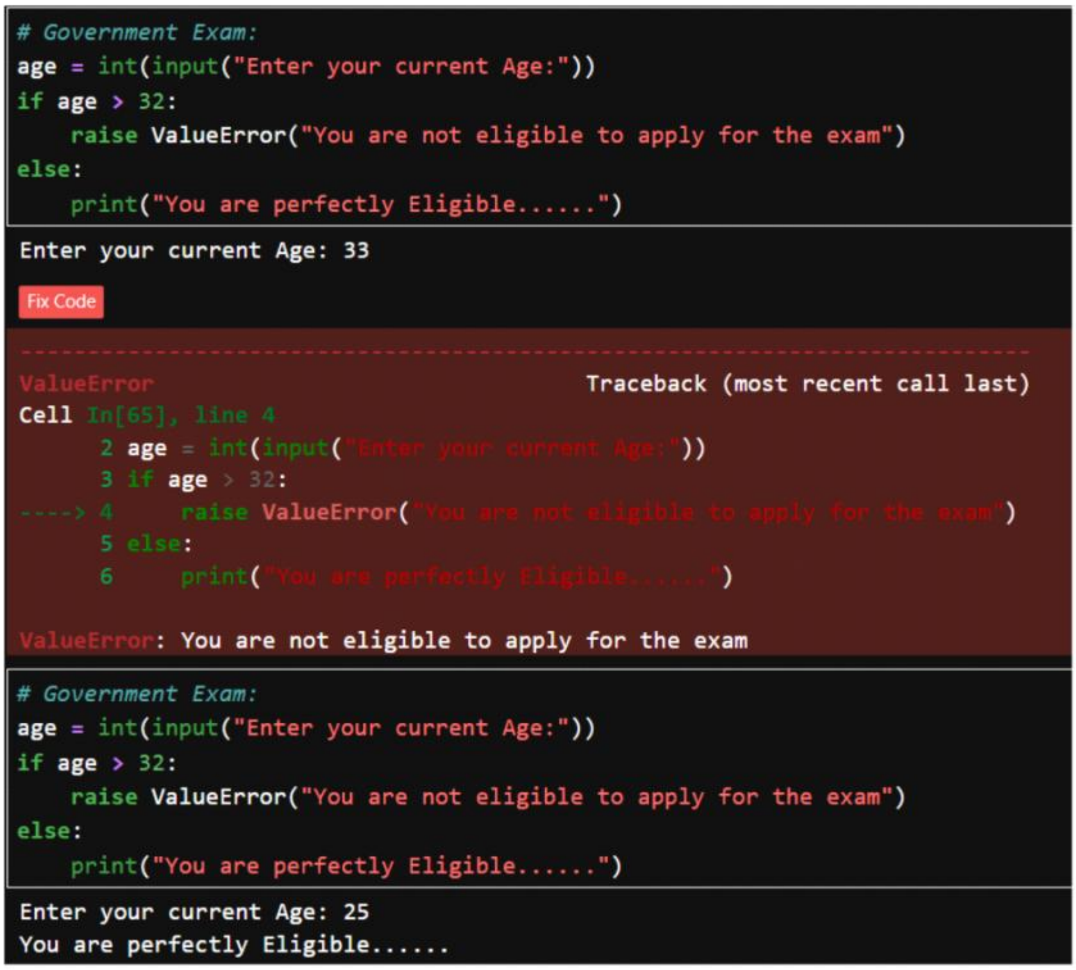
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[63], line 3
      1 temperature = int(input("Enter the Outside Temperature in degree celcius:"))
      2 if temperature > 40:
----> 3   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

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



```
temperature = int(input("Enter the Outside Temperature in degree celcius:"))
if temperature < 40:
    print("Temperature is below the safety threshold, no need to panic!")
else:
    raise ValueError("Temperature is above the safety threshold!")
```

```
Enter the Outside Temperature in degree celcius: 33
Temperature is below the safety threshold, no need to panic!
```



```python
# Government Exam:
age = int(input("Enter your current Age:"))
if age > 32:
    raise ValueError("You are not eligible to apply for the exam")
else:
    print("You are perfectly Eligible......")
```

```
Enter your current Age: 33
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[65], line 4
      2 age = int(input("Enter your current Age:"))
      3 if age > 32:
----> 4     raise ValueError("You are not eligible to apply for the exam")
      5 else:
      6     print("You are perfectly Eligible......")

ValueError: You are not eligible to apply for the exam
```

```python
# Government Exam:
age = int(input("Enter your current Age:"))
if age > 32:
    raise ValueError("You are not eligible to apply for the exam")
else:
    print("You are perfectly Eligible......")
```

```
Enter your current Age: 25
You are perfectly Eligible......
```

```
You are given a nested dictionary of users like this:
users = {
    "101": {"name": "Alice", "age": "23"},
    "102": {"name": "Bob", "age": "twenty"},
}
Write a program that:
Asks for a user ID.
Tries to fetch the user’s details.
Inside that, tries to convert their age into an integer.
Handle the following:
User ID not found.
Age not convertible to integer.
Any other unexpected error.
```