# Python Exception - Control Flow

# Agenda

- Python Exception - Control flow
  - With try / except
  - With try / except / else
  - With try /except/ finally
  - With try/except / else / finally

# Python Exception - Control flow

- When a ***try*** block comes across *break*, *continue* or *return* statement
  - the *else* clause does not execute
  - the *finally* clause however, executes whether an exception occurs or not
- When both ***try*** and ***finally*** block have a ***return*** statement, the returned value ***finally*** block takes precedence

# Python Exception - Control flow

```
def func_with_no_exception_handling():
    print("In function call")
    print(some_value)

print("Before function call")
func_with_no_exception_handling()
print("After function call")
```

- Above code when executed will not print the statement "*After function call*"
- Exception at *print(some_value)* will stop the program abruptly

# Python Exception - Control flow

```
import sys
def func_with_specific_exception_handler():
    try:
        print(xval)
    except NameError as e:
        print(e)
        print(str(sys.exc_info()))

print("Before function call")
func_with_specific_exception_handler()
print("After function call")
```

- Above code when executed will print the statement "*After function call*"
- Exception has been handled and consumed by the *except* block

# Python Exception - Control flow

- Using ***else*** in try / except clause. Remember, *else* block executes when no exception occurs

```
def divide_with_else_flow(num1, num2):
    result = None
    try:
        result = num1 / num2
    except (ZeroDivisionError, TypeError) as e:
        print(str(e), "\n", str(sys.exc_info()), "\n")
    else:
        print("In the else part")
    return result

divide_with_else_flow(5,5)
divide_with_else_flow(5,0)
```

# Python Exception - Control flow

- During the function *divide_with_else_flow(5, 5)* execution, when no exception is raised,
  - *else* block of the code executes
  - function returns the result (1.0)
- During the function *divide_with_else_flow(5, 0)* execution, when an exception is raised,
  - *except* block of the code executes
  - *else* block of the code will not execute
  - function returns None

# Python Exception - Control flow

```
def divide_with_else_flow_and_try_returns(num1, num2):
    try:
        return num1 / num2
    except (ZeroDivisionError, TypeError) as e:
        print(str(e), "\n", str(sys.exc_info()), "\n")
    else:
        print("In the else part")

divide_with_else_flow_and_try_returns(5,5)
divide_with_else_flow_and_try_returns(5,0)
```

# Python Exception - Control flow

- During the function *divide_with_else_flow_and_try_returns(5, 5)* execution, when no exception is raised,
  - *else* block of the code does not executes due to return statement in try block
  - function returns the result (1.0)
- During the function *divide_with_else_flow_and_try_returns(5, 0)* execution, when an exception is raised,
  - *except* block of the code executes
  - *else* block of the code will not execute
  - function returns None

# Python Exception - Control flow

```
def divide_with_else_flow_and_else_returns(num1, num2):
    result = None
    try:
        result = num1 / num2
    except (ZeroDivisionError, TypeError) as e:
        print(str(e), "\n", str(sys.exc_info()), "\n")
    else:
        print("In the else part")
        return 20
    return result
print(divide_with_else_flow_and_else_returns(5, 5))
print(divide_with_else_flow_and_else_returns(5, 0))
```

# Python Exception - Control flow

- During the function *divide_with_else_flow_and_else_returns(5, 5)* execution, when no exception is raised,
  - *else* block of the code executes
  - function returns 20
- During the function *divide_with_else_flow_and_else_returns(5, 0)* execution, when an exception is raised,
  - *except* block of the code executes
  - *else* block of the code will not execute
  - function returns none
- If a function with *try / except / else* block has to return a value, the return statement has to be the last statement in the function

# Python Exception - Control flow

- Using ***finally*** in try / except clause. Remember, *finally* block always executes

```
def divide_with_finally_flow(num1, num2):
    result = None
    try:
        result = num1 / num2
    except (ZeroDivisionError, TypeError) as e:
        print(str(e), "\n", str(sys.exc_info()), "\n")
    finally:
        print("In the finally part")
    return result

print(divide_with_finally_flow(5,5))
print(divide_with_finally_flow(5,0))
```

# Python Exception - Control flow

- During the function *divide_with_finally_flow(5, 5)* execution, when no exception is raised,
  - *finally* block of the code executes
  - function returns the result (1.0)
- During the function *divide_with_finally_flow(5, 0)* execution, when an exception is raised,
  - *except* block of the code executes
  - *finally* block of the code executes
  - Function returns none

# Python Exception - Control flow

```
def divide_with_finally_flow_and_try_returns(num1, num2):
    try:
        return num1 / num2

    except (ZeroDivisionError, TypeError) as e:
        print(str(e), "\n", str(sys.exc_info()), "\n")

    finally:
        print("In the finally part")

print(divide_with_finally_flow_and_try_returns(5,5))
print(divide_with_finally_flow_and_try_returns(5,0))
```

# Python Exception - Control flow

- During the function *divide_with_finally_flow_and_try_returns(5, 5)* execution, when no exception is raised,
  - *finally* block of the code executes
  - function returns the result (1.0)
- During the function *divide_with_finally_flow_and_try_returns(5, 0)* execution, when an exception is raised,
  - *except* block of the code executes
  - *finally* block of the code executes
  - Function returns none
- Note: the return statement of *finally* takes precedence over other return statements in the function. Refer example in next slide

# Python Exception - Control flow

```
def divide_with_finally_flow_and_both_try_and_finally_returns(num1, num2):
    try:
        return num1 / num2

    except (ZeroDivisionError, TypeError) as e:
        print(str(e), "\n", str(sys.exc_info()), "\n")

    finally:
        print("In the finally part")
        return 5

print(divide_with_finally_flow_and_both_try_and_finally_returns(5,5)) # returns 5
print(divide_with_finally_flow_and_both_try_and_finally_returns(5,0)) # returns 5
```

# Python Exception - Control flow

```
def divide_with_else_and_finally_flow(num1, num2):
    result = None
    try:
        result = num1 / num2
    except (ZeroDivisionError, TypeError) as e:
        print(str(e), "\n", str(sys.exc_info()), "\n")
    else:
        print("In the else part")
    finally:
        print("In the finally part")
    return result

print(divide_with_else_and_finally_flow(5,5))
print(divide_with_else_and_finally_flow(5,0))
```

# Python Exception - Control flow

- During the function *divide_with_else_and_finally_flow(5, 5)* execution, when no exception is raised,
  - *else* block of the code executes
  - *finally* block of the code executes
  - function returns the result (1.0)
- During the function *divide_with_else_and_finally_flow(5, 0)* execution, when an exception is raised,
  - *except* block of the code executes
  - *finally* block of the code executes
  - Function returns none
- Note: the return statement of *finally* takes precedence over other return statements in the function. Refer example in next slide

# Summary

- We learned about the control flow for python exceptions handling using try/ except / else / finally

# Thank You