

## Լրացուցիչ խնդիրներ(Decorators, Exceptions)

1. Ստեղծել `safe_execute` դեկորատոր, որը.

- Կկատարի ֆունկցիան `try/except` բլոկում
- Եթե առաջանա որևէ Exception,
  - Տպի սխալի հաղորդագրություն
  - Վերադարձնի `"Error occurred"`

```
@safe_execute
def divide(a, b):
    return a / b

print(divide(10, 2)) # 5.0
print(divide(10, 0)) # Error occurred
```

2. Ստեղծել `type_check(expected_type)` դեկորատոր, որը.

- Ստուգում է ֆունկցիայի բոլոր արգումենտների տիպը
- Եթե որևէ արգումենտ սխալ տիպի է → գում է `TypeError`

```
@type_check(int)
def add(a, b):
    return a + b

print(add(3, 5)) # 8
print(add(3, "5")) # TypeError
```

3. Իրականացնել `handle_specific(exception_type)` դեկորատոր, որը.

- Բռնում է միայն նշված exception-ը
- Տպում է `"Handled <ExceptionName>"`
- Այլ exception-ները չի բռնում

```
@handle_specific(ZeroDivisionError)
def divide(a, b):
    return a / b

divide(10, 0) # Handled ZeroDivisionError
```

4. Ստեղծել `log_errors_to_file` դեկորատոր, որը.

- Եթե առաջանա exception,
- Կգրի սխալի տեքստը `errors.txt` ֆայլի մեջ
- Հետո կգցի exception-ը

```
@log_errors_to_file
```

```
def divide(a, b):  
    return a / b
```

```
divide(10, 0)
```

5. Ստեղծել `validate_range` դեկորատոր, որը.

- Ստուգում է, որ ֆունկցիայի վերադարձվող թիվը լինի `[min_val, max_val]` միջակայքում
- Եթե ոչ → գցում է `ValueError`

```
@validate_range(0, 100)
```

```
def get_number():  
    return 150
```

```
print(get_number()) # ValueError
```

6. Ստեղծել `require_positive` դեկորատոր, որը.

- Ստուգում է, որ բոլոր թվային արգումենտները լինեն դրական
- Եթե կա բացասական թիվ → գցում է `ValueError("Negative value not allowed")`

```
@require_positive
```

```
def square_root(x):  
    return x ** 0.5
```

```
print(square_root(9)) # 3.0
```

```
print(square_root(-4)) # ValueError
```

7. Ստեղծել `timeout_simulator` դեկորատոր, որը.

- Կչափի ֆունկցիայի կատարման ժամանակը
- Եթե այն գերազանցում է օրինակ 2 վայրկյանը → գցում է `TimeoutError`

```
@timeout_simulator
def slow_function():
    time.sleep(3)
    return "Done"

slow_function() # TimeoutError
```

8. Ստեղծել `validate_non_empty` դեկորատոր, որը.

- Եթե ֆունկցիայի վերադարձը դատարկ տող է կամ դատարկ list → գցում է `ValueError("Empty result")`

```
@validate_non_empty
def get_list():
    return []

get_list() # ValueError
```

9. Գրել ֆունկցիա, որի վրա միաժամանակ կիրառված են հետևյալ դեկորատորները`

- `@safe_execute`
- `@type_check(int)`
- `@require_positive`

Փորձարկել տարբեր դեպքերով`

- սխալ տիպ
- բացասական թիվ
- ճիշտ արգումենտ