
DIGGIBYTE
CREATING VALUE WITH DATA

DIGGIBYTE
CREATING VALUE WITH DATA

Python

DIGGIBYTE
CREATING VALUE WITH DATA

Agenda

- Higher Order Functions
- Error Handling

DIGGIBYTE
CREATING VALUE WITH DATA

Higher Order Functions

- Functions that take other functions as arguments or return functions
- Used to achieve code reusability and abstraction
- Allow for functional programming style in Python

```
def apply_twice(func, arg):
    return func(func(arg))


1 usage
def add_five(x):
    return x + 5

print(apply_twice(add_five,  arg: 10))
```

DIGGIBYTE
CREATING VALUE WITH DATA

Higher Order Functions

- Built-in higher-order functions in Python:
  - map(): Applies a function to each item in an iterable
  - filter(): Creates a new iterable with elements that satisfy a condition
  - reduce(): Applies a function of two arguments cumulatively to the elements of a sequence

DIGGIBYTE
CREATING VALUE WITH DATA

Higher Order Functions

```
numbers = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x**2, numbers))
print(squared)

even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)

from functools import reduce
sum_of_numbers = reduce(lambda x, y: x + y, numbers)
print(sum_of_numbers)
```

DIGGIBYTE
CREATING VALUE WITH DATA

Higher Order Functions

- Decorators:
  - Functions that modify or extend the behavior of other functions
  - Useful for adding functionality to existing functions without modifying their source code

```
def uppercase(func):
    def wrapper(arg):
        result = func(arg)
        return result.upper()
    return wrapper
1 usage
@uppercase
def greet(name):
    return f"Good Morning, {name}"
print(greet("Amit"))  # Output: "GOOD MORNING, AMIT"
```

DIGGIBYTE
CREATING VALUE WITH DATA

Error Handling

- Exceptions and try-except blocks
- Raising custom exceptions
- finally and else clauses

```
try:
    result = 10 / 0
except ZeroDivisionError as e:
    print(f"Error: {e}")
else:
    print(f"Result: {result}")
finally:
    print("This will always be executed")
```

DIGGIBYTE
CREATING VALUE WITH DATA

Error Handling

- Handling multiple exceptions

- Exception hierarchy and catch-all exception handler

```
file_path = "data_unavailable_path.csv"
try:
    with open(file_path, "r") as file:
        data = file.read()
except FileNotFoundError:
    print(f"Error: File '{file_path}' not found")
except PermissionError:
    print(f"Error: Permission denied to access '{file_path}'")
except Exception as e:
    print(f"Error: {e}")
else:
    pass
```

DIGGIBYTE
CREATING VALUE WITH DATA

Error Handling

- Logging exceptions

- Provides a way to track and record errors for debugging and monitoring purposes

```
import logging
logging.basicConfig(filename='error.log',level = logging.ERROR)

try:
    result = 10 / 0
except ZeroDivisionError as e:
    logging.error( msg: f"Error: {e}", exc_info=True)
```

DIGGIBYTE
CREATING VALUE WITH DATA

Error Handling

error.log

```
1  ERROR:root:Error: division by zero
2  Traceback (most recent call last):
3      result = 10 / 0
4  ZeroDivisionError: division by zero
```

DIGGIBYTE

CREATING VALUE WITH DATA

THANK YOU !!