Raising Exceptions
Assertions

Raising Exceptions

Definition:

- Raising an exception means explicitly triggering an error in your program using the raise statement.

Purpose:

- To signal that an **unexpected condition** or **error** has occurred, so the normal flow of execution should be interrupted or redirected.

Why Raise Exceptions?

1. Enforce valid inputs or states
2. Protect code from incorrect usage
3. Improve reliability by catching issues early
4. Enable custom error-handling logic

Syntax

Purpose: Catch and handle a specific type of error.

syntax

```
raise ExceptionType("Optional error message")
```

- **ExceptionType**: any class inheriting from BaseException (typically Exception or a subclass).
- **Message**: a human-readable string, accessible via the exception's .args or str(e).

Example

python

```python
age = -5
if age < 0:
    raise ValueError("Age cannot be negative")
```

Output

```
Age cannot be negative
```

Assertions

- An **assertion** is a sanity check you put in code: “**this must be true here; otherwise, something is wrong with my program**.”
- ”If the condition is **False**, Python raises **AssertionError** (optionally with your custom message).
- Assertions are meant for **developer errors / internal invariants**, not for **validating user input** or **runtime environment issues**.

Syntax

syntax

```
assert condition, "Optional error message"
```

- **condition** - A boolean expression that should be true.
- **message (optional)** - An optional message to be displayed if the assertion fails.

Example

python

```python
def sqrt_non_negative(x):
    assert x >= 0, "x must be non-negative"
    return x ** 0.5

print(sqrt_non_negative(9))
print(sqrt_non_negative(-1))
```

Output

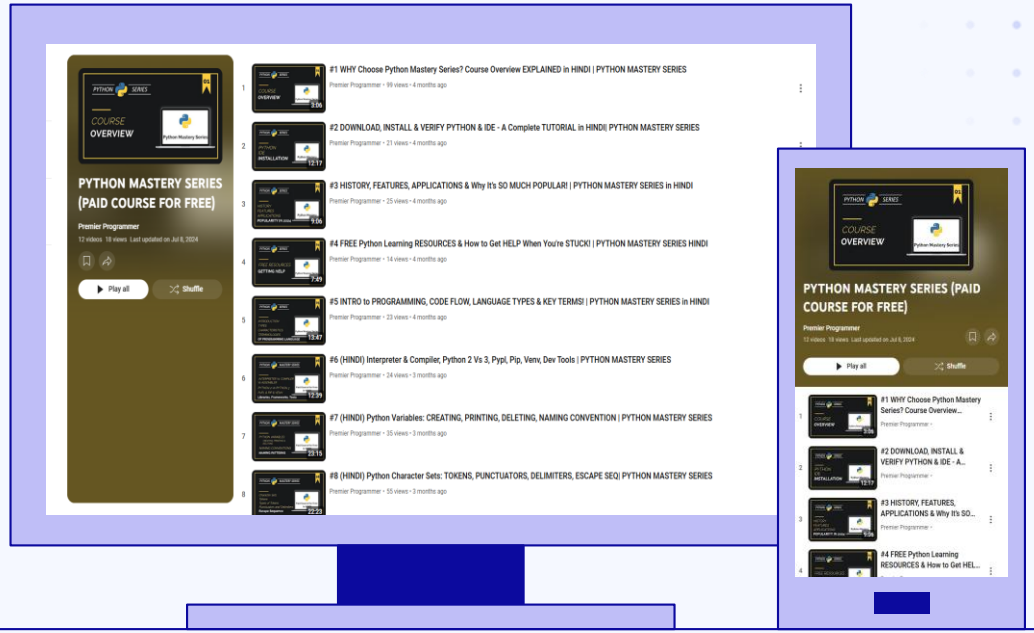
```
3.0
AssertionError in debug mode
```

Practice Set

Download Link in
Description
and
Pinned Comment

WATCH

Level up your coding with each episode in this focused Python series.



Next Video!

Practice Set Solution