// DEF — define a function

```
def greet(name, role="developer"):
    print(f"Hello, {name}. You are a {role}.")

greet("Bull")              # uses default role
greet("Horn", "student")   # overrides default
```

Define before calling. Parentheses required.

// RETURN — send a value back

```
def calculate_bmi(weight, height):
    return weight / height ** 2

bmi = calculate_bmi(70, 1.75)
print(f"{bmi:.1f}")    # 22.9

# no return — function returns None
# return stops the function immediately
```

print inside != return. return sends value out.

// MULTIPLE RETURN VALUES

```
def first_and_last(word):
    return word[0], word[-1]

first, last = first_and_last("RedHorn")
print(first, last)    # R n
```

return a, b — unpack with x, y = function(). Order matters.

// SCOPE

```
name = "Bull"         # global — readable everywhere

def greet():
    message = "Hello."    # local — gone when done
    print(name)           # can read global

# print(message)    # NameError outside
```

Local lives and dies with the function. Use return, not globals.

// TRY / EXCEPT / ELSE / FINALLY

```
try:
    age = int(input("Enter age: "))
except ValueError:
    print("Numbers only.")
else:
    print(f"Age: {age}.")   # try succeeded
finally:
    print("Done.")          # always runs
```

else = success. finally = always. Use only what you need.

// RAISE — throw your own errors

```
def get_input(prompt):
    while True:
        try:
            value = float(input(prompt))
            if value <= 0:
                raise ValueError("Must be > 0.")
            return value
        except ValueError as e:
            print(f"Invalid: {e}")
```

raise enforces your rules. as e gives access to the message.

// COMMON MISTAKES

```
! Define before calling — top to bottom
! Parentheses required — greet() not greet
! print != return — print displays, return sends
! return stops function — nothing after it runs
! Default parameters always last
! Use parameters and return — avoid globals
```