# match / case — The Full Picture

## // MATCH / CASE — the basics

```
match command:
    case "quit":
        print("Quitting.")
    case "help":
        print("Help.")
    case "start" | "run":  # | combines values
        print("Starting.")
    case _:
        print("Unknown.")  # wildcard — always last
```

One variable. Many cases. First match wins.

## // GUARDS — conditions inside cases

```
match score:
    case n if n >= 90:
        print("Excellent.")
    case n if n >= 70:
        print("Good.")
    case n if n >= 60:
        print("Passed.")
    case _:
        print("Failed.")
```

n captures the value. Guard adds condition. Both must be True.

## // NORMALIZE BEFORE MATCHING

```
# Wrong — misses 'Quit', 'QUIT', etc.
match command:
    case "quit":
        print("Quitting.")

# Right — normalize first
match command.lower():
    case "quit":
        print("Quitting.")
```

match is case-sensitive. Always .lower() before matching strings.

## // MATCH VS IF / ELIF

```
# match — one variable, many values
match product:
    case "water":   price = 1.00
    case "coffee":  price = 2.00
    case _:         price = None

# if / elif — complex, multi-variable logic
if score > 90 and attendance > 80:
    print("Honours.")
```

match for one variable — if/elif for complex logic.

## // VENDING MACHINE PATTERN

```
match product:
    case "water":      price = 1.00
    case "coffee":     price = 2.00
    case "chips":      price = 1.50
    case _:
        print("Not found.")
        price = None

if price is not None:
    if amount >= price:
        change = amount - price
        print(f"Change: {change:.2f} EUR")
```

match sets the value — if handles the logic after.

## // COMMON MISTAKES

```
! match requires Python 3.10+
! Always add case _ — unknown values silently ignored
! Normalize with .lower() — match is case-sensitive
! case _ always goes last — matches everything
! match for one variable — if/elif for complex logic
```