1.2 Selection – choosing between paths

Name: _____ Class: _____ Date: _____ **Total: 20 marks**

KEY WORDS selection • boolean • condition

Objective

Write selection with `if/elif/else` whose branch order is deliberate, build boolean conditions with `and`/`or`/`not`, and avoid the `==` versus `=` and floating-point comparison traps.

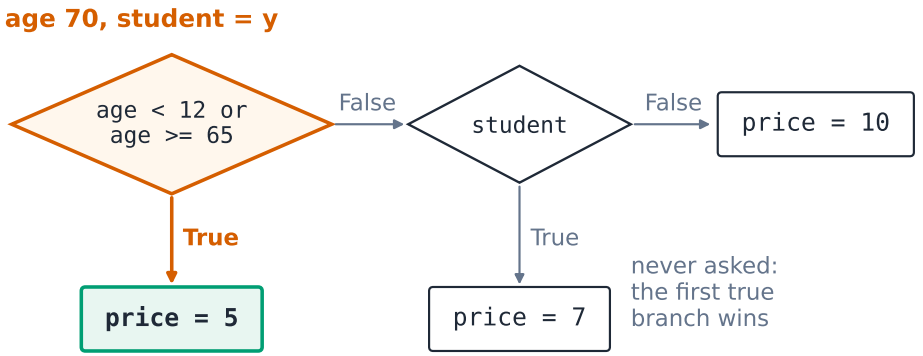
1 Key knowledge and worked examples

The first true branch wins

Selection runs a branch only when its condition holds. Conditions are **boolean** expressions built from comparisons (`==`, `!=`, `<`, `>=`) and logic (`and`, `or`, `not`). Python tests branches top to bottom and runs the FIRST true one – so the order is a design decision, not a detail.

```
if temp >= 38:
    print("Fever")
elif temp >= 36:
    print("Normal")
else:
    print("Low")
```

Indentation is the block: everything indented under a branch belongs to it.



Worked example: branch order decides the price

Cinema pricing: under 12 or 65+ pay 5, students pay 7, others 10.

```
age = int(input("Age? "))
student = input("Student? y/n ") == "y"
if age < 12 or age >= 65:
    price = 5
elif student:
    price = 7
else:
    price = 10
print("Pay", price)
```

A 70-year-old student hits the first branch and pays 5 – the ordering says so. Swap the branches and the same person pays 7: same conditions, different program. `or` joins the age group; `== "y"` turns the answer into a boolean, no `if` needed.

Two classic traps

- `==` compares, `=` assigns. `if x = 5:` is a `SyntaxError` – Python catches it for you; the dangerous version is assigning when you meant to compare inside an expression elsewhere.
- Binary floats cannot store 0.1 exactly: `0.1 + 0.2 == 0.3` is `False`. Compare with a tolerance: `abs(0.1 + 0.2 - 0.3) < 1e-9` is `True`.

2 Practice

2.1 Write an `if/elif/else` that prints `Pass` for a mark of 50 or more, `Resit` for 40–49, and `Fail` below 40. State which branch a mark of exactly 50 takes. [3]

2.2 Give the value of each: `3 < 5 and 5 < 4`, `3 < 5 or 5 < 4`, `not (3 == 3)`. [3]

2.3 A door opens if the badge is valid AND it is not a weekend. Write the boolean expression, and its value for a valid badge on Saturday. [3]

3 Exam-style questions

3.1 A delivery app charges: free over 50; otherwise 4.95, unless the customer has membership, in which case free over 25 and 2.95 below.

Write the complete pricing `if/elif/else` (inputs: `total`, `member` as a boolean). Justify your branch order in one sentence, give the charge for total 30 with membership, and for total 55 without. [6]

3.2 Explain, with a worked example each, (a) why `0.1 + 0.2 == 0.3` evaluates to `False`; (b) how the tolerance comparison fixes it; (c) why rounding both sides to one decimal would also work here but is the weaker habit. [5]

Solutions

2.1

- `if mark >= 50: print("Pass") / elif mark >= 40: print("Resit") / else: print("Fail")`.
- Exactly 50 takes `Pass` – `>=` includes the boundary.

2.2

- `False` (and needs both).
- `True` (or needs one).
- `False` (3 == 3 is True; not flips it).

2.3

- `valid and not weekend` (equivalently `valid and weekend == False`).
- Valid badge, Saturday → `True and not True` → `False`: the door stays locked.

3.1

```
if member and total > 25:
    charge = 0
elif total > 50:
    charge = 0
elif member:
    charge = 2.95
else:
    charge = 4.95
```

- Member-free branch first (a member over 50 must also be free – most generous condition first prevents it falling into the wrong bucket).
- Non-member free over 50.
- Member below 25.
- Non-member default.
- Total 30 with membership → `0`.
- Total 55 without → `0`.

3.2

- (a) 0.1 and 0.2 have no exact binary form; their stored values sum to slightly more than 0.3's stored value, so `==` is `False`.
- (b) `abs(0.1 + 0.2 - 0.3) < 1e-9` asks "close enough?" instead of "exactly equal?" – `True`.
- (c) Rounding to one decimal works for these particular numbers, but picks an arbitrary precision per comparison; the tolerance states the acceptable error once and works at any scale.