1.1 Values, types and input/output

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

KEY WORDS value • type • string • variable • float

Objective

Use the four core types with their operations, convert between them explicitly, read input and format output – and predict the type of every expression before running it.

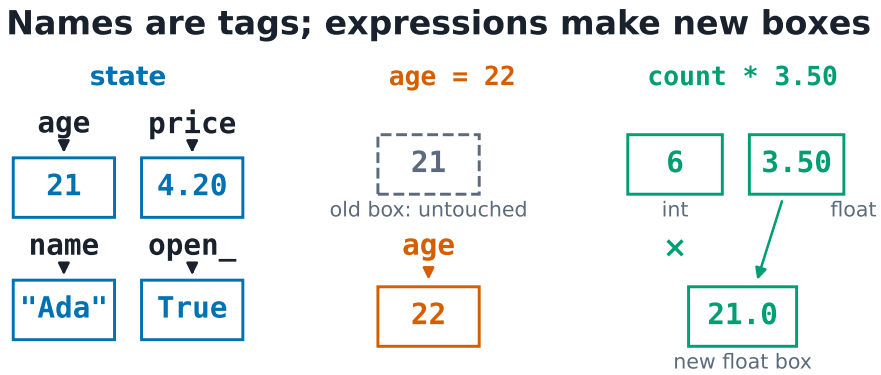
1 Key knowledge and worked examples

The type decides what you can do

A **value** has a **type**: `int`, `float`, `str`, `bool`. Operations follow the types: `"4" + "2"` is `"42"` (string joining), `4 + 2` is 6, `"4" + 2` is a `TypeError` – Python never guesses a conversion.

```
age = 21
print(type(age))        # <class 'int'>
name = "Ada"
print(name * 2)         # AdaAda - str * int repeats
```

Variables are names bound to values; `age = 22` rebinds the name to a new value, it does not change the old one.



Worked example: the café till

```
count = int(input("Coffees? "))
print("Total:", count * 3.50)
```

With 6 entered, this prints `Total: 21.0`. The `.0` is the tell: `int * float` promotes to **float**. With `count * 350` (pence) it would print an `int` with no decimal point. Which type comes out of an expression is unit 1.1's whole habit – predict it first, then run.

`input` always returns a **string**: `int("21")` converts, and `int("21.0")` raises `ValueError` (use `float` first, then `int`, for such strings).

f-strings: formatted output

```
name = "Ada"
balance = 12.5
print(f"{name} has £{balance:.2f}")   # Ada has £12.50
```

The `f` prefix turns `{...}` into slots; `:.2f` fixes two decimals. Formatting never changes the value – `balance` is still `12.5`.

2 Practice

2.1 Give the type and value of each: `7 / 2`, `7 // 2`, `7 % 2`, `2 ** 3`. [4]

2.2 Explain in one sentence each why `"3" + "4"` and `3 + 4` give different results, and why `"3" * 4` works but `"3" + 4` raises an error. [3]

2.3 Write two lines that read a price as a decimal number from the user and print it rounded to whole currency units, stating what happens if the user types `seven`. [3]

3 Exam-style questions

3.1 A membership system reads a birth year and today's year, both typed by the user, then prints the member's age and whether they are an adult (18+).

Write the program (about five lines) using `input`, explicit conversion, and one comparison. State the exact output for birth year 2000 with today 2026, and name the exception raised if the user enters `oops`. [5]

3.2 A receipt line must print $3 \times$ £4.20 = £12.60 from `count = 3` and `unit = 4.2`.

Write the f-string that produces exactly that line, with the total always at two decimals. Then state why `count * unit` gives `12.600000000000001` on some systems and what your formatting does about it. [5]