# 6.1 Data types, strings, exceptions and trace tables

Name: _____ Class: _____ Date: _____ **Total: 29 marks**

**KEY WORDS** data type • casting • syntax error • runtime error • logic error • trace table

## Objective

Build the skills to answer exam questions on **programming fundamentals** – data types, strings, exceptions and trace tables, in Python.

**You must be able to:**

- choose a suitable **data type** for a value, and **cast** between types
- use string indexing, slicing and methods, and state what they return
- tell a **syntax error** from a **runtime error** and a **logic error**
- handle invalid input with `try … except`
- complete a **trace table** for code with a loop and selection

## 1 Key knowledge and worked examples

Study these first. Each one shows the method for a question type used later.

■ **Key knowledge: types and strings**

| Type | Python | Example |
|---|---|---|
| integer | `int` | `42` |
| real | `float` | `3.75` |
| Boolean | `bool` | `True` |
| string | `str` | `"Ada"` |

- `input()` always returns a string: `int(input())` casts it to an integer.
- `7 / 2` is `3.5`; `7 // 2` is `3`; `7 % 2` is `1`.

For `s = "network"` (indexes 0 to 6):

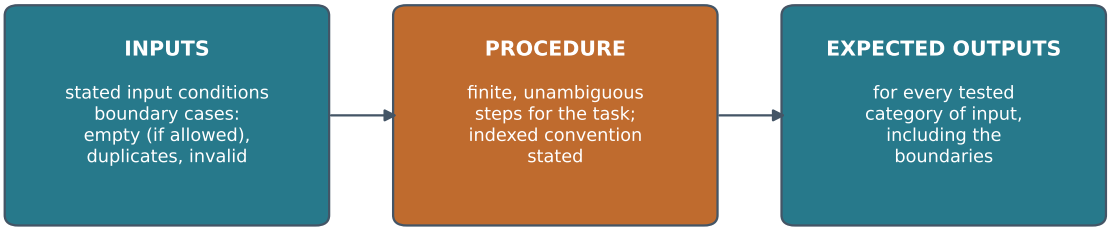
| Expression | Value | Rule |
|---|---|---|
| `len(s)` | `7` | number of characters |
| `s[0]` | `"n"` | the first index is 0 |
| `s[3:7]` | `"work"` | from 3 up to, not including, 7 |
| `s.find("w")` | `3` | first index, or -1 if absent |
| `s.upper()` | `"NETWORK"` | a new string; `s` is unchanged |

## Key knowledge: three kinds of error

| Kind | When it shows | Example |
|---|---|---|
| syntax | before the program runs | `if x > 0` with no colon |
| runtime | while it runs: an **exception** | `int("two")` raises `ValueError` |
| logic | never reported: wrong output | `<` where `<=` was meant |

## Worked example: handling bad input

A program contract: inputs, procedure, expected outputs




A failure is diagnosed as a wrong algorithm, a wrong implementation, or an incomplete requirement.

A program asks for an age from 0 to 120 and must not crash.

```
while True:
    try:
        age = int(input("Age: "))
        if 0 <= age <= 120:
            break
        print("Out of range.")
    except ValueError:
        print("Enter a whole number.")
```

1. `int("abc")` raises `ValueError`: the `except` block prints a message and the loop repeats.
2. `int("150")` succeeds, but 150 fails the range check: the loop repeats.
3. `int("17")` passes both: `break` leaves the loop with `age` = 17.
4. Test data: normal (17), **boundary** (0 and 120), invalid (-1, 121, `"abc"`).

### ■ Worked example: a trace table

```
n = 13
steps = 0
while n != 1:
    if n % 2 == 0:
        n = n // 2
    else:
        n = n - 1
    steps = steps + 1
print(steps)
```

| n | n % 2 == 0 | steps |
|---|---|---|
| 13 | | 0 |
| 12 | False | 1 |
| 6 | True | 2 |
| 3 | True | 3 |
| 2 | False | 4 |
| 1 | True | 5 |

Output: **5**. Write one row each time a variable changes; stop when the `while` condition is false.

## 2 Practice

**2.1** State the most suitable data type for: (i) the number of seats in a cinema; (ii) the price of a ticket, 8.50; (iii) whether a seat is booked; (iv) a booking reference such as `"K7-204"`. [4]

**2.2** Evaluate: (i) `17 // 5`; (ii) `17 % 5`; (iii) `17 / 5`. [3]

**2.3** `code = "IB2027CS"`. State: (i) `len(code)`; (ii) `code[2:6]`; (iii) `code.find("C")`; (iv) `code[-1]`. [4]

**2.4** Classify each error as syntax, runtime or logic. [3]

(i) `print("Total:", total` with no closing bracket

(ii) an average calculated as `total / 10` when there are 12 items

(iii) `marks[5]` when `marks` has 5 items

## 3 Exam-style questions

**3.1** What is printed by `print("3" + "4")`? [1]

- **A** `7`
- **B** `34`
- **C** `"3" + "4"`
- **D** an error message

**3.2** `word = "python"`. Which expression gives `"tho"`? [1]

- **A** `word[2:4]`
- **B** `word[2:5]`
- **C** `word[3:5]`
- **D** `word[3:6]`

**3.3** Which input makes `int(input())` raise a `ValueError`? [1]

- **A** `42`
- **B** `-7`
- **C** `007`
- **D** `4.5`

**3.4** A program reads temperatures until the user enters `x`.

```
count = 0
total = 0
high = -100
t = input()
while t != "x":
    v = int(t)
    count = count + 1
    total = total + v
    if v > high:
        high = v
    t = input()
print(total / count, high)
```

(a) Complete a trace table for the inputs `18`, `23`, `19`, `x`, with columns `t`, `count`, `total` and `high`, and state the output. [4]

(b) State the output if the first input is `x`, and explain why. [2]

(c) Explain why the program crashes if the user types `twenty`, and describe how `try … except` would prevent it. [3]

(d) Give **three** test inputs for the temperature, one normal, one boundary and one invalid, for a sensor that reads from -40 to 50. [3]