Finger Exercises 1

Introduction to Programming & Objects

≈ 15 minutes

Comp 102 — Programming Fundamentals

What this is. A short warm-up to lock in what we just covered in class.
How to do it. On **paper**, **no computer**, in one sitting. If an item takes you more than a minute, skip it, finish the rest, then come back. The answer key is on the last page — do not peek until you are done.
Why bother. These are exactly the four moves your quizzes ask for: trace, fill in, fix, and answer a concept question.

Part A • Do You Remember?

RECALL ≈ 3 min

A1. The lecture said a computer is fast, has a huge memory, is accurate, and never gets tired — but it has two big limitations. A computer is _____ and it is _____.

A2. The only language a computer really understands is called _____ , and it is written using only the digits ____ and ____.

A3. Every object has three things. Name them:

1. _____ 2. _____ 3. _____

A4. Circle **True** or **False**:

(a) A computer can invent a brand-new way to solve a problem by itself. True / False

(b) Python code is turned into machine language before the computer runs it. True / False

(c) If a program shows no error message, its answer must be correct. True / False

(d) In Python, everything is an object. True / False

Part B • What Does It Print?

TRACE ≈ 4 min

Write the output exactly, one value per line, in order.

B1.

```
repeat 3:
    print(1)
print(2)
```

Output: _____

B2.

```
repeat 2:
    print('a')
    print('b')
```

Output: _____

B3. Same two lines, one of them moved. What changes?

```
repeat 2:
    print('a')
print('b')
```

Output: _____

Part C • Robot Commands

TRACE ≈ 3 min

Reminder: `forward()` moves one step, `left()` turns 90° left, and `repeat n:` repeats the indented lines `n` times.

C1. Rewrite this using `repeat`, so that `forward()` is written only **once**:

```
forward()
forward()
forward()
```

Your version:

_____

_____

C2. How many steps forward does the robot take in total?

```
repeat 3:
    forward()
    forward()
```

Total steps: _____

C3. The robot must reach the cup. Write the shortest program you can, using `repeat` once.

robot cup

_____

_____

Part D • Syntax, Semantics, or Neither?

CLASSIFY ≈ 2 min

Tick one column for each. Remember: **syntax** = the rules of writing it down; **semantics** = whether it has a sensible meaning.

Expression	Syntax error	Semantic error	Neither — it is fine
`2 5 +`	☐	☐	☐
`3 / 'hi'`	☐	☐	☐
`'hello' * 3`	☐	☐	☐
`print('hi'`	☐	☐	☐
`'5' + 5`	☐	☐	☐

Part E • Scalar or Non-scalar?

CLASSIFY ≈ 2 min

Write **S** for scalar (cannot be sub-divided) or **N** for non-scalar (can be sub-divided).

E1. `42` _____ **E4.** `3.14` _____

E2. `True` _____ **E5.** a list of your marks _____

E3. `'a'` _____ **E6.** a `Student` _____

Part F • One Sentence Each

EXPLAIN ≈ 1 min

F1. Why do we write programs in a high-level language like Python instead of writing 1s and 0s directly?

_____

F2. Your program runs, shows no error message, and prints an answer — but the answer is wrong. Is this a syntax error? Why or why not?

_____

Stuck on more than two items? Re-read the Lecture 1 summary slide before the next class — everything above came straight from it.

Answer Key

Check your work *after* attempting everything. If an answer surprises you, that item is worth re-reading in the slides.

A1. **stupid** (it does exactly what you tell it) and **not creative** (it cannot invent new ways to solve a problem).

A2. **machine language**; the digits `1` and `0`.

A3. a **type**; a set of **attributes** (the data it holds); a set of **operations** you can perform on it.

A4. (a) **False** — computers are not creative. (b) **True** — that is what an interpreter is for. (c) **False** — a program can run happily and still give the wrong answer. (d) **True**.

B1. `1 1 1 2` — the indented line repeats three times, then `print(2)` runs once because it is *not* indented.

B2. `a b a b` — *both* indented lines belong to the repeat, so the pair runs twice.

B3. `a a b` — indentation is the *only* thing that changed, and it changed the whole meaning. This is the single most important habit to build this week.

C1. `repeat 3:` then, indented, `forward()`.

C2. **6** — two steps per round, three rounds.

C3. `repeat 4:` then, indented, `forward()` — the robot needs **4** steps, not 5. Count the *moves between squares*, not the squares.

D. `2 5 +` — **syntax** (the symbols are in the wrong order).
`3 / 'hi'` — **semantics** (written correctly, but dividing by a word means nothing).
`'hello' * 3` — **neither**; this is perfectly valid and repeats the string three times.
`print('hi'` — **syntax** (the closing bracket is missing).
`'5' + 5` — **semantics** (a string and a number cannot be added, even though `'5'` looks like a number).

E. **S, S, N, S, N, N.** Watch item 3: `'a'` is a *string* that happens to be one character long — strings are always non-scalar. Items 1, 2 and 4 are a number, a truth value and a number, so all scalar.

F1. Because 1s and 0s are unreadable and painfully slow for humans to write; a high-level language is close to English, and an interpreter does the translating for us.

F2. **No.** The rules of writing were all followed, so the computer was happy to run it. The program has exactly one meaning — it just is not the meaning you intended.