Quiz 1 — Solutions

Lei Wang (Griffith University)

*The solutions are prepared specifically for **Brisbane City (South Bank) students in 1811ICT only** and are intended to support their understanding of the assessed concepts and solutions.*

Questions 1-4 are drawn from question pools, so the code, values, and answer choices may vary; Q5-6 are also drawn from pools, with the specific conditions or divisors varying.

Quiz 1 at a glance

Question	Format	Marks	Main skill
Q1	identify syntax errors	2	Recognise Python syntax/structure errors
Q2	choose output	1	Trace if / elif / else
Q3	choose output	1	Trace for, range(), step and counter
Q4	choose output	1	Trace while, %, break and updates
Q5	Coding	5	for loop, range(), condition and output
Q6	Coding	5	while, continue, filtering, min/max

Q1: Finding syntax errors

Q1 asks you to inspect a short Python program and identify two locations containing syntax errors.

A syntax error means Python cannot correctly parse the program according to Python's grammar.

Look for missing punctuation, incorrect keywords, malformed expressions, and incorrect structure.

What to check

- Missing colon (:) after if, elif, else, for or while.
- Missing or mismatched quotation marks.
- Missing closing),], or }.
- Incorrectly written keywords such as if, elif or else.

- Malformed assignment or expression syntax.
- Incorrect indentation when it causes the program structure to be invalid.
- Remember that = is assignment; == is comparison. A wrong operator can also change the logic even when the syntax is valid.

Example:

```
name = input("Enter your name: ")
age = int(input("Enter your age: "))
if age < 60:
    print(f"Sorry, {Name}, you are too young.")
else if age >= 60:
    print(f"Welcome, {name}! You can join the seniors club.")
```

Check both syntax and identifier/variable-name problems. Look for missing punctuation, incorrect Python keywords, unmatched brackets or quotation marks, indentation/structure problems, and inconsistent variable names such as `name` vs `Name`. Remember that Python variable names are case-sensitive.

The key structural issue is the use of 'else if'. Python uses 'elif' for an additional condition. Also check the exact punctuation and quotation marks.

```
if age < 60:
    print(f"Sorry, {name}, you are too young.")
elif age >= 60:
    print(f"Welcome, {name}! You can join the seniors club.")
```

Strategy

1. Read the program once without trying to understand every output.
2. Scan each line for :, parentheses and quotation marks.
3. Check if/elif/else structure and indentation.
4. Check that keywords are written exactly as Python expects.
5. If two places are requested, select two actual syntax errors, not merely lines that could be improved.

Q2: Tracing if / elif / else

Q2 asks which output is produced by a short conditional program. The important skill is understanding that an if/elif/else chain selects the first condition that evaluates to True.

In an if/elif/else chain, Python checks conditions from top to bottom and executes only the first True branch.

```
tuition = 3000

if tuition < 5000:
    print("Tuition is less than 5000")
elif tuition < 4000:
    print("Tuition is less than 4000")
elif tuition < 3500:
    print("Tuition is less than 3500")
elif tuition == 3000:
    print("Tuition is 3000")
```

For tuition = 3000, the first condition, tuition < 5000, is already True. Therefore Python prints that message and does not continue checking the remaining elif branches.

What you should understand

- Conditions are evaluated in order.
- Only one branch in an if/elif/else chain executes.
- A later condition can be True but still never execute if an earlier condition is already True.
- Do not choose an answer merely because a statement is mathematically true; determine which branch Python reaches first.

Q3: Tracing for and range()

Q3 tests whether you can determine the values generated by range() and how many times a for loop executes. The example uses a step of 2 and a counter variable.

```
y = 0
for i in range(1, 9, 2):
    y += 1
print(y)
```

range(start, stop, step) includes start, repeatedly adds step, and stops before stop.

range(1, 9, 2) produces 1, 3, 5, 7. It does not include 9. The loop therefore executes four times, so y becomes 4.

Range patterns to know

Expression	Values	Iterations
range(5)	0, 1, 2, 3, 4	5
range(2, 7)	2, 3, 4, 5, 6	5
range(1, 9, 2)	1, 3, 5, 7	4
range(10, 4, -2)	10, 8, 6	3

Strategy

- Write down the exact sequence produced by range().
- Remember that the stop value is excluded.
- Count the iterations.
- Track any variable changed inside the loop.
- Only then select the output.

Q4: Tracing while, %, break and Updates

Q4 tests several Week 4 ideas together. The example below uses a while loop, a remainder test and break.

```
i = -1
while i < 10:
    if i % 3 == 0:
        break
    i += 3
    print(i, end=' ')
```

Trace the value of i after each iteration.

Starting at -1, the first test -1 % 3 == 0 is False, so i becomes 2 and 2 is printed.

Next, 2 % 3 == 0 is False, so i becomes 5 and 5 is printed.

Next, 5 % 3 == 0 is False, so i becomes 8 and 8 is printed.

Next, 8 % 3 == 0 is False, so i becomes 11 and 11 is printed.

The while condition is then checked again; 11 < 10 is False, so the loop ends.

break immediately terminates the entire loop. It does not merely skip the current iteration.

Trace table method

Before test	Condition	Update	Printed	Next check
-1	-1 % 3 == 0 → False	-1 + 3 = 2	2	2 < 10
2	2 % 3 == 0 → False	2 + 3 = 5	5	5 < 10
5	5 % 3 == 0 → False	5 + 3 = 8	8	8 < 10
8	8 % 3 == 0 → False	8 + 3 = 11	11	11 < 10 → False

Q4 concepts to recognise

- while condition: controls whether another iteration starts.
- %: returns the remainder after division.
- ==: compares two values.
- break: exits the loop immediately.
- i += 3: updates i by adding 3.
- end=' ': keeps subsequent output on the same line with a space.

What Q5 & Q6 assess

Question	Task	Main skills
Q5	Two integers → print qualifying values	if/else, for, range(), %, output formatting
Q6	Repeated integers → maximum/minimum of qualifying values	while, continue, sentinel 0, flag, comparisons

Q5: Two Integers and a for Loop

Q5 asks you to enter two integers and print all integers between and including them that satisfy a condition. The values must be printed in descending order, on one line, separated by spaces.

> Find the larger and smaller values → loop downward with one for loop → test each number with % → print only qualifying numbers.

- Input two integers.
- Determine the larger and smaller values.
- Use one for loop and no while loop.
- Use range() in descending order and include the smaller endpoint.
- Use % for the divisibility test.
- Print on the same line using end=' '.
- Do not use max() or min().

General solution

```
num1 = int(input("Enter first integer: "))
num2 = int(input("Enter second integer: "))

if num1 > num2:
    start = num1
    end = num2
else:
    start = num2
    end = num1

for i in range(start, end - 1, -1):
    if i % DIVISOR == 0:
        print(i, end=" ")
```

Replace DIVISOR and, when required, the comparison condition to match the question.

Q5.1 — Divisible by 3

```
num1 = int(input("Enter first integer: "))
num2 = int(input("Enter second integer: "))

if num1 > num2:
    start = num1
    end = num2
else:
    start = num2
    end = num1

for i in range(start, end - 1, -1):
    if i % 3 == 0:
        print(i, end=" ")
```

The condition i % 3 == 0 keeps numbers whose remainder after division by 3 is zero.

Q5.2 — Not divisible by 5

```
num1 = int(input("Enter first integer: "))
num2 = int(input("Enter second integer: "))

if num1 > num2:
    start = num1
```

```
    end = num2
else:
    start = num2
    end = num1

for i in range(start, end - 1, -1):
    if i % 5 != 0:
        print(i, end=" ")
```

The condition changes to != 0 because the question asks for numbers that cannot be divided evenly by 5.

Q5.3 — Divisible by 4

```
num1 = int(input("Enter first integer: "))
num2 = int(input("Enter second integer: "))

if num1 > num2:
    start = num1
    end = num2
else:
    start = num2
    end = num1

for i in range(start, end - 1, -1):
    if i % 4 == 0:
        print(i, end=" ")
```

The algorithm is unchanged; only the divisor changes.

Q5 marking guide — 5 marks

Criterion	Marks
Input the two integers correctly	1
Find the larger and smaller values correctly	1
Use a single for loop with the correct range()	1
Output the correct numbers in the correct order	1
Print on the same line with spaces	1

Q5 common mistakes

- Using range(start, end, -1), which excludes the smaller endpoint.
- Using the wrong loop direction.
- Testing the wrong variable, e.g. 1 % 3 instead of i % 3.
- Using print(i) and therefore putting each result on a new line.
- Confusing divisible (== 0) with not divisible (!= 0).
- Using more than one for loop.

Q6: Repeated input, continue, and maximum/minimum

Q6 repeatedly reads integers until 0 is entered. Values meeting the exclusion condition are skipped using continue. Among the remaining values, the program finds the maximum and minimum. If there are no qualifying values, it prints the required message.

while num != 0 → reject unwanted values with continue → use the first qualifying value to initialise maximum/minimum → compare later qualifying values → handle the no-value case.

- Use a while loop.
- Use 0 as the sentinel.
- Use continue to skip excluded values.
- Do not use break.
- Do not use max() or min().
- Initialise maximum and minimum from the first qualifying value.
- Update both values manually.
- Handle the case where no qualifying value exists.

General solution

```
num = int(input("Enter an integer (0 to stop): "))

found = False

while num != 0:
    if num % DIVISOR == 0:
        num = int(input("Enter an integer (0 to stop): "))
        continue

    if found == False:
        maximum = num
        minimum = num
        found = True
    else:
        if num > maximum:
            maximum = num
        if num < minimum:
            minimum = num

    num = int(input("Enter an integer (0 to stop): "))

if found == False:
    print("NO QUALIFYING NUMBER")
else:
    print("Maximum:", maximum)
    print("Minimum:", minimum)
```

Q6.1: Maximum and minimum odd numbers

```
num = int(input("Enter an integer (0 to stop): "))

found = False

while num != 0:
    if num % 2 == 0:
        num = int(input("Enter an integer (0 to stop): "))
        continue

    if found == False:
```

```
            maximum = num
            minimum = num
            found = True
        else:
            if num > maximum:
                maximum = num
            if num < minimum:
                minimum = num

        num = int(input("Enter an integer (0 to stop): "))

    if found == False:
        print("No odd number is entered")
    else:
        print("Maximum odd number:", maximum)
        print("Minimum odd number:", minimum)
```

Q6.2: Maximum and minimum values not divisible by 3

```
num = int(input("Enter an integer (0 to stop): "))

found = False

while num != 0:
    if num % 3 == 0:
        num = int(input("Enter an integer (0 to stop): "))
        continue

    if found == False:
        maximum = num
        minimum = num
        found = True
    else:
        if num > maximum:
            maximum = num
        if num < minimum:
            minimum = num

    num = int(input("Enter an integer (0 to stop): "))

if found == False:
    print("All numbers entered can be divided by 3")
else:
    print("Maximum:", maximum)
    print("Minimum:", minimum)
```

Q6.3: Maximum and minimum values not divisible by 4

```
num = int(input("Enter an integer (0 to stop): "))

found = False

while num != 0:
    if num % 4 == 0:
        num = int(input("Enter an integer (0 to stop): "))
        continue

    if found == False:
        maximum = num
        minimum = num
        found = True
    else:
        if num > maximum:
```

```
            maximum = num
        if num < minimum:
            minimum = num

    num = int(input("Enter an integer (0 to stop): "))

if found == False:
    print("All numbers entered can be divided by 4")
else:
    print("Maximum:", maximum)
    print("Minimum:", minimum)
```

Q6 marking guide — 5 marks

Criterion	Marks
while loop is used correctly	1
continue is used correctly	1
No-qualified-number case is handled correctly	1
Maximum and minimum are computed correctly in all cases	1
Overall logic is correct and satisfies the stated restrictions	1

Why the found flag matters in Q6

Do not initialise maximum and minimum to 0. The input may contain negative numbers. Instead, use the first qualifying value as both maximum and minimum, then compare every later qualifying value against them.

```
found = False

if found == False:
    maximum = num
    minimum = num
    found = True
else:
    if num > maximum:
        maximum = num
    if num < minimum:
        minimum = num
```

Example: If the qualifying values are -8, -3 and -12, the correct maximum is -3 and the correct minimum is -12. Starting both values at 0 would give the wrong result.

Q6 trace example

For the 'not divisible by 4' version, suppose the user enters 8, 12, 7, 3, 20, 5, 0.

Input	Divisible by 4?	Action	Minimum	Maximum

8	Yes	Skip	—	—
12	Yes	Skip	—	—
7	No	First valid value	7	7
3	No	Update	3	7
20	Yes	Skip	3	7
5	No	Update	3	7
0	Stop	Finish	3	7

Common Q6 mistakes

- Using break even though it is forbidden.
- Using continue without reading the next number first, which can cause an infinite loop.
- Processing an excluded number before skipping it.
- Initialising maximum/minimum to 0.
- Using max() or min().
- Forgetting to handle the case where every value is excluded.
- Using == 0 when the question asks for not divisible, or != 0 when it asks for divisible.
- Forgetting the final input statement inside the loop.

Coding quiz strategy

1. Read every restriction before coding.
2. For Q5, solve the larger/smaller and descending-range problem first.
3. For Q6, write the while-loop and sentinel structure first, then add continue and the maximum/minimum logic.
4. Test reversed inputs, equal inputs, no qualifying values, one qualifying value, negative values, and 0.
5. Check output formatting as well as the algorithm.