IQM3010

LEVEL 3 · CONTROLLED STAFF DOCUMENT

INTO QUALIFICATIONS

Computer Science

Assessment 2: Written exam

Assessment window Unassigned (sample)	**Exam date** _____
Paper Paper 2	**Assessment set** Sample 2026 · ID 2026-S

STAFF ONLY · Staff mark scheme and AO2 marking map

Duration
2 hours

Total marks
100

Assessment window
Unassigned (sample)

Assessment set
Set S of 2026

Audience
Staff only

Release timing
exam release

Exam date
_____

Assessment set ID
2026-S

STAFF ONLY - The complete paper maps to AO2. Credit valid equivalent responses where they satisfy the approved set-specific criteria.

Section A - Short Answer and Object-Oriented Programming

Locked section budget: 20 marks. Objective allocation: AO2.

ID	Approved answer / marking points	Alternative valid responses	Max	AO
A1	1 mark: Each object has its own instance state/title attribute. 1 mark: Assignment through one instance changes that instance, rather than the other instance. Mapping: LO36, LO38; AO2 2.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	2	AO2
A2	1 mark: The first parameter is self (or a consistently used equivalent). 1 mark: The assignment is self.label = label. Mapping: LO36, LO38; AO2 2.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	2	AO2
A3	1 mark: Calls room.free_spaces() with parentheses and no explicit self argument. 1 mark: Assigns the result to spaces: spaces = room.free_spaces(). Mapping: LO36, LO38; AO2 2.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	2	AO2
A4	1 mark: Create self.attempts = [] within __init__. 1 mark: This creates a separate list for each object, avoiding an unintentionally shared mutable class list. Mapping: LO36, LO38; AO2 2.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	2	AO2

ID	Approved answer / marking points	Alternative valid responses	Max	AO
A5	1 mark: class DigitalBook(Book): 1 mark: Inherits title_text(), which returns its title as a string, without redefining that method. Credit an equivalent accurate description of this supplied behaviour. Mapping: LO38; AO2 2.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	2	AO2
A6	1 mark: Uses super().__init__(...) or Book.__init__(self, ...). 1 mark: Passes title correctly: super().__init__(title). Mapping: LO38; AO2 2.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	2	AO2
A7	1 mark: Each class supplies its own implementation/overrides price. 1 mark: Calling the method on each object uses that object’s implementation, so the same call can return its appropriate price. Mapping: LO36, LO38; AO2 2.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	2	AO2
A8	1 mark: return len(self.books) 1 mark: books alone is an unqualified name; self.books accesses the current object’s stored attribute. Mapping: LO36, LO38; AO2 2.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	2	AO2

ID	Approved answer / marking points	Alternative valid responses	Max	AO
A9	1 mark: Request 4 places. 1 mark: Accept the booking, leaving zero places; a return value such as True is valid when explained. Mapping: LO37, LO38; AO2 2.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	2	AO2
A10	1 mark: n comparisons (linear/O(n)). 1 mark: The target is absent or is the final object in the list. Mapping: LO32, LO36; AO2 2.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	2	AO2

Section B - Code Analysis

Locked section budget: 30 marks. Objective allocation: AO2.

ID	Approved answer / marking points	Alternative valid responses	Max	AO
B1	1 mark: First printed line: True. 1 mark: Second printed line: False. 1 mark: Third printed line: 5 8. 1 mark: Each constructor assigns an instance attribute self.total. 1 mark: The method updates the total through the receiving object; a and b are different instances. 1 mark: a.add(0) returns False. 1 mark: a.total remains 5. 1 mark: The guard amount <= 0 returns before any update. 1 mark: a.add(-1) returns False. 1 mark: a.total remains 5. Mapping: LO36, LO37, LO38; AO2 10.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	10	AO2
B2	1 mark: Change count < 0 to count <= 0. 1 mark: This also rejects zero, which is not a positive request. 1 mark: Change count >= self.remaining to count > self.remaining. 1 mark: Equality must be accepted because exactly enough seats remain. 1 mark: Change self.remaining = count to self.remaining = self.remaining - count (or -= count). 1 mark: Subtracts the booked seats from existing capacity rather than replacing capacity with the request. 1 mark: Request 0: False, remaining 5. 1 mark: Request 5: True, remaining 0. 1 mark: Request 1: False, remaining 0. 1 mark: For example fresh capacity 5, request -1: False with remaining 5. Mapping: LO36, LO37, LO38; AO2 10.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	10	AO2

ID	Approved answer / marking points	Alternative valid responses	Max	AO
B3	2 marks: First line A 5: one mark code A, one fee 5. 2 marks: Second line B 8: one mark code B, one fee 8. 1 mark: Calls the inherited/parent Job constructor on the current object. 1 mark: Initialises that ExpressJob instance’s code attribute using code. 1 mark: Job("A") uses Job.fee and returns 5. 1 mark: ExpressJob("B", 3) uses its overriding fee, adding bonus 3 to inherited fee 5. 1 mark: The loop executes once per job, so n iterations become 2n. 1 mark: Linear/O(n) assuming constant-time fee calculations. Mapping: LO32, LO36, LO37, LO38; AO2 10.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	10	AO2

Section C - Code Writing and Inheritance

Locked section budget: 50 marks. Objective allocation: AO2.

ID	Approved answer / marking points	Alternative valid responses	Max	AO
C1	1 mark: Defines class KitPool with appropriate method signatures/self. 1 mark: Constructor stores self.code. 1 mark: Constructor stores self.capacity. 1 mark: Initialises separate self.available to capacity. 1 mark: checkout rejects non-positive count. 1 mark: checkout rejects count > available. 1 mark: checkout rejection returns False. 1 mark: checkout rejection leaves state unchanged. 1 mark: Valid checkout subtracts count from available once. 1 mark: Valid checkout returns True. 1 mark: return_kits rejects non-positive count. 1 mark: return_kits rejects count greater than capacity - available/on_loan(). 1 mark: return_kits rejection returns False. 1 mark: return_kits rejection leaves state unchanged. 1 mark: Valid return adds count to available once. 1 mark: Valid return returns True. 1 mark: on_loan calculates capacity - available. 1 mark: on_loan returns that value without changing state. 1 mark: Constructs pool = KitPool("K1", 6) or equivalent. 1 mark: Calls pool.checkout(2). Mapping: LO36, LO37, LO38; AO2 20.	Credit equivalent valid Python 3 solutions and explanations. Award each stated feature once; do not penalise the same error repeatedly.	20	AO2
C2	1 mark: Declares class LimitedPool(KitPool). 1 mark: Constructor signature accepts code, capacity and limit with self. 1 mark: Calls parent __init__ on current object. 1 mark: Passes code and capacity correctly. 1 mark: Stores self.limit = limit. 1 mark: Overrides checkout with self and count. 1 mark: Tests count > self.limit. 1 mark: Returns False without mutation when above limit. 1 mark: Otherwise calls parent checkout with count, avoiding recursive self.checkout. 1 mark: Returns the result of the parent call. 1 mark: Creates limited = LimitedPool("L1", 8, 3). 1 mark: first = limited.checkout(4). 1 mark: second = limited.checkout(3). 1 mark: first is False and second is True (both required). 1 mark: limited.available is 5. Mapping: LO36, LO37, LO38; AO2 15.	Credit equivalent explicit parent calls. Inheriting return_kits and on_loan requires no repeated definitions. Mark independently of C1; the given parent interface is available to every candidate.	15	AO2

ID	Approved answer / marking points	Alternative valid responses	Max	AO
C3	1 mark: Defines request_kits with the three parameters. 1 mark: Iterates over the supplied list. 1 mark: Compares each object's code to the requested code. 1 mark: Calls checkout on the matching object. 1 mark: Passes count unchanged. 1 mark: Calls checkout exactly once for the match. 1 mark: Returns the checkout Boolean, including False on rejection. 1 mark: Does not return False prematurely after a non-match. 1 mark: Returns False after exhausting the list, including empty list. 1 mark: Preserves the list and uses the method interface without direct state mutation. 1 mark: Defines total_on_loan with pools parameter. 1 mark: Initialises a numeric total to 0. 1 mark: Visits all objects and calls each object's on_loan(). 1 mark: Adds returned values to the total once per object. 1 mark: Returns the sum after iteration, including 0 for empty list. Mapping: LO36, LO37, LO38; AO2 15.	Credit sum comprehensions or other equivalent valid solutions for total_on_loan. A search may use a Boolean found flag. Do not require a particular local variable name. Mark independently of C1/C2 using the supplied interfaces.	15	AO2

C1 · Executable model solution

Illustrative solution only. Apply the item's feature marks to equivalent valid solutions; do not require identical variable names or this exact implementation.

```
class KitPool:
    def __init__(self, code, capacity):
        self.code = code
        self.capacity = capacity
        self.available = capacity

    def checkout(self, count):
        if count <= 0 or count > self.available:
            return False
        self.available -= count
        return True

    def return_kits(self, count):
        if count <= 0 or count > self.on_loan():
            return False
        self.available += count
        return True

    def on_loan(self):
        return self.capacity - self.available

pool = KitPool("K1", 6)
pool.checkout(2)
```

C2 · Executable model solution

Illustrative solution only. Apply the item's feature marks to equivalent valid solutions; do not require identical variable names or this exact implementation.

```
class LimitedPool(KitPool):
    def __init__(self, code, capacity, limit):
        super().__init__(code, capacity)
        self.limit = limit

    def checkout(self, count):
        if count > self.limit:
            return False
        return super().checkout(count)

limited = LimitedPool("L1", 8, 3)
first = limited.checkout(4)
second = limited.checkout(3)
```

C3 · Executable model solution

Illustrative solution only. Apply the item's feature marks to equivalent valid solutions; do not require identical variable names or this exact implementation.

```
def request_kits(pools, code, count):
    for pool in pools:
        if pool.code == code:
            return pool.checkout(count)
    return False

def total_on_loan(pools):
    total = 0
    for pool in pools:
        total += pool.on_loan()
    return total
```

Marking and source reconciliation

Award method and feature marks where intent is clear. A minor syntax slip must not erase independently demonstrated logic. Python-specific syntax required by a short item remains part of that item. Do not award the same feature twice. Mark C1–C3 independently using the supplied interfaces.

Follow September guide Python/OOP/inheritance requirements over HS1.3 language-neutral-only instructions. Follow guide no-equipment condition over template calculator permission. Preserve template section budgets 20/30/50 and item maxima 10×2, 3×10, 20/15/15.

AO2 applies to all items per guide and template. Short answers use concrete programming contexts rather than isolated definitions. Directly scored outcomes are sampled within LO29–38. Python objects, lists, assignments, control structures and functions map to LO36–38; prerequisite LO24/26/28 are not claimed as assessed Paper 2 outcomes. LO29–31 specifically require pseudocode/flowcharts and are not claimed for Python-only responses. A10 and B3 additionally assess LO32 through explicit questions about how work grows with input size; C3 implementation alone does not assess efficiency. All 100 marks remain AO2; section totals remain 20/30/50.

C1–C3 share a fully stated interface. Explicit independence instructions prevent earlier incomplete code from blocking later marks. No extra AO3 allocation or separate performance-band caps invented.