Editorial-Evaluation: Mid Module A Quiz

Question 1

You are designing a smart irrigation system that waters crops based on soil moisture levels. If the moisture level is below 30%, the system turns on the water pump.

Which of the following Python code snippets correctly implements this logic?

Options: A) if moisture_level < 30: turn_on_pump()
B) if moisture_level <= 30: turn_on_pump()
C) if moisture_level > 30: turn_on_pump()
D) if moisture_level == 30: turn_on_pump()

Correct Answer:
A

Explanation:

- The system should turn on the pump when the moisture level is **below 30%**.
- Option A correctly implements this condition using < 30.
- Option B (<= 30) includes **30%**, which is incorrect.
- Options C and D have incorrect conditions.

Question 2

A bank uses selection sort to arrange customer transaction records in ascending order based on transaction amounts. What is the worst-case time complexity of this approach?

Options: A) O(N log N)
B) O(N)
C) O(N^2)
D) O(1)

Correct Answer:
C

Explanation:

- Selection Sort has a worst-case **time complexity of O(N^2)**.
- It always runs in O(N^2) since it repeatedly finds the minimum and swaps elements.
- O(N log N) (Option A) is the complexity of **Merge Sort** and **Quick Sort** (average case).

Question 3

You are implementing a password validation system where the rules are:

- Must be at least 8 characters long
- Must contain at least one uppercase letter
- Must contain at least one digit

Which of the following code snippets correctly implement the validation logic?

Options:

A)

```
if len(password) > 8:
   print("Valid password")
```

B)

```
if len(password) >= 8 and any(c.isupper() for c in password) and any(c.isdigit() for c in password):
   print("Valid password")
```

C)

```
if password.isalnum() and len(password) >= 8:
   print("Valid password")
```

D)

```
if len(password) >= 8 and any(c.isupper() for c in password) and any(c.isdigit() for c in password):
   print("Valid password")
else:
   print("Invalid password")
```

Correct Answers:
B, D

Explanation:

- The password must be **at least 8 characters long**, contain **one uppercase letter**, and **one digit**.
- **Option B** correctly checks all conditions.
- **Option D** does the same as B but includes an explicit "Invalid password" message.
- **Option A** only checks the length.
- **Option C** only checks isalnum() (which allows letters and numbers but doesn't enforce uppercase).

Question 4

You need to process data from a CSV file containing sales records. Which exception handling techniques should be used to prevent errors due to missing or corrupted data?

Options:

A) try-except FileNotFoundError
B) try-except ZeroDivisionError
C) try-except KeyError
D) try-except ValueError

Correct Answers:
A, D

Explanation:

- **Option A (FileNotFoundError)** prevents crashes when the file is missing.
- **Option D (ValueError)** handles incorrect data types in numeric fields.

Question 5: Warehouse Inventory System

A warehouse manages its inventory using a dictionary where product names are the keys and stock counts are the values.
Which of the following efficiently checks if an item exists in the inventory?

Options:

A) if item in inventory:
B) inventory.contains(item)
C) inventory.search(item)
D) inventory.find(item)

Correct Answer: A

Explanation:

- The in keyword is the correct way to check for a **key** in a dictionary.

Question 6

A self-driving car monitors traffic lights to determine when to stop. If the light is "Red", the car must stop.

Which conditional statement correctly implements this logic?

Options: A) if light_color = "Red": stop_car()
B) if light_color == "Red": stop_car()

C) if light_color != "Red": stop_car()
D) if light_color > "Red": stop_car()

Correct Answer:
B

Explanation:

- **Option B** uses the correct **comparison operator (==)**.

Question 7

Which sorting algorithms maintain the relative order of duplicate elements (i.e., are **stable sorts**)?

Options: A) Merge Sort
B) Quick Sort
C) Insertion Sort
D) Selection Sort

Correct Answers:
A, C

Explanation:

- **Stable sorting algorithms** preserve the order of duplicate elements.
- **Merge Sort (A) and Insertion Sort (C)** are stable.

Question 8

A social media platform stores user posts in a list. To quickly find a post by ID, which search algorithm should be used **if the list is sorted**?

Options: A) Linear Search
B) Binary Search

Correct Answer:
B

Explanation:

- **Binary Search (O(log N))** is the fastest way to find an item in a sorted list.

Question 9

You are developing a recommendation system that stores past user searches in a list. What is the best way to remove duplicate searches from the list?

Options: A) list(set(searches))
B) searches.remove_duplicates()
C) searches.unique()
D) searches.find_duplicates()

Correct Answer:
A

Explanation:

- **Converting to a set and back to a list** removes duplicates efficiently.

Question 10

Which of the following correctly writes data to a file?

Options: A) file.write('data')
B) file.append('data')
C) file.insert('data')
D) file.read('data')

Correct Answer:
A

Explanation:

- **write()** is the correct function to write to a file.

Question 11: Budget Planner

You are developing a **budget planner** for a family to track their monthly expenses. You need to decide whether a person can make a particular purchase based on available funds.

Which of the following Python expressions will correctly evaluate whether a purchase can be made?

Options:

A) can_buy = budget > expense

B) can_buy = budget >= expense

C) can_buy = budget < expense

D) can_buy = budget == expense

Correct Answer: B

Explanation:

- A person can make a purchase if the **budget is at least equal to the expense**.

Question 12: Smart Home Automation

A smart home system controls the room lights based on whether a person is in the room or not. If a person enters the room, the light should turn ON, otherwise, it should remain OFF.

Which logical operation correctly represents this condition?

Options:

A) light_on = person_in_room and True

B) light_on = not person_in_room

C) light_on = person_in_room or False

D) light_on = not (person_in_room and False)

Correct Answer: A

Explanation:

- The light should be on **if the person is in the room**.

Question 13: Traffic Control System

A city's traffic light system works based on a timer. The green light remains on for **60 seconds**, yellow for **10 seconds**, and red for **30 seconds**.

What conditional statement ensures the traffic light follows this cycle?

Options:

A) if time < 60: light = 'Green' elif time < 70: light = 'Yellow' else: light = 'Red'

B) if time < 60: light = 'Green' elif time < 30: light = 'Yellow' else: light = 'Red'

C) if time >= 0 and time <= 60: light = 'Green' elif time > 60 and time <= 70: light = 'Yellow' else: light = 'Red'

D) if time < 60: light = 'Yellow' elif time < 70: light = 'Green' else: light = 'Red'

Correct Answer: A

Explanation:

- This correctly defines the **traffic light timing logic**.

Question 14: Online Shopping Cart

An **e-commerce website** offers a discount when customers buy at least 5 items.

Which Python statement correctly checks if the customer qualifies for a discount?

Options:

A) if items < 5: discount = True

B) if items >= 5: discount = True

C) if items > 5: discount = False

D) if items == 5: discount = False

Correct Answer: B

Explanation:

- Customers **qualify for a discount if they buy 5 or more items**.

Question 15: Robot Vacuum Cleaner

A robot vacuum cleaner stops moving when its battery level drops below **20%**, but it continues if the house is still dirty.

Which Python loop condition best represents this logic?

Note: Ambiguous question-Marks are alloted for this question.

Options:

A) while battery > 20 or dirty: continue

B) while battery < 20 and dirty: continue

C) while battery > 20 and dirty: continue

D) while battery <= 20 or not dirty: continue

Correct Answer: C

Explanation:

The robot vacuum should continue working only if:

1. The battery level is above 20% (battery > 20).
2. The house is still dirty (dirty == True).

When either condition fails, the vacuum should stop. Only option C presents the correct logic with and. Only then vacuum continues running.

Question 16: Library Management System

A library only allows borrowing books if a user has no **overdue books**.

Which Python condition correctly ensures a user can borrow a book?

Options:

A) if overdue_books == 0: allow_borrow = True

B) if overdue_books != 0: allow_borrow = True

C) if overdue_books > 0: allow_borrow = True

D) if overdue_books < 0: allow_borrow = False

Correct Answer: A

Explanation:

- A book **can be borrowed only if there are no overdue books**.

Question 17: Fitness Tracker

A fitness tracker calculates the **average steps per day** over a **7-day period**.

Which Python code snippet correctly computes this?

Options:

A) avg_steps = sum(steps) * 7

B) avg_steps = sum(steps) / 7

C) avg_steps = steps / 7

D) avg_steps = total(steps) / 7

Correct Answer: B

Explanation:

- **Summing and dividing by 7 correctly calculates the average**.

Question 18: Warehouse Inventory System

A warehouse keeps track of items in stock using a **list**. To find whether an item is available, the system searches for it.

Which Python statement correctly checks if an item is in stock?

Options:

A) if item in warehouse_stock: print('Available')

B) if warehouse_stock.find(item): print('Available')

C) if item.contains(warehouse_stock): print('Available')

D) if warehouse_stock[item]: print('Available')

Correct Answer: A

Explanation:

- **Checking in a list is the correct way to see if an item exists**.

Question 19:

Ride-Sharing App - Fare Calculation (Mathematical Operators)

A ride-sharing app calculates the fare based on the following criteria:

- Base fare = **₹50**
- Per kilometer charge = **₹10**
- If the total distance is more than **10 km**, a **10% discount** is applied to the total fare.

Which of the following code snippets correctly calculates the final fare?

A)

```
def calculate_fare(distance):
    fare = 50 + (distance * 10)
    if distance > 10:
        fare = fare * 0.9
    return fare

print(calculate_fare(12))
```

B)

```
def calculate_fare(distance):
    fare = 50 + (distance * 10)
    return fare if distance <= 10 else fare * 0.9

print(calculate_fare(12))
```

C)

```
def calculate_fare(distance):
    return (50 + distance * 10) * (0.9 if distance > 10 else 1)

print(calculate_fare(12))
```

D)

```
def calculate_fare(distance):
    return 50 + distance * 10 - (10 if distance > 10 else 0)


print(calculate_fare(12))
```

Correct Answer: A, B, C

Explanation:

- **All three options correctly implement the discount condition**.

Question 20: Gym Membership - Discount Eligibility (Comparison & Logical Operators)

A gym offers discounts based on the following conditions:

- If **age is below 18 or above 60**, a **20% discount** is given.
- If the person is a **student** (age < 25) and has an ID, a **15% discount** is given.
- Otherwise, there is no discount.

Which of the following code snippets correctly applies the discount?

A)

```
def get_discount(age, is_student, has_id):
    if age < 18 or age > 60:
        return 20
    elif is_student and has_id and age < 25:
        return 15
    return 0
```

B)

```
def get_discount(age, is_student, has_id):
    return 20 if age < 18 or age > 60 else (15 if is_student and has_id and age < 25 else 0)
```

C)

```
def get_discount(age, is_student, has_id):
    discount = 0
    if age < 18 or age > 60:
        discount = 20
    elif is_student and has_id and age < 25:
        discount = 15
    return discount
```

D)

```
def get_discount(age, is_student, has_id):
    return 15 if is_student and has_id and age < 25 else 20 if age < 18 or age > 60 else 0
```

Correct Answer: A, B, C

Question 21: Smart Irrigation System – Loop with Break and Continue

A smart irrigation system waters plants based on soil moisture levels. The system should:

- **Stop watering** (break) if soil moisture is **above 80%**.
- Skip watering (continue) if the moisture level is **above 60%** but **less than or equal to 80%**.
- Otherwise, water the plant.

Which of the following code snippets correctly implements this logic?

Options:

A)

```
moisture_levels = [45, 65, 55, 82, 75, 30]


for level in moisture_levels:
    if level > 80:
        break
    if level > 60 and level <= 80:
        continue
    print(f"Watering at moisture level {level}")
```

B)

```
moisture_levels = [45, 65, 55, 82, 75, 30]
```

```
for level in moisture_levels:
    if level > 80:
        break
    elif 60 < level <= 80:
        continue
    else:
        print(f"Watering at moisture level {level}")
```

C)

```
moisture_levels = [45, 65, 55, 82, 75, 30]

for level in moisture_levels:
    if level > 80:
        break
    if level <= 60:
        print(f"Watering at moisture level {level}")
```

D)

```
moisture_levels = [45, 65, 55, 82, 75, 30]

for level in moisture_levels:
    if level <= 60:
        print(f"Watering at moisture level {level}")
    elif level > 80:
        break
```

Correct Answer: A, B, C, D

Question 22 E-Learning Platform - Student Performance Tracking (List Operations)

An e-learning platform tracks student quiz scores in a list. It needs to:

1. Add a new score to the list.
2. Find the **highest** and **lowest** scores.

3. Calculate the **average score**.

Which of the following code snippets correctly implements this?

A)

```
scores = [80, 90, 70, 85, 88]
scores.append(92)

print("Highest:", max(scores))
print("Lowest:", min(scores))
print("Average:", sum(scores) / len(scores))
```

B)

```
scores = [80, 90, 70, 85, 88]
scores.insert(len(scores), 92)

print("Highest:", max(scores))
print("Lowest:", min(scores))
print("Average:", sum(scores) / len(scores))
```

C)

```
scores = [80, 90, 70, 85, 88]
scores.append(92)

highest = sorted(scores)[-1]
lowest = sorted(scores)[0]
average = sum(scores) / len(scores)

print("Highest:", highest)
print("Lowest:", lowest)
print("Average:", average)
```

D)

```
scores = [80, 90, 70, 85, 88]
scores.insert(0, 92)
```

```
print("Highest:", scores[-1])
print("Lowest:", scores[0])
print("Average:", sum(scores) / len(scores))
```

Correct Answer: A, B, C

Explanation:

- The platform needs to:
    i. **Add a new score** → append() or insert().
    ii. **Find the highest & lowest scores** → max() and min().
    iii. **Calculate the average** → sum(scores) / len(scores).
- **Option A, B, and C correctly implement these tasks.**
- **Option D is incorrect because it assumes scores are sorted and uses scores[-1], which may not give the correct highest value.**

Question 23 Inventory Management System - Product Lookup (Dictionary Operations)

A store maintains an inventory of products in a dictionary with product names as keys and stock quantities as values. The system should:

1. Check if a product exists.
2. Add a new product.
3. Update the stock of an existing product.

Which of the following code snippets correctly implements this?

A)

```
inventory = {"Apples": 10, "Bananas": 5, "Oranges": 8}


if "Grapes" not in inventory:
    inventory["Grapes"] = 12


inventory["Bananas"] += 3
print(inventory)
```

B)

```
inventory = {"Apples": 10, "Bananas": 5, "Oranges": 8}
```

```
inventory.setdefault("Grapes", 12)
inventory["Bananas"] = inventory.get("Bananas", 0) + 3

print(inventory)
```

C)

```
inventory = {"Apples": 10, "Bananas": 5, "Oranges": 8}

inventory.update({"Grapes": 12})
inventory["Bananas"] += 3

print(inventory)
```

D)

```
inventory = {"Apples": 10, "Bananas": 5, "Oranges": 8}

if "Grapes" in inventory:
    inventory["Grapes"] = 12

inventory["Bananas"] += 3

print(inventory)
```

Correct Answer: A, B, C

Explanation:

- **Checking if a product exists** → if "Grapes" not in inventory: inventory["Grapes"] = 12.
- **Adding a new product** → inventory.setdefault("Grapes", 12).
- **Updating stock** → inventory["Bananas"] += 3.
- **Option A, B, and C correctly implement inventory operations.**
- **Option D is incorrect because it assumes "Grapes" is already in the inventory before updating.**

Question 24

You are programming a drone delivery system to locate items in different storage zones of a warehouse.

- **Zone A**: Items are randomly scattered (unsorted).
- **Zone B**: Items are sorted by weight in ascending order.

The drone's battery consumption depends on the number of comparisons it makes during a search. The drone needs to find an item with **ID P567**:

- Zone A has 500 items.
- Zone B has 1024 items.
- Each comparison in **Zone A** costs 2 units of battery, and each comparison in **Zone B** costs 1 unit of battery.

What is the total battery consumption in the **worst case** if the drone uses linear search in Zone A and binary search in Zone B?

Options: A) 500 units
B) 1010 units
C) 1000 units
D) 1024 units

Correct Answer:
B) 1010 units

Explanation:

- **Zone A (unsorted)** → Uses **Linear Search**, **worst-case comparisons = 500**.
    - Battery consumption: **500 × 2 = 1000 units**.
- **Zone B (sorted)** → Uses **Binary Search**, **worst-case comparisons = $\log_2(1024)$ = 10**.
    - Battery consumption: **10 × 1 = 10 units**.
- **Total = 1000 + 10 = 1010 units**.

Question 25 Online Library - Fine Calculation (Functions)

A library charges a fine for late book returns:

- **₹2 per day** for the first 7 days.
- **₹5 per day** for the next 7 days.
- If the book is more than **14 days late**, the user is banned.

Which function correctly calculates the fine?

A)

```
def calculate_fine(days):
    if days <= 7:
        return days * 2
    elif days <= 14:
        return (7 * 2) + (days - 7) * 5
    else:
        return "User banned"

print(calculate_fine(10))
```

B)

```
def calculate_fine(days):
    fine = 0
    if days > 14:
        return "User banned"
    if days > 7:
        fine += (days - 7) * 5
        days = 7
    fine += days * 2
    return fine

print(calculate_fine(10))
```

C)

```
def calculate_fine(days):
    return min(days, 7) * 2 + max(0, days - 7) * 5 if days <= 14 else "User banned"

print(calculate_fine(10))
```

D)

```
def calculate_fine(days):
    if days > 14:
        return "User banned"
    return (days * 2) + (days // 7) * 3
```

```
print(calculate_fine(10))
```

Correct Answer: A, B, C

Explanation:

- **Fine rules**:
    - **₹2 per day** for **first 7 days**.
    - **₹5 per day** for **next 7 days**.
    - **If >14 days, the user is banned.**
- **Options A, B, and C correctly implement the logic.**
- **Option D does not properly separate the first 7 days at ₹2 per day and the next 7 days at ₹5 per day. Instead, it applies a mixed fine calculation formula that does not align with the problem statement.**

Question 26 Store Inventory - Finding a Product (Linear Search)

A store maintains a product list. Customers can search for a product by name.
Which of the following functions correctly implements **linear search**?

A)

```
def search_product(products, target):
    for product in products:
        if product == target:
            return True
    return False


print(search_product(["Milk", "Eggs", "Bread"], "Eggs"))
```

B)

```
def search_product(products, target):
    return target in products


print(search_product(["Milk", "Eggs", "Bread"], "Eggs"))
```

C)

```
def search_product(products, target):
    return [product for product in products if product == target] != []


print(search_product(["Milk", "Eggs", "Bread"], "Eggs"))
```

D)

```
def search_product(products, target):
    index = 0
    while index < len(products):
        if products[index] == target:
            return True
        index += 1
    return False


print(search_product(["Milk", "Eggs", "Bread"], "Eggs"))
```

Correct Answer: A, B, C, D

Explanation:

- **Linear search checks each item one by one**.
- **All four options correctly implement linear search** using:
    - **Option A**: A for loop checking if product == target.
    - **Option B**: Python's built-in in operator.
    - **Option C**: A list comprehension.
    - **Option D**: A while loop implementation.

Question 27: Phonebook – Finding a Contact (Binary Search)

A phonebook stores contacts in **sorted order**.
Which function correctly implements **binary search** to find a contact?

Options:

A)

```
def binary_search(contacts, target):
    low, high = 0, len(contacts) - 1
    while low <= high:
        mid = (low + high) // 2
        if contacts[mid] == target:
            return True
        elif contacts[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return False
```

B)

```
def binary_search(contacts, target):
    return target in contacts
```

C)

```
def binary_search(contacts, target):
    if not contacts:
        return False
    mid = len(contacts) // 2
    if contacts[mid] == target:
        return True
    elif contacts[mid] < target:
        return binary_search(contacts[mid+1:], target)
    else:
        return binary_search(contacts[:mid], target)
```

D)

```
def binary_search(contacts, target):
    return any(contact == target for contact in contacts)
```

Correct Answer: A, C

Explanation:

- **Option A implements an iterative binary search.**
- **Option C implements a recursive binary search.**
- **B and D use linear search techniques**, which are incorrect for binary search.

Question 28:

Given the string s = "2024, September 14; ERROR - Log Entry: NONE."

What will the below line return?

s.lower().replace(';', ',').replace(' - ', ' ').strip()

Options:
A. 2024, september 14, error log entry: none.
B. 2024,september 14,error log entry:none.
C. 2024, september 14, error log entry : none.
D. 2024, september 14 error log entry none.

Correct option: A

Explanation:

- s.lower() converts the entire string to lowercase:
  "2024, september 14; error - log entry: none."
- .replace(';', ',') replaces ; with ,:
  "2024, september 14, error - log entry: none."
- .replace(' - ', ' ') replaces ' - ' (spaces around the dash) with a single space:
  "2024, september 14, error log entry: none."
- .strip() removes any leading/trailing whitespaces (though there aren't any here).

Final output:
"2024, september 14, error log entry: none."

Question 29:

Consider the tuple tup = (6, 14, 21, 28, 35). What is the output of the following code?

result = tup[1] + tup[-2]

print(result)

The output is: ______

Options:
A. 41
B. 49
C. 42
D. 43

Correct Option: C

Explanation:

- tup[1] is 14 (the second element).
- tup[-2] refers to the second-last element, which is 28.
- 14 + 28 = 42.
  So, the output is 42.

Question 30:

Consider the following Python program:

```
for _ in range(int(input())):
    s = input()
    if len(s) % 2 == 0:
        if sorted(s[:len(s)//2]) == sorted(s[len(s)//2:]):
            print('YES')
        else:
            print('NO')
    else:
        if sorted(s[:len(s)//2]) == sorted(s[(len(s)//2) + 1:]):
            print('YES')
        else:
            print('NO')
```

Given the input:

```
6
gaga
abcde
rotor
xyzxy
abbaab
ababc
```

What will the program output for each string?

Options:
A. YES, NO, YES, YES, NO, NO
B. NO, YES, NO, NO, YES, YES

C. YES, YES, NO, YES, NO, NO
D. NO, NO, YES, YES, YES, NO

Correct option: A

Explanation:
This program checks whether the first half and the second half of the string (ignoring the middle character if odd length) are anagrams (i.e., have the same characters in any order).

- gaga: even length, first half ga, second half ga → sorted equal → YES
- abcde: odd length, halves ab and de → sorted unequal → NO
- rotor: odd length, halves ro and or → sorted equal → YES
- xyzxy: odd length, halves xy and xy → sorted equal → YES
- abbaab: even length, halves abb and aab → sorted unequal → NO
- ababc: odd length, halves ab and bc → sorted unequal → NO

The outputs are: YES, NO, YES, YES, NO, NO.