# Test Name: Python Data Science & AI Intern

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Section Name | Unique Identifier | Problem Name | Type | Remarks |
| Django Project | 5dro83 | Django: Contact Manager Application | Project Based : BackEnd |  |
| Django Project | r0n53v | Django: Student Management API | Project Based : BackEnd |  |
| Data Processing and EDA | dx9vo1 | Olympics Analysis | Data Science |  |
| Data Processing and EDA | n31low | Data Prep Odyssey: Insurance Analytics | Data Science |  |
| AI/ML/DS | 5dldyn | Binary Classification: Deposit Subscriptions | DataScience Jupyter NoteBook |  |
| AI/ML/DS | 3ada14 | Binary Classification: Loan Approval | DataScience Jupyter NoteBook |  |
| AI/ML/DS | d9emx1 | Predicting Cyber Attack Incidents | DataScience Jupyter NoteBook |  |
| Python Coding | 638v5 | Word Length | Coding |  |
| Python Coding | 35oq4 | Algorithms: Numbered Lists | Coding |  |
| Python Coding | 84xg9n | Find the odd man out | Coding |  |
| Python Coding | ooevw | Member or Not | Coding |  |
| Python Coding | lbnp4 | Longest Word | Coding |  |

# Details

### Member or Not

Slug: ooevw

Type: Coding

Write a program that takes a list of values X (i.e., a number, string, etc) and a list of values A, and print true for each index in A if it is a member of X, and false otherwise.Note that this is exactly what the in operator does, but for the sake of the exercise, you should pretend Python did not have this operator.Input FormatThe first line contains an integer T, denoting the length of the list X.The following T lines contain the values of X.The next line will contain an integer, N, indicating the number of elements in the list A.The next N lines will contain the elements of the list A.Output FormatPrint true for each index in A if it is a member of X, and false otherwise.Evaluation ParametersSample Input2  
hello  
bug  
3  
bug  
in  
production  
Sample Outputtrue  
false  
false  
ExplanationElements of list X [hello, bug]Elements of lis A [bug, in, production]As only bug from list A is present in list X, so true has been printed only for 0th index.

Correct option: None

### Word Length

Slug: 638v5

Type: Coding

You are given a list of words, your task is to find the lengths of the corresponding words.Input FormatThe first line will contain an integer, N, representing the number of words.The next N lines will contain a word.Constraints1 <= N <= 101 <= length of each word <= 10Output FormatPrint the length of each word on a separate line.Evaluation ParametersInput3  
one  
two  
three  
Output3  
3  
5  
ExplanationHere 3, 3 and 5 corresponds to length of 'one', 'two' and 'three' respectively.

Correct option: None

### Longest Word

Slug: lbnp4

Type: Coding

Write a program that takes a list of words and returns the length of the longest one.Input FormatFirst line will contain an integer N, the number of words.The next N lines will have a word.Constraints1 <= N <= 501 <= length of word <= 100Output Formatprint length of the largest wordEvaluation ParametersInput #13  
which  
is  
largest  
Output #17  
ExplanationHere the word "largest" has the largest length which is 7

Correct option: None

### Algorithms: Numbered Lists

Slug: 35oq4

Type: Coding

As a beginner in Python programming, your task is to generate and print a list of 10 numbers within a given range, between a start point and an end point.The program should handle multiple test cases and check certain conditions before printing the range.Input FormatThe first line contains an integer T, the number of test cases.The next T lines each contain two integers: a start point and an end point.ConstraintsThe start point should NOT be less than 1; else print Out of RangeThe end point should NOT exceed 100; else print Out of RangeThe difference between the start point and end point (inclusive of both) should always be equal to 10; else print Difference Not in RangeThe limits constraint should be checked before the difference constraint.Output FormatFor each test case, print a list containing exactly 10 integers between the given start and end if the conditions are met.If any condition is violated, print the corresponding error message.Evaluation ParametersSample Input 13  
1 9  
22 31  
90 103  
Sample Output 1Difference Not in Range  
[22, 23, 24, 25, 26, 27, 28, 29, 30, 31]  
Out of Range  
Explanation:For the first test case, the difference between 1 and 9 is 9, which is not valid (the inclusive difference must be 10).For the second test case, the difference between 22 and 31 is 10, so it prints the range from 22 to 31.For the third test case, the end point 103 exceeds the maximum value of 100, so it prints Out of Range.Sample Input 22  
2 4  
3 12  
Sample Output 2Difference Not in Range  
[3, 4, 5, 6, 7, 8, 9, 10, 11, 12]  
Explanation:For the first test case, the difference between 2 and 4 is 3, which is not valid (the inclusive difference must be 10).For the second test case, the difference between 3 and 12 is 10, so it prints the range from 3 to 12.

Correct option: None

### Find the odd man out

Slug: 84xg9n

Type: Coding

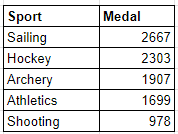
You are given two arrays containing user IDs. array\_1 has N user IDs and array\_2 has all the user IDs i.e array\_1 IDs and additional IDs. Your task is to find out these additional IDs.Input FormatFirst line will contain space separated integers of array\_1 ID.Next line will contain space separated integers of C.Constraints1<=array1 ID, array\_2 ID<=102.array\_2 ID>array\_1 IDOutput FormatPrint the additional IDs in space separated format.Evaluation ParametersSample Input1 4 7 3  
1 2 3 4 5 6 7  
Sample Output2 5 6  
Explanation2, 5, and 6 are not present in the array\_1 ID so that means its additional ID

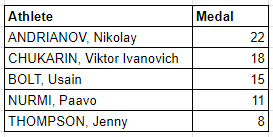
Correct option: None

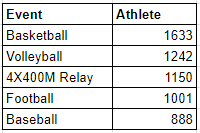
### Olympics Analysis

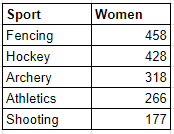
Slug: dx9vo1

Type: Data Science









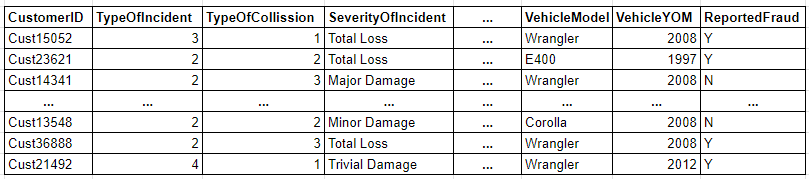
As the world gears up for another thrilling Olympic Games, SportsStat Inc. takes on the monumental task of sifting through a dataset spanning the modern Olympics' storied history. In a quest to celebrate excellence and inspire the next generation, analysts are poised to uncover the most decorated sports, athletes, and events that have captivated audiences worldwide. This analysis not only honors past achievements but also shines a light on the evolving landscape of international sports competition.Your task is to do the following :Identify the top 5 sports that have awarded the most medals.Identify the top 5 athletes who have won the most gold medals.Identify the top 5 Olympic events that have attracted the largest number of athletes. (take unique values for only task3)Identify the top 5 sports with the most number of female gold medalists.DatasetYou can use the given dataset for the task.The dataset contains 9 attributes: Year, City, Sport, Discipline, Athlete, Country, Gender, Event, and Medal.You can access the dataset at the following location :/data/training/dataset.csv  
OutputSave your file at the given location with the following data:1) The top 5 sports that have awarded the most medals./code/task1.csv  
2) The top 5 athletes who have won the most gold medals./code/task2.csv  
3) The top 5 Olympic events that have attracted the largest number of athletes./code/task3.csv  
4) The top 5 sports with the most number of female gold medalists./code/task4.csv  
Sample Output1) The top 5 sports that have awarded the most medals.2) The top 5 athletes who have won the most gold medals.3) The top 5 Olympic events that have attracted the largest number of athletes.4) The top 5 sports with the most number of female gold medalists.NOTE:Make sure your Output files have the same column names as those provided in the Sample Output.Ignore the validation dataset and evaluation dataset. The datasets for this problem are provided in the training dataset.

Correct option: None

### Data Prep Odyssey: Insurance Analytics

Slug: n31low

Type: Data Science



In the dynamic realm of insurance, where precision and accuracy shape the future, GlobalAssure, a pioneering insurance corporation, embarks on a mission to refine its vast datasets. Tasked with transforming a cluttered insurance dataset into a model-ready format, analysts stand at the forefront of innovation. This dataset, marred by inconsistencies and missing values, holds the secrets to predicting future claims and understanding risk factors. Your objective is to cleanse, preprocess, and prepare this data, unlocking the potential for groundbreaking insights.Your task is to do the following :Remove these columns from the dataset: VehicleID, IncidentAddress, Country, InsurancePolicyNumber, DateOfPolicyCoverage, Country, DateOfIncident.Standardize missing values by replacing '?', '???', '-1', '-5', 'MISSINGVALUE', and 'mode' with null values.Eliminate any column that has more than 30% null values to ensure data quality.For categorical variables, fill in missing values with the most frequent value (mode) within each column.For numerical variables, impute missing entries with the column's average value (mean).Transform the TypeOfIncident column by mapping:'Single Vehicle Collision' -> 1'Multi-vehicle Collision' -> 2'Parked Car' -> 3'Vehicle Theft' -> 4Recode the TypeOfCollission column values as follows:'Side Collision' -> 1'Rear Collision' -> 2'Front Collision' -> 3DatasetYou can use the given dataset for the given tasks.The dataset contains 42 attributes: CustomerID, DateOfIncident, TypeOfIncident, TypeOfCollission, SeverityOfIncident, AuthoritiesContacted, IncidentState, IncidentCity, IncidentAddress, IncidentTime, NumberOfVehicles, PropertyDamage, BodilyInjuries, Witnesses, PoliceReport, AmountOfTotalClaim, AmountOfInjuryClaim, AmountOfPropertyClaim, AmountOfVehicleDamage, InsuredAge, InsuredZipCode, InsuredGender, InsuredEducationLevel, InsuredOccupation, InsuredHobbies, CapitalGains, CapitalLoss, Country, InsurancePolicyNumber, CustomerLoyaltyPeriod, DateOfPolicyCoverage, InsurancePolicyState, Policy\_CombinedSingleLimit, Policy\_Deductible, PolicyAnnualPremium, UmbrellaLimit, InsuredRelationship, VehicleID, VehicleMake, VehicleModel, VehicleYOM, and ReportedFraud.You can access the dataset at the following location:/data/training/dataset.csv  
OutputSave your file at the given location with the following data:/code/output.csv  
Sample Output

Correct option: None

### Django: Student Management API

Slug: r0n53v

Type: Project Based : BackEnd

Implement a simple REST API to manage a collection of student records.Each student record is a JSON entry with the following keys:id: The unique ID of the student. (Integer)first\_name: The name of the student. (String)last\_name: The last name of the student. (String)date\_of\_birth: The date of birth of the student. (Date)grade: The grade of the student. (Integer)phone: The phone number of the student. (String)email: The email of the student. (String)Here is an example of a student data JSON object:{  
 "id": 1,  
 "first\_name": "John",  
 "last\_name": "Doe",  
 "date\_of\_birth": "2000-02-08",  
 "grade": 7,  
 "phone": "1234567890",  
 "email": "john@email.com"  
}  
There is already an implementation of the Student model. Implement a REST service that exposes the /students endpoint to manage the collection of student records as follows:POST request to /students:creates a new student recordexpects a JSON student object without an id property as a body payload. You can assume that the given object is always valid.adds the given student object to the collection of student records and assigns a unique integer id to it. The first created student record must have id 1, the second one 2, and so on.the response code is 201, and the response body is the created student recordGET request to /students:returns a collection of all student recordsthe response code is 200, and the response body is an array of all student records ordered by their ids in increasing order.GET request to /students/<id>:returns a student record with the given idif the matching student exists, the response code is 200 and the response body is the matching student recordif there is no student with the given id, the response code is 404 with an error message Student doesn't exist.PATCH request to /students/<id>:updates a student record with the given idexpects a JSON student object without an id property as a body payload. You can assume that the given object is always valid.on success, the response code is 200, and the response body is the updated student recordif there is no student with the given id in the collection, the response code is 404 with an error message Student doesn't exist.Complete the given project so that it passes all the test cases when running the provided unit tests. The project by default supports the use of the SQLite3 database. Implement the POST request to /students/ first because testing the other methods requires POST requests to work correctly.Example requests and responsesPOST request to /studentsRequest body:{  
 "first\_name": "John",  
 "last\_name": "Doe",  
 "date\_of\_birth": "2000-02-08",  
 "grade": 7,  
 "phone": "1234569079",  
 "email": "john@email.com"  
}  
The response code is 201, and when converted to JSON, the response body is:{  
 "id": 1,  
 "first\_name": "John",  
 "last\_name": "Doe",  
 "date\_of\_birth": "2000-02-08",  
 "grade": 7,  
 "phone": "1234569079",  
 "email": "john@email.com"  
}  
This adds a new object to the collection with the given properties and id 1.GET request to /studentsAssuming these are all of the objects in the collection, the response code is 200, and when converted to JSON, the response body is:[  
 {  
 "id": 1,  
 "first\_name": "John",  
 "last\_name": "Doe",  
 "date\_of\_birth": "2000-02-08",  
 "grade": 7,  
 "phone": "+1234569079",  
 "email": "john@email.com"  
 },  
 {  
 "id": 2,  
 "first\_name": "James",  
 "last\_name": "Smith",  
 "date\_of\_birth": "2004-07-25",  
 "grade": 5,  
 "phone": "7209374654",  
 "email": "smith@email.com"  
 }  
]  
GET request to /students/1Assuming that the object with id 1 exists, then the response code is 200 and the response body, when converted to JSON, is as follows:{  
 "id": 1,  
 "first\_name": "John",  
 "last\_name": "Doe",  
 "date\_of\_birth": "2000-02-08",  
 "grade": 7,  
 "phone": "1234567890",  
 "email": "john@email.com"  
}  
If an object with id 1 doesn't exist, then the response code is 404 with an error message "Student doesn't exist"{  
 "message": "Student doesn't exist"  
}  
PATCH request to /students/1Request body:{  
 "first\_name": "John",  
 "last\_name": "Doe",  
 "date\_of\_birth": "2010-05-01",  
 "grade": 9,  
 "phone": "1234325679",  
 "email": "jhon.doe@email.com"  
}  
Assuming that the object with id 1 exists, then the response code is 200 and the response body, when converted to JSON, is as follows:{  
 "id": 1,  
 "first\_name": "John",  
 "last\_name": "Doe",  
 "date\_of\_birth": "2010-05-01",  
 "grade": 9,  
 "phone": "1234325679",  
 "email": "jhon.doe@email.com"  
}  
If an object with id 1 does not exist, then the response code is 404 with an error message "Student doesn't exist".

Correct option: None

### Django: Contact Manager Application

Slug: 5dro83

Type: Project Based : BackEnd

Django API for User and Contact ManagementImplement a simple REST API to manage a collection of user and contact records.Each user and contact record is a JSON entry with the following keys:Userid: The unique ID of the user. (Integer)username: The username of the user. (String)email: The email of the user. (String)first\_name: The first name of the user. (String)last\_name: The last name of the user. (String)date\_joined: The date the user joined. (DateTime)Contactid: The unique ID of the contact. (Integer)creator: The ID of the user who created the contact. (Integer)first\_name: The first name of the contact. (String)last\_name: The last name of the contact. (String)email: The email of the contact. (String)phone\_number: The phone number of the contact. (String)address: The address of the contact. (String)date\_added: The date the contact was added. (DateTime)API EndpointsPOST request to /users/Creates a new user record.Expects a JSON user object without an id property as a body payload. You can assume that the given object is always valid.Adds the given user object to the collection of user records and assigns a unique integer id to it. The first created user record must have id 1, the second one 2, and so on.The response code is 201, and the response body is the created user record.Example request body:{  
 "username": "john\_doe",  
 "email": "john@example.com",  
 "first\_name": "John",  
 "last\_name": "Doe"  
}  
Example response:{  
 "id": 1,  
 "username": "john\_doe",  
 "email": "john@example.com",  
 "first\_name": "John",  
 "last\_name": "Doe",  
 "date\_joined": "2023-05-01T10:00:00Z"  
}  
POST request to /contacts/Creates a new contact record.Expects a JSON contact object without an id property as a body payload. You can assume that the given object is always valid.Adds the given contact object to the collection of contact records and assigns a unique integer id to it. The first created contact record must have id 1, the second one 2, and so on.The response code is 201, and the response body is the created contact record.Example request body:{  
 "creator": 1,  
 "first\_name": "Jane",  
 "last\_name": "Doe",  
 "email": "jane@example.com",  
 "phone\_number": "+1234567890",  
 "address": "123 Main St"  
}  
Example response:{  
 "id": 1,  
 "creator": 1,  
 "first\_name": "Jane",  
 "last\_name": "Doe",  
 "email": "jane@example.com",  
 "phone\_number": "+1234567890",  
 "address": "123 Main St",  
 "date\_added": "2023-05-01T10:00:00Z"  
}  
GET request to /contacts/Returns a collection of all contact records.The response code is 200, and the response body is an array of all contact records ordered by their ids in increasing order.Example response:[  
 {  
 "id": 1,  
 "creator": 1,  
 "first\_name": "Jane",  
 "last\_name": "Doe",  
 "email": "jane@example.com",  
 "phone\_number": "+1234567890",  
 "address": "123 Main St",  
 "date\_added": "2023-05-01T10:00:00Z"  
 },  
 {  
 "id": 2,  
 "creator": 1,  
 "first\_name": "Alice",  
 "last\_name": "Smith",  
 "email": "alice@example.com",  
 "phone\_number": "+0987654321",  
 "address": "456 Elm St",  
 "date\_added": "2023-05-02T11:00:00Z"  
 }  
]  
PATCH request to /contacts/<id>/Updates a contact record with the given id.Expects a JSON contact object without an id property as a body payload. You can assume that the given object is always valid.On success, the response code is 200, and the response body is the updated contact record.If there is no contact with the given id in the collection, the response code is 404 with an error message Contact with id: <id> does not exist.If no Id is provided, the response code is 400 with an error message No id providedExample request body:{  
 "first\_name": "Jane",  
 "last\_name": "Doe",  
 "email": "jane.doe@example.com",  
 "phone\_number": "+1234567890",  
 "address": "123 Main St, Apt 4"  
}  
Example response for a valid contact:{  
 "id": 1,  
 "creator": 1,  
 "first\_name": "Jane",  
 "last\_name": "Doe",  
 "email": "jane.doe@example.com",  
 "phone\_number": "+1234567890",  
 "address": "123 Main St, Apt 4",  
 "date\_added": "2023-05-01T10:00:00Z"  
}  
Example response for an invalid contact:{  
 "error": "Contact with id: <id> does not exist"  
}  
DELETE request to /contacts/<id>/Deletes a contact record with the given id.If the contact exists, the response code is 200 and the response body contains a success message.If there is no contact with the given id, the response code is 404 with an error message Contact with id: <id> does not exist.If no Id is provided, the response code is 400 with an error message No id providedExample response for a valid contact:{  
 "message": "Contact with id: <id> was deleted successfully"  
}  
Example response for an invalid contact:{  
 "error": "Contact with id: <id> does not exist"  
}  
Example of no id provided{  
 "error": "No id provided"  
}

Correct option: None

### Predicting Cyber Attack Incidents

Slug: d9emx1

Type: DataScience Jupyter NoteBook

With the rapid rise in digital transactions globally, the threat of cyber attacks, especially those targeting financial information, has significantly increased. Hackers often exploit vulnerabilities associated with digital payment systems, and even minimal data like a phone number linked to a bank account can be sufficient for initiating malicious activities. To proactively defend against these cyber threats, developing a predictive model that can analyze incident logs and identify patterns indicating potential cyber-attacks is crucial.The task is to classify whether an incident is a potential malicious offense or not based on anonymized logging parameters.Training DatasetYou can use the training dataset to train the model.The training dataset contains 19 attributes: ID, INCIDENT\_ID, DATE, X\_1, X\_2, X\_3, X\_4, X\_5, X\_6, X\_7, X\_8, X\_9, X\_10, X\_11, X\_12, X\_13, X\_14, X\_15, and MALICIOUS\_OFFENSE, where MALICIOUS\_OFFENSE shows whether an incident is a potential malicious offense or not.You can access the training dataset at the following location :train.csv  
Testing DatasetYou can use the testing dataset for the prediction.The testing dataset contains 18 attributes: ID, INCIDENT\_ID, DATE, X\_1, X\_2, X\_3, X\_4, X\_5, X\_6, X\_7, X\_8, X\_9, X\_10, X\_11, X\_12, X\_13, X\_14, and X\_15.You can access the testing dataset at the following location :test.csv  
OutputSave your file at the given location with the following data:output.csv  
Sample OutputID MALICIOUS\_OFFENSE  
M#11 1  
M#12 1  
M#13 1  
M#14 1  
M#15 0  
EvaluationYour solution will be accepted if the classification accuracy exceeds 70%.Positive Class: 1NOTE: You must also perform data cleaning and pre-processing to test data.

Correct option: None

### Binary Classification: Loan Approval

Slug: 3ada14

Type: DataScience Jupyter NoteBook

A financial institution is looking to enhance its loan approval system by leveraging machine learning to make faster and more accurate decisions. The task is to build a model that predicts whether a loan should be approved based on an applicant's profile, including details like income, employment status, credit history, and other relevant factors. The goal is to reduce processing time while maintaining a high level of accuracy in loan approval decisions.The task is to classify whether an applicant loan is Approved (1) or Not Approved (0) based on the applicant data.Training DatasetYou can use the training dataset to train the model.The training dataset contains 24 attributes:ID, ApplicationDate, EmploymentStatus, EducationLevel, MaritalStatus, HomeOwnershipStatus, LoanPurpose, AnnualIncome, BaseInterestRate, LoanAmount, NetWorth, MonthlyLoanPayment, CreditScore, Age, Experience, LoanDuration, CreditCardUtilizationRate, NumberOfCreditInquiries, JobTenure, NumberOfOpenCreditLines, CheckingAccountBalance, DebtToIncomeRatio, BankruptcyHistory, and LoanApproved where LoanApproved shows whether an applicant Loan is Approved or Not.You can access the training dataset at the following location:train.csv  
Testing DatasetYou have to use the testing dataset for the predictions.The testing dataset contains 23 attributes: ID, ApplicationDate, EmploymentStatus, EducationLevel, MaritalStatus, HomeOwnershipStatus, LoanPurpose, AnnualIncome, BaseInterestRate, LoanAmount, NetWorth, MonthlyLoanPayment, CreditScore, Age, Experience, LoanDuration, CreditCardUtilizationRate, NumberOfCreditInquiries, JobTenure, NumberOfOpenCreditLines, CheckingAccountBalance, DebtToIncomeRatio, and BankruptcyHistory.Once you train the model using Training Dataset, you need to generate the predictions for Testing Dataset and save it as output.csv file. Refer Sample Output for output.csv file format.You can access the testing dataset at the following location:test.csv  
OutputSave your file at the given location with the following data:output.csv  
Sample OutputID LoanApproved  
5349 0  
340 1  
13592 0  
8154 0  
16346 0  
Sample Output is given just for illustrative purposes.EvaluationYour solution will be accepted if the classification accuracy on test dataset exceeds 70%.Positive Class: 1 (Indicates Loan Approved)Negative Class: 0 (Indicates Loan Not Approved)NOTE: You must also clean and pre-process the test data.

Correct option: None

### Binary Classification: Deposit Subscriptions

Slug: 5dldyn

Type: DataScience Jupyter NoteBook

A prominent bank seeks to optimize its marketing campaigns for term deposit subscriptions. The bank uses telemarketing to approach potential clients, but the success rate is low. With a large dataset of customer information and past marketing outcomes, your task is to build a machine learning model to predict whether a client will subscribe to a term deposit based on historical campaign data. This will enhance targeting and improve subscription rates.The task is to classify whether a client subscribes for term deposit (yes) or (no).Training DatasetYou can use the training dataset to train the model.The training dataset contains 18 attributes: ID, age, job, marital, education, default, balance, housing, loan, contact, day, month, duration, campaign, pdays, previous, poutcome, and y, where y shows whether a client subscribes for term deposit or not.You can access the training dataset at the following location :train.csv  
Testing DatasetYou have to use the testing dataset for the prediction.The testing dataset contains 17 attributes: ID, age, job, marital, education, default, balance, housing, loan, contact, day, month, duration, campaign, pdays, previous, and poutcome.You can access the testing dataset at the following location :test.csv  
OutputSave your file at the given location with the following data:output.csv  
Sample OutputID y  
20261 yes  
34966 no  
40084 no  
16799 yes  
23775 no  
EvaluationYour solution will be accepted if the classification accuracy on test dataset exceeds 70%.Positive Class: yes (client subscribes for term deposit)Negative Class: no (client will not subscribes for term deposit)NOTE: You must also clean and pre-process the testing data before testing the model.

Correct option: None