Applied Artificial Intelligence

Index

Sr.No	Title	Date	Sign
1	Design a bot using AIML.		
2	Design an Expert system using AIML.		
3	Implement Bayes Theorem using Python.		
4	Implement Conditional Probability and Joint Probability using Python.		
5	Write a program to implement Rule based system (Prolog).		
6	Design a Fuzzy based application using Python/R.		
7	[A] Write an application to simulate supervised learning model. [B] Write an application to simulate unsupervised learning model.		
8	Write an application to implement clustering algorithm.		
9	Write an application to implement Support Vector Machine Algorithm.		
10	Simulate artificial neural network model with both feedforward and backpropagation approach.		

Practical No: 1

Aim: Design a bot using AIML.

Code:

Step 1: Create an XML file.

Open the notepad, write the following code and save it as *std-startup.xml*

```
<aiml version="1.0.1" encoding="UTF-8">
  <category>
   <pattern>LOAD AIML B</pattern>
   <template>
    <learn>basic_chat.aiml</learn>
   </template>
  </category>
</aiml>
```

Step 2: Create an aiml file.

Open the notepad, write the following code and save it as *basic_chat.aiml.*

```
<aiml version="1.0.1" encoding="UTF-8">
 <category>
   <pattern>HELLO</pattern>
   <template>
    Well, hello!
   </template>
 </category>
```

```
<category>
  <pattern>WHAT ARE YOU</pattern>
  <template>
    I'm a bot, silly!
  </template>
</category>
<category>
  <pattern>MY NAME IS *</pattern>
  <template>
    <set name = "username"><star/></set> is also my favourite.
  </template>
</category>
<category>
  <pattern>I LIKE *</pattern>
  <template>
    <set name = "liking"><star/></set> is a nice name.
  </template>
</category>
<category>
  <pattern>MY DOG NAME IS *</pattern>
  <template>
    <set name = "dog"><star/></set> THAT IS INTERESTING THAT YOU HAVE A DOG NAMED.
  </template>
</category>
<category>
  <pattern>Bye *</pattern>
  <template>
    <set name = "Bye!!!"><star/></set> Thanks for talking with me.
```

```
   </template>
  </category>
</aiml>
```

Step 3: Install aiml package through command prompt.

"pip install aiml"

or

"pip3 install aiml"

Step 4: Create chatbot.py file.

```
import aiml
kernel = aiml.Kernel()
kernel.learn("std-startup.xml")
kernel.respond("load aiml b")
while True:
  message = input("Enter your message to bot: ")
  if message == "quit":
    break
  else:
    bot_response = kernel.respond(message)
    print(bot_response)
```

Output:

```
Loading std-startup.xml...done (0.02 seconds)
Loading basic_chat.aiml...done (0.00 seconds)
Enter your message to the bot: hello
Well, hello!
Enter your message to the bot: what are you
I'm a bot, silly!
Enter your message to the bot: my dog name is sam
sam THAT IS INTERESTING THAT YOU HAVE A DOG NAMED.
Enter your message to the bot: bye bot
bot Thanks for talking with me.
Enter your message to the bot: quit
```

Practical No: 2

Aim: Design an Expert system using AIML.

Code:

Step 1: Create an XML file.

Open the notepad, write the following code and save it as *std-startup1.xml*

```
<aiml version="1.0.1" encoding="UTF-8">
  <category>
   <pattern>LOAD AIML B</pattern>
   <template>
    <learn>basic_chat1.aiml</learn>
   </template>
 </category>
</aiml>
```

Step 2: Create an aiml file.

Open the notepad, write the following code and save it as *basic_chat1.aiml*

```
<aiml version="1.0.1" encoding="UTF-8">
 <category>
   <pattern>HELLO</pattern>
   <template>
    WHAT WOULD YOU LIKE TO DISCUSS? :HEALTH,MOVIES
   </template>
```

```
  </category>
  <category>
    <pattern>MOVIES</pattern>
    <template>
      YES <set name = "topic">MOVIES</set>
    </template>
  </category>
<category>
    <pattern>HEALTH</pattern>
    <template>
      YES <set name = "topic">HEALTH</set>
    </template>
  </category>
  <topic name = "MOVIES">
    <category>
      <pattern>*</pattern>
      <template>
        DO YOU LIKE COMEDY MOVIES?
      </template>
    </category>
    <category>
      <pattern>YES</pattern>
      <template>
        I TOO LIKE COMEDY MOVIES!
      </template>
    </category>
    <category>
      <pattern>NO</pattern>
```

```
    <template>
      BUT I LIKE COMEDY MOVIES
    </template>
  </category>
</topic>
<topic name = "HEALTH">
  <category>
    <pattern>*</pattern>
    <template>
      DO YOU HAVE FEVER?
    </template>
  </category>
  <category>
    <pattern>YES</pattern>
    <template>
      PLEASE TAKE MEDICINES AND PROPER REST
    </template>
  </category>
  <category>
    <pattern>NO</pattern>
    <template>
      GO OUT FOR A WALK AND LISTEN MUSIC
    </template>
  </category>
</topic>
<category>
  <pattern>NICE TALKING TO YOU</pattern>
  <template>
```

```
    SAME HERE...!
   </template>
  </category>
</aiml>
```

Step 3: Install aiml package through command prompt.

"pip install aiml"

or

"pip3 install aiml"

Step 4: Create chatbot.py file.

```
import aiml
kernel = aiml.Kernel()
kernel.learn("std-startup1.xml")
kernel.respond("load aiml b")
while True:
  message = input("Enter your message to bot: ")
  if message == "quit":
    break
  else:
    bot_response = kernel.respond(message)
    print(bot_response)
```

Output:

```
Loading basic_chat1.aiml...done (0.00 seconds)
Enter a message for chatbot: hello
WHAT WOULD YOU LIKE TO DISCUSS? :HEALTH,MOVIES
Enter a message for chatbot: health
YES HEALTH
Enter a message for chatbot: i am feeling tired
DO YOU HAVE FEVER?
Enter a message for chatbot: no
GO OUT FOR A WALK AND LISTEN MUSIC
Enter a message for chatbot: movies
YES MOVIES
Enter a message for chatbot: i love movies
DO YOU LIKE COMEDY MOVIES?
Enter a message for chatbot: yes
I TOO LIKE COMEDY MOVIES!
Enter a message for chatbot: nice talking to you
SAME HERE...!
Enter a message for chatbot: quit
```

Practical No: 3

Aim: Implement Bayes Theorem using Python.

Code:

```
def bayes_theorem(p_a, p_b_given_a, p_b_given_not_a):
    not_a = 1 - p_a
    p_b = p_b_given_a * p_a + p_b_given_not_a * not_a
    p_a_given_b = (p_b_given_a * p_a) / p_b
    return p_a_given_b

p_a = 0.0002
p_b_given_a = 0.85
p_b_given_not_a = 0.05

result = bayes_theorem(p_a, p_b_given_a, p_b_given_not_a)
print('P(A|B): %.3f%%' % (result * 100))
```

Output:

```
P(A|B): 0.339%
```

Practical No: 4

Aim: Implement Conditional Probability and Joint Probability using Python

Code:

```
import enum, random
class Kid(enum.Enum):
   BOY = 0
   GIRL = 1

def random_kid() -> Kid:
   return random.choice([Kid.BOY, Kid.GIRL])
both_girls = 0
older_girl = 0
either_girl = 0
random.seed(0)

for _ in range(10000):
   younger = random_kid()
   older = random_kid()
   if older == Kid.GIRL:
      older_girl += 1
   if older == Kid.GIRL and younger == Kid.GIRL:
      both_girls += 1
   if older == Kid.GIRL or younger == Kid.GIRL:
      either_girl += 1

print("Older girl: ",older_girl)
```

```
print("Both girls: ", both_girls)
print("Either girl: ", either_girl)
print("P(both | older): ", both_girls/older_girl)
print("P(both | either): ",both_girls/either_girl)
```

Output:

```
Older girl:  4937
Both girls:  2472
Either girl:  7464
P(both | older):  0.5007089325501317
P(both | either):  0.3311897106109325
```

Practical No: 5

Aim: Write a program to implement Rule based system (Prolog).

Code:

```
   hypothesis(Disease),
   write('I believe that the patient have '),
   write(Disease),
   nl,
   write('TAKE CARE '),
   undo.

/*Hypothesis that should be tested*/
hypothesis(cold) :- cold, !.
hypothesis(flu) :- flu, !.
hypothesis(typhoid) :- typhoid, !.
hypothesis(measles) :- measles, !.
hypothesis(malaria) :- malaria, !.
hypothesis(unknown). /* no diagnosis*/

/*Hypothesis Identification Rules*/

cold :-
verify(headache),
verify(runny_nose),
verify(sneezing),
verify(sore_throat),
write('Advices and Sugestions:'),
```

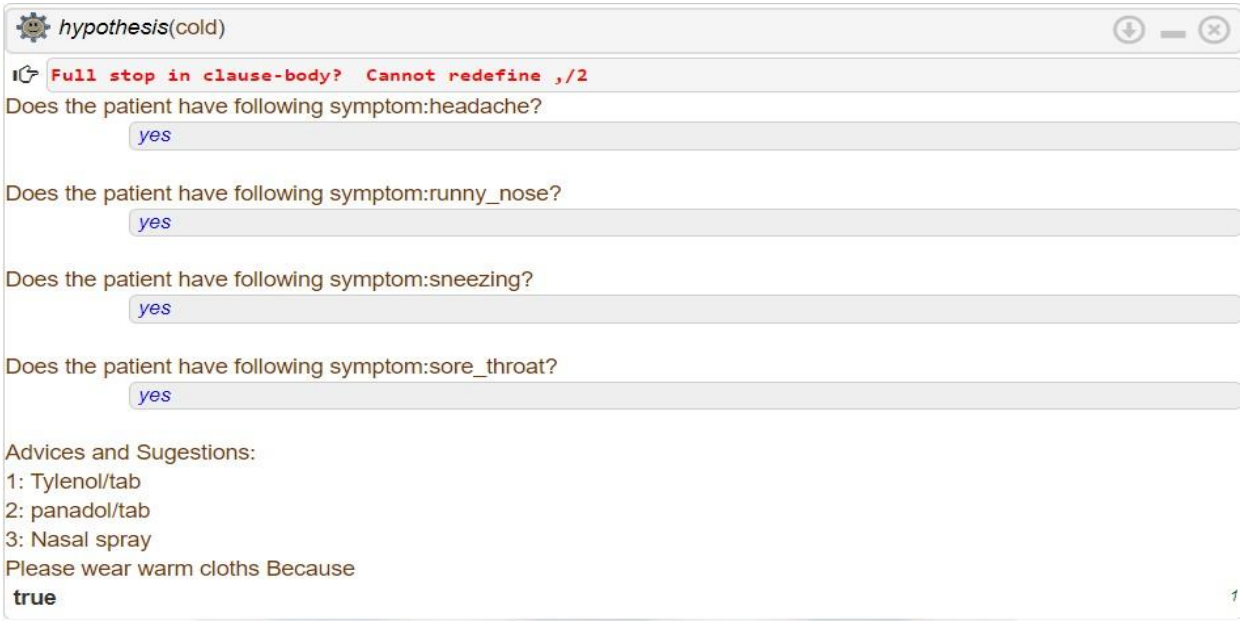
```
nl,
write('1: Tylenol/tab'),
nl,
write('2: panadol/tab'),
nl,
write('3: Nasal spray'),
nl,
write('Please wear warm cloths Because'),
nl.

flu :-
verify(fever),
verify(headache),
verify(chills),
verify(body_ache),
write('Advices and Sugestions:'),
nl,
write('1: Tamiflu/tab'),
nl,
write('2: panadol/tab'),
nl,
write('3: Zanamivir/tab'),
nl,
write('Please take a warm bath and do salt gargling Because'),
nl.

typhoid :-
verify(headache),
```

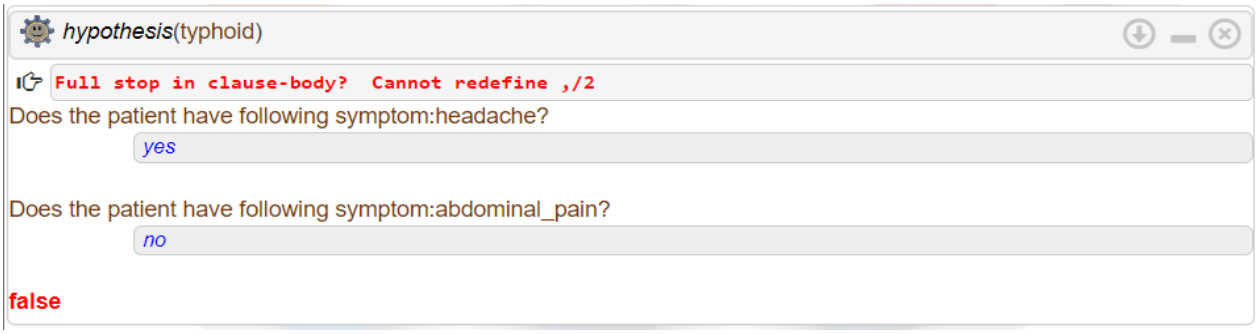
```
verify(abdominal_pain),
verify(poor_appetite),
verify(fever),
write('Advices and Sugestions:'),
nl,
write('1: Chloramphenicol/tab'),
nl,
write('2: Amoxicillin/tab'),
nl,
write('3: Ciprofloxacin/tab'),
nl,
write('4: Azithromycin/tab'),
nl,
write('Please do complete bed rest and take soft Diet Because'),
nl.

measles :-
verify(fever),
verify(runny_nose),
verify(rash),
verify(conjunctivitis),
write('Advices and Sugestions:'),
nl,
write('1: Tylenol/tab'),
nl,
write('2: Aleve/tab'),
nl,
write('3: Advil/tab'),
```

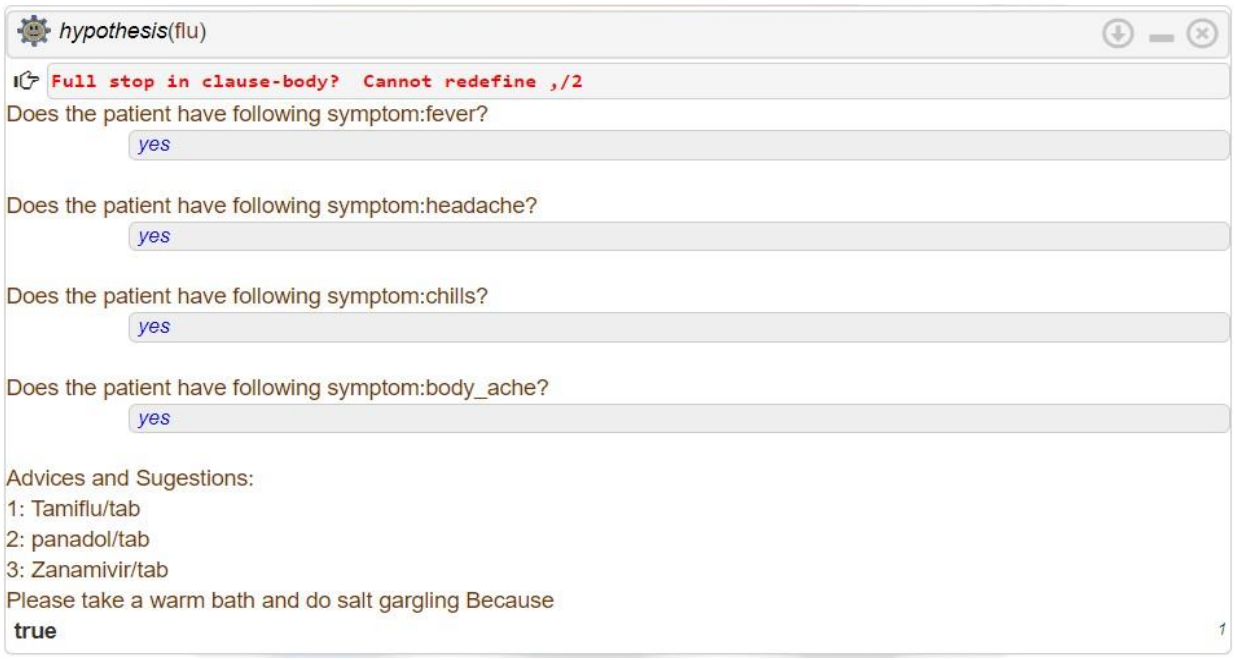
```
nl,
write('4: Vitamin A'),
nl,
write('Please Get rest and use more liquid Because'),
nl.

malaria :-
verify(fever),
verify(sweating),
verify(headache),
verify(nausea),
verify(vomiting),
verify(diarrhea),
write('Advices and Sugestions:'),
nl,
write('1: Aralen/tab'),
nl,
write('2: Qualaquin/tab'),
nl,
write('3: Plaquenil/tab'),
nl,
write('4: Mefloquine'),
nl,
write('Please do not sleep in open air and cover your full skin Because'),
nl.

/* how to ask questions */
ask(Question) :-
```

```
write('Does the patient have following symptom:'),
write(Question),
write('? '),
read(Response),
nl,
( (Response == yes ; Response == y)
->
assert(yes(Question)) ;
assert(no(Question)), fail).

:- dynamic yes/1,no/1.
/*How to verify something */
verify(S) :-
(yes(S)
->
true ;
(no(S)
->
fail ;
ask(S))).
/* undo all yes/no assertions*/
undo :- retract(yes(_)),fail.
undo :- retract(no(_)),fail.
undo.
```

Output:


hypothesis(cold)
Full stop in clause-body? Cannot redefine ,/2
Does the patient have following symptom:headache?
yes
Does the patient have following symptom:runny_nose?
yes
Does the patient have following symptom:sneezing?
yes
Does the patient have following symptom:sore_throat?
yes
Advices and Sugestions:
1: Tylenol/tab
2: panadol/tab
3: Nasal spray
Please wear warm cloths Because
true
1


hypothesis(typhoid)
Full stop in clause-body? Cannot redefine ,/2
Does the patient have following symptom:headache?
yes
Does the patient have following symptom:abdominal_pain?
no
false


hypothesis(flu)
Full stop in clause-body? Cannot redefine ,/2
Does the patient have following symptom:fever?
yes
Does the patient have following symptom:headache?
yes
Does the patient have following symptom:chills?
yes
Does the patient have following symptom:body_ache?
yes
Advices and Sugestions:
1: Tamiflu/tab
2: panadol/tab
3: Zanamivir/tab
Please take a warm bath and do salt gargling Because
true
1

Practical No: 6

Aim: Design a Fuzzy based application using Python/ R.

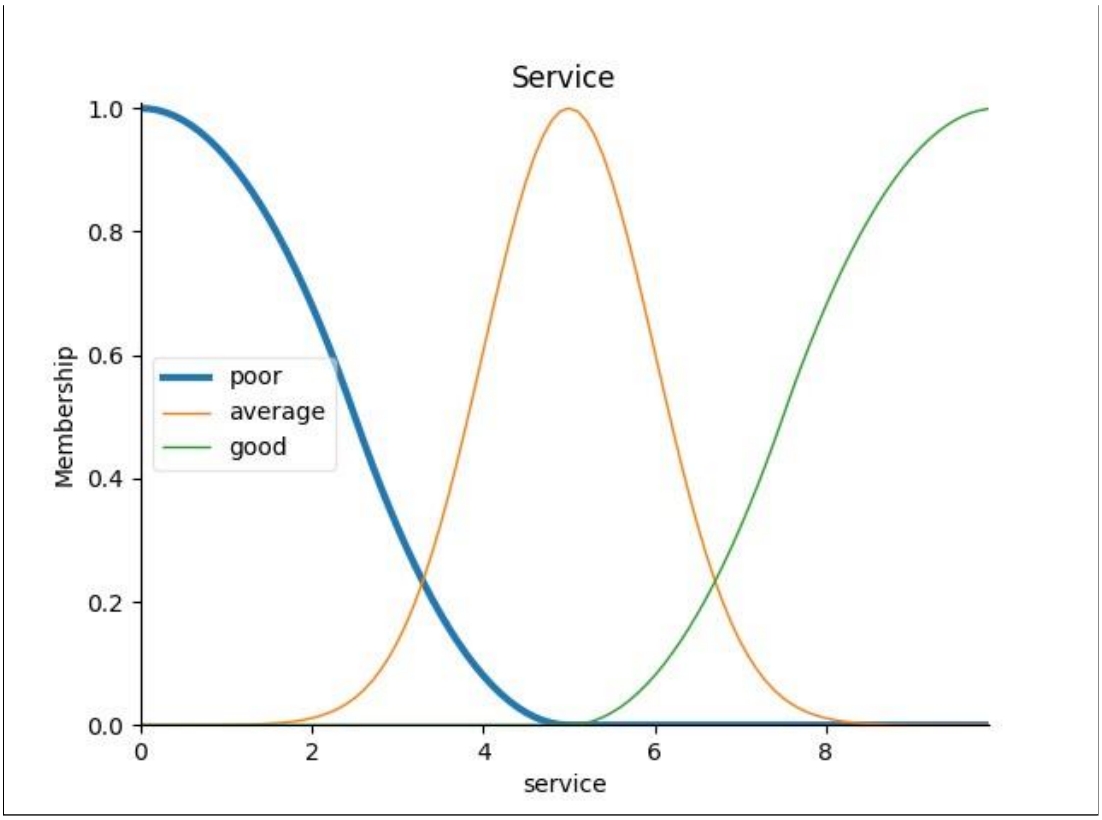
Code:

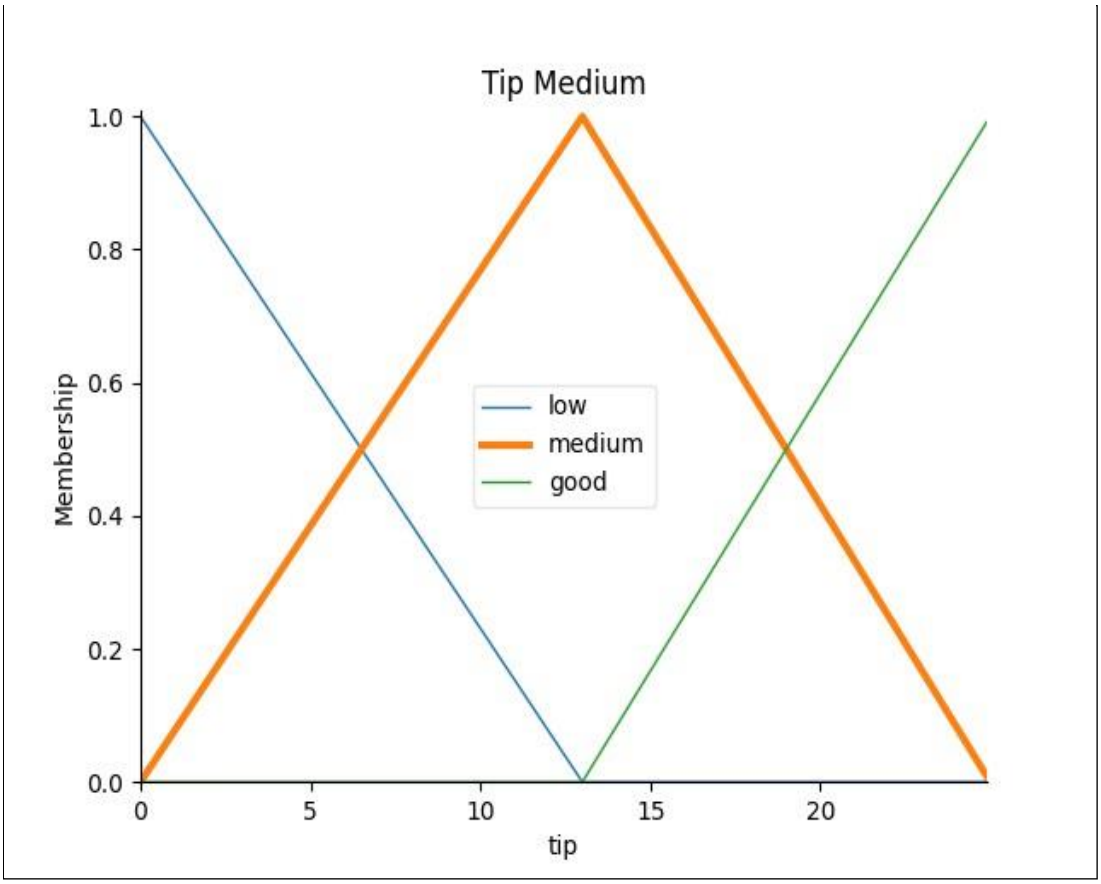
```
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt
from skfuzzy import control as ctrl
from mpl_toolkits.mplot3d import Axes3D

quality = ctrl.Antecedent(np.arange(0, 10, 0.1), 'quality')
service = ctrl.Antecedent(np.arange(0, 10, 0.1), 'service')
tip = ctrl.Consequent(np.arange(0, 25, 0.1), 'tip')

quality['poor'] = fuzz.zmf(quality.universe, 0,5)
quality['average'] = fuzz.gaussmf(quality.universe, 5,1)
quality['good'] = fuzz.smf(quality.universe, 5,10)

service['poor'] = fuzz.zmf(service.universe, 0,5)
service['average'] = fuzz.gaussmf(service.universe, 5,1)
service['good'] = fuzz.smf(service.universe, 5,10)

tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
tip['good'] = fuzz.trimf(tip.universe, [13, 25, 25])

quality['average'].view()
```

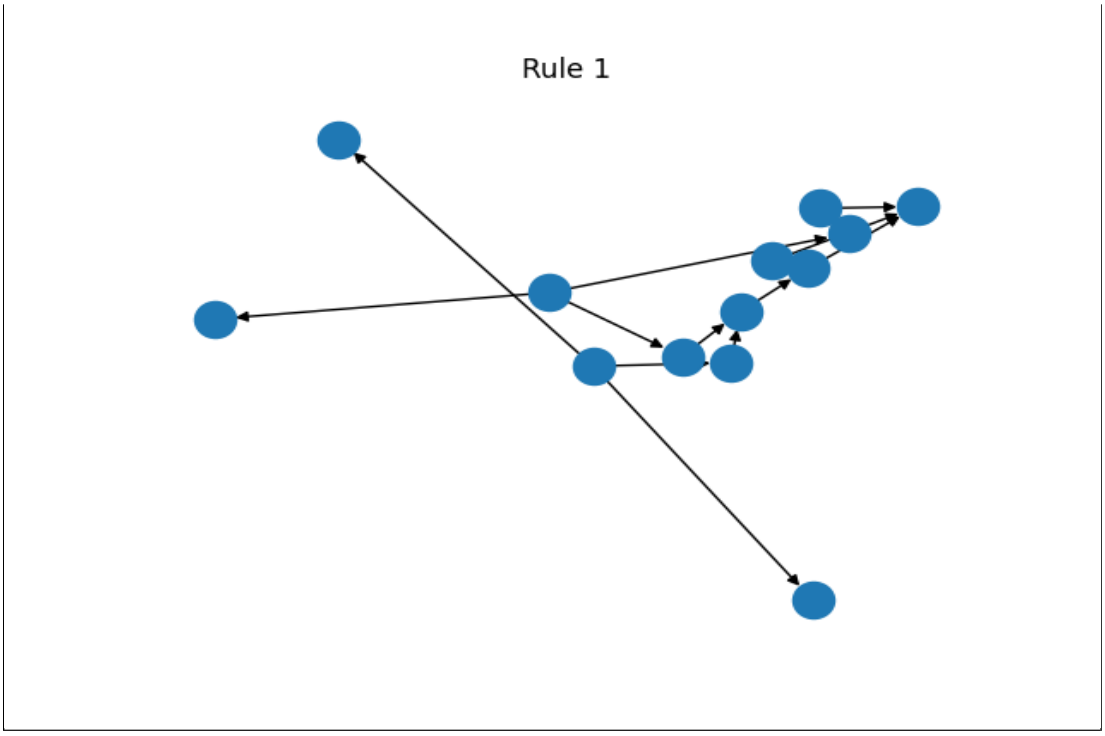
```
plt.title('Quality')
service['poor'].view()
plt.title('Service')
tip['medium'].view()
plt.title('Tip Medium')

rule1 = ctrl.Rule(quality['poor'] | service['poor'], tip['low'])
rule2 = ctrl.Rule(service['average'], tip['medium'])
rule3 = ctrl.Rule(service['good'] | quality['good'], tip['high'])
rule1.view()
plt.title('Rule 1')
rule2.view()
plt.title('Rule 2')
rule3.view()
plt.title('Rule 3')

tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
tipping = ctrl.ControlSystemSimulation(tipping_ctrl)
tipping.input['quality'] = 6.5
tipping.input['service'] = 9.8
tipping.compute()
tip.view(sim=tipping)

plt.title('Result')
plt.show(block=True)
```

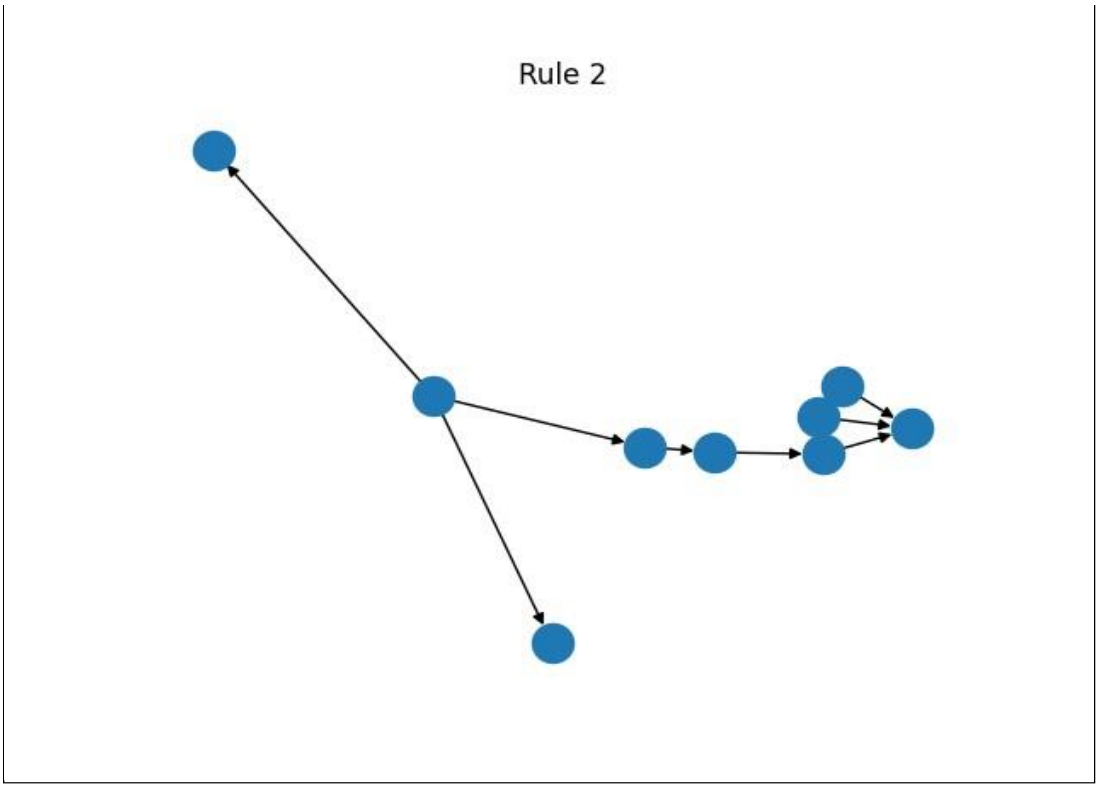
Output:

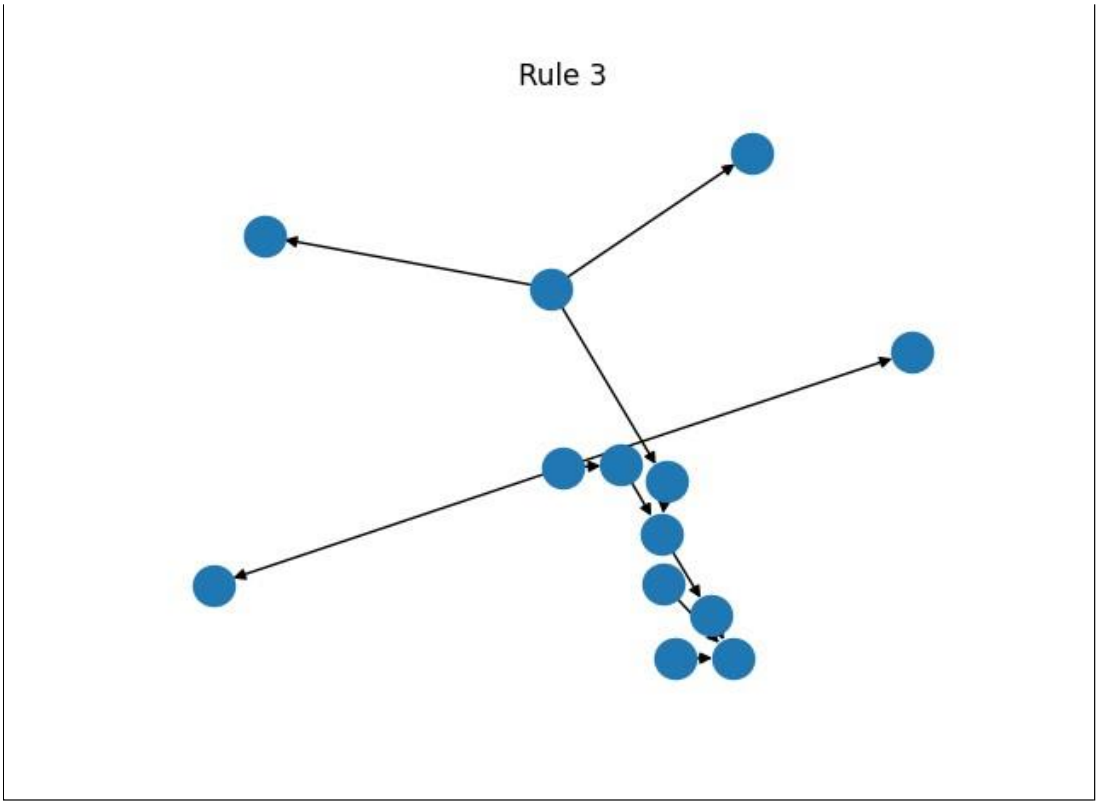

Quality
1.0
0.8
0.6
0.4
0.2
0.0
Membership
poor
average
good
0
2
4
6
8
quality

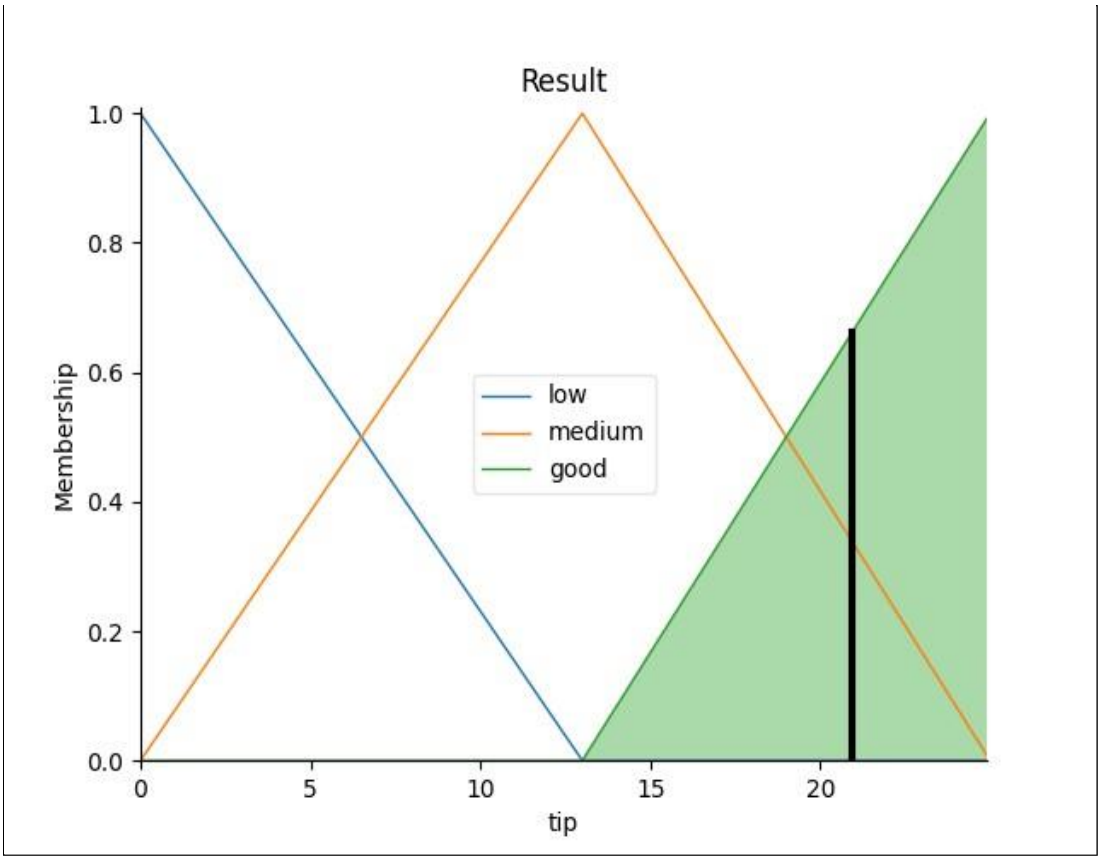

Service
1.0
0.8
0.6
0.4
0.2
0.0
Membership
poor
average
good
0
2
4
6
8
service


Tip Medium
1.0
0.8
0.6
0.4
0.2
0.0
Membership
low
medium
good
0
5
10
15
20
tip


Rule 1


Rule 2


Rule 3


Result
1.0
0.8
0.6
0.4
0.2
0.0
Membership
low
medium
good
0
5
10
15
20
tip

Practical No: 7

Aim: [A] Write an application to simulate supervised learning model.

Code:

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import datasets

iris = datasets.load_iris()
x = iris.data
y = iris.target
print('sepal-length', 'sepal-width', 'petal-length', 'petal-width')
print(x)
print('Class: 0-Iris-Setosa, 1- Iris-Versicolour, 2- Iris-Virginica')
print(y)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)

classifier = KNeighborsClassifier(n_neighbors = 5)
classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)
print('Confusion Matrix')
print(confusion_matrix(y_test, y_pred))
print('Accuracy Metrics')
print(classification_report(y_test, y_pred))
```

Output:

```
sepal-length sepal-width petal-length petal-width
```

Squeezed text (150 lines).

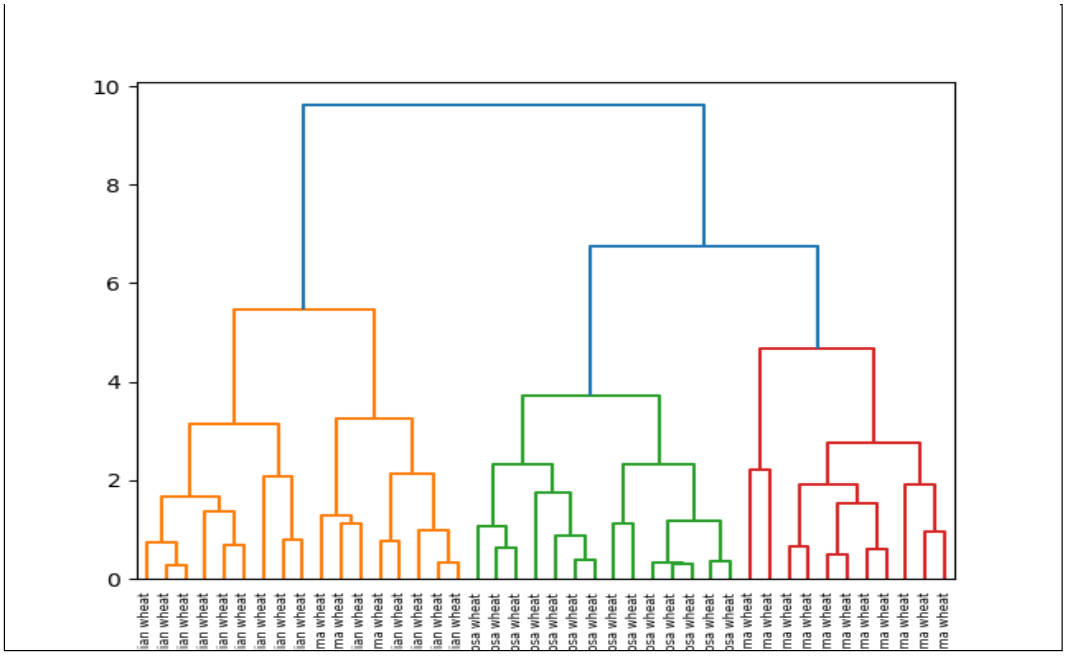
```
Class: 0-Iris-Setosa, 1- Iris-Versicolour, 2- Iris-Virginica
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
Confusion Matrix
[[15  0  0]
 [ 0 13  1]
 [ 0  0 16]]
Accuracy Metrics
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      0.93      0.96        14
           2       0.94      1.00      0.97        16

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

Aim: [B] Write an application to simulate unsupervised learning model.

Code:

```
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt
import pandas as pd

seeds_df = pd.read_csv("C:\\Users\\mihir\\Downloads\\seeds-less-rows.csv")
varieties = list(seeds_df.pop('grain_variety'))
samples = seeds_df.values

mergings = linkage(samples, method = 'complete')

dendrogram(mergings, labels=varieties, leaf_rotation=90, leaf_font_size=6)
plt.show()
```

Output:



Practical No: 8

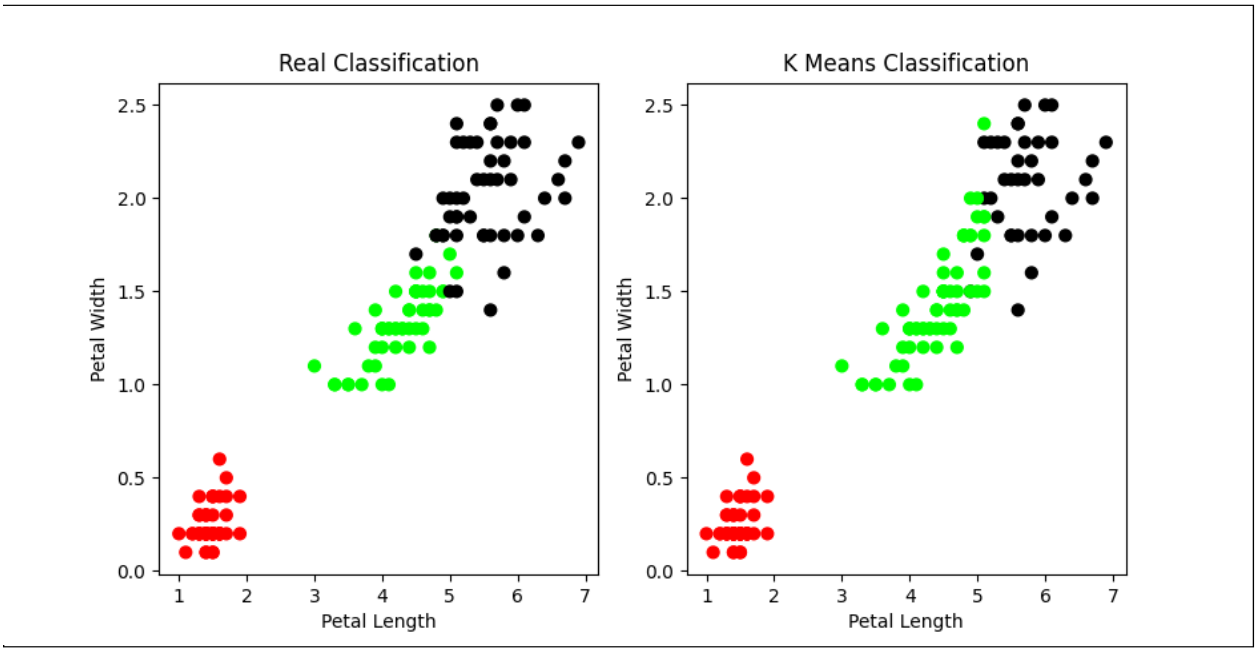
Aim: Write an application to implement clustering algorithm.

Code:

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import sklearn.metrics as sm
import pandas as pd
import numpy as np

iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=['Sepal_Length', 'Sepal_Width', 'Petal_Length',
'Petal_Width'])
y = pd.DataFrame(iris.target, columns=['Targets'])

model = KMeans(n_clusters=3, n_init=10)
model.fit(X)

colormap = np.array(['red', 'lime', 'black'])
plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[iris.target], s=40)
plt.title('Red Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')

plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
```

```
plt.title('K Means Clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()
print('The Accuracy score of K-Means: ', sm.accuracy_score(iris.target, model.labels_))
print('The Confusion matrix of K-Means: ')
print(sm.confusion_matrix(iris.target, model.labels_))
```

Output:




```
The Accuracy score of K-Means:  0.8933333333333333
The Confusion matrix of K-Means:
[[50  0  0]
 [ 0 48  2]
 [ 0 14 36]]
```

Practical No: 9

Aim: Write an application to implement Support Vector Machine algorithm.

Code:

```
from sklearn import datasets

from sklearn import svm

from sklearn.model_selection import train_test_split

from sklearn import metrics


cancer = datasets.load_breast_cancer()

print('Features: ', cancer.feature_names)

print('Labels: ', cancer.target_names)

print('Data shape: ', cancer.data.shape)


X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.3,
random_state=109)

clf = svm.SVC(kernel='linear')

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

print('Accuracy: ', metrics.accuracy_score(y_test, y_pred))

print('Precision: ', metrics.precision_score(y_test, y_pred))

print('Recall: ', metrics.recall_score(y_test, y_pred))
```

Output:

```
Features:  ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
Labels:  ['malignant' 'benign']
Data shape:  (569, 30)
Accuracy:  0.9649122807017544
Precision:  0.9811320754716981
Recall:  0.9629629629629629
```

Practical No: 10

Aim: Simulate artificial neural network model with both feedforward and backpropagation approach.

Code:

```
import numpy as np

X = np.array(([2,9], [1,5], [3,6]), dtype=float)
y = np.array(([92], [86], [89]), dtype=float)
X = X / np.amax(X, axis=0)
y = y / 100

def sigmoid(x):
   return 1 / (1 + np.exp(-x))

def derivatives_sigmoid(x):
   return x * (1 - x)

epoch = 5000
lr = 0.1
inputlayer_neurons = 2
hiddenlayer_neurons = 3
outputlayer_neurons = 1

wh = np.random.uniform(size=(inputlayer_neurons, hiddenlayer_neurons))
bh = np.random.uniform(size=(1, hiddenlayer_neurons))
wout = np.random.uniform(size=(hiddenlayer_neurons, outputlayer_neurons))
```

```
bout = np.random.uniform(size=(1, outputlayer_neurons))

for i in range(epoch):
    #Forward Propogation
    hinp1 = np.dot(X,wh)
    hinp = hinp1 + bh
    hlayer_act = sigmoid(hinp)
    outinp1 = np.dot(hlayer_act, wout)
    outinp = outinp1 + bout
    output = sigmoid(outinp)

    #Backpropagation
    EO = y - output
    outgrad = derivatives_sigmoid(output)
    d_output = EO * outgrad
    EH = d_output.dot(wout.T)
    hiddengrad = derivatives_sigmoid(hlayer_act)
    d_hiddenlayer = EH * hiddengrad

wout += hlayer_act.T.dot(d_output) * lr
wh += X.T.dot(d_hiddenlayer) * lr

print('Input: \n'+ str(X))
print('Actual Output: \n'+ str(y))
print('Predicted Outpur: \n', output)
```

Output:

```
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Outpur:
 [[0.91611219]
 [0.90155494]
 [0.91941754]]
```

Machine Learning

Index

Sr. No.	Title	Date	Sign
1a	Design a simple machine learning model to train the training instances and test the same.		
1b	Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file		
2a	Perform Data Loading, Feature selection (Principal Component analysis) and Feature Scoring and Ranking.		
2b	For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.		
3a	Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets		
3b	Write a program to implement Decision Tree and Random forest with Prediction, Test Score and Confusion Matrix.		
4a	For a given set of training data examples stored in a .CSV file implement Least Square Regression algorithm		
4b	For a given set of training data examples stored in a .CSV file implement Logistic Regression algorithm.		
5a	Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.		
5b	Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set.		
6a	Implement the different Distance methods (Euclidean) with Prediction, Test Score and Confusion Matrix.		
6b	Implement the classification model using clustering for the following techniques with K means clustering with Prediction, Test Score and Confusion Matrix.		

7	Implement the classification model using clustering for the following techniques with hierarchical clustering with Prediction, Test Score and Confusion Matrix		
8a	Exploratory Data Analysis (EDA)		
8b	Time Series Analysis		
9	Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model to perform this task.		
10	Perform Text pre-processing, Text clustering, classification with Prediction, Test Score and Confusion Matrix		

Practical 1

A. Design a simple machine learning model to train the training instances and test the same.

In [1]:
```
# Generating the Training Set

# python library to generate random numbers
from random import randint

# the limit within which random numbers are generated
TRAIN_SET_LIMIT = 1000

# to create exactly 100 data items
TRAIN_SET_COUNT = 100

# list that contains input and corresponding output
TRAIN_INPUT = list()
TRAIN_OUTPUT = list()
```

In [4]:
```
# loop to create 100 data items with three columns each
for i in range(TRAIN_SET_COUNT):
        a = randint(0, TRAIN_SET_LIMIT)
        b = randint(0, TRAIN_SET_LIMIT)
        c = randint(0, TRAIN_SET_LIMIT)

# creating the output for each data item
        op = a + (2 * b) + (3 * c)
        TRAIN_INPUT.append([a, b, c])

# adding each output to output list
        TRAIN_OUTPUT.append(op)
```

In [6]:
```
# printing first 10 records
TRAIN_OUTPUT[:10]
```

Out[6]:
```
[1122, 2518, 5291, 3471, 3522, 4008, 4547, 3760, 4414, 1890]
```

In [7]:
```
# printing first 10 records
TRAIN_INPUT[:10]
```

Out[7]:
```
[[306, 270, 92],
 [477, 233, 525],
 [514, 908, 987],
 [576, 84, 909],
 [517, 664, 559],
 [652, 424, 836],
 [672, 559, 919],
 [805, 210, 845],
 [221, 763, 889],
 [300, 666, 86]]
```

The Model can be created in two steps:-

1. Training the model with Training Data

2. Testing the model with Test Data

In [8]:
```
# Training the Model
# The data that was created using the above code is used to train the model

# Sk-Learn contains the linear regression model
from sklearn.linear_model import LinearRegression

# Initialize the linear regression model
predictor = LinearRegression(n_jobs =-1)

# Fill the Model with the Data
predictor.fit(X = TRAIN_INPUT, y = TRAIN_OUTPUT)
```

Out[8]:
```
LinearRegression(n_jobs=-1)
```

Testing the Data

The testing is done Manually. Testing can be done using some random data and testing if the model gives the correct result for the input data.

In [9]:
```
# Random Test data
X_TEST = [[ 10, 20, 30 ]]   #---> 10 + 20*2 + 30*3 = 140.

# Predict the result of X_TEST which holds testing data
outcome = predictor.predict(X = X_TEST)

# Predict the coefficients
coefficients = predictor.coef_

# Print the result obtained for the test data
print('Outcome : {}\nCoefficients : {}'.format(outcome, coefficients))
```

```
Outcome : [140.]
Coefficients : [1. 2. 3.]
```

In [ ]:
```

```

B. Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Readthe training data from a .CSV file.

In [2]:
```
import random
import csv


attributes = [['Sunny','Rainy'],
              ['Warm','Cold'],
              ['Normal','High'],
              ['Strong','Weak'],
              ['Warm','Cool'],
              ['Same','Change']]


num_attributes = len(attributes)
```

In [3]:
```
print (" \n The most general hypothesis : ['?','?','?','?','?','?']\n")
print ("\n The most specific hypothesis : ['0','0','0','0','0','0']\n")
```

```
 The most general hypothesis : ['?','?','?','?','?','?']


 The most specific hypothesis : ['0','0','0','0','0','0']
```

In [4]:
```
a = []
print("\n The Given Training Data Set \n")

with open('/content/data.csv', 'r') as csvFile:
    reader = csv.reader(csvFile)
    for row in reader:
        a.append (row)
        print(row)
```

```
 The Given Training Data Set

['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'Yes']
['Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same', 'Yes']
['Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change', 'No']
['Sunny', 'Warm', 'High', 'Strong', 'Cool', 'Change', 'Yes']
```

In [5]:
```
print("\n The initial value of hypothesis: ")
hypothesis = ['0'] * num_attributes
print(hypothesis)

# Comparing with First Training Example
for j in range(0,num_attributes):
        hypothesis[j] = a[0][j];

# Comparing with Remaining Training Examples of Given Data Set

print("\n Find S: Finding a Maximally Specific Hypothesis\n")

for i in range(0,len(a)):
    if  a[i][num_attributes]=='Yes':
```

```
            for j in range(0,num_attributes):
                if  a[i][j]!=hypothesis[j]:
                    hypothesis[j]='?'
                else :
                    hypothesis[j]= a[i][j]
    print(" For Training Example No :{0} the hypothesis is ".format(i),hypothesis)

print("\n The Maximally Specific Hypothesis for a given Training Examples :\n")
print(hypothesis)
```

```
 The initial value of hypothesis:
['0', '0', '0', '0', '0', '0']

 Find S: Finding a Maximally Specific Hypothesis

 For Training Example No :0 the hypothesis is  ['Sunny', 'Warm', 'Normal', 'Stron
g', 'Warm', 'Same']
 For Training Example No :1 the hypothesis is  ['Sunny', 'Warm', '?', 'Strong', 'W
arm', 'Same']
 For Training Example No :2 the hypothesis is  ['Sunny', 'Warm', '?', 'Strong', 'W
arm', 'Same']
 For Training Example No :3 the hypothesis is  ['Sunny', 'Warm', '?', 'Strong',
'?', '?']

 The Maximally Specific Hypothesis for a given Training Examples :

['Sunny', 'Warm', '?', 'Strong', '?', '?']
```

In [ ]:

Practical 2

A. Perform Data Loading, Feature selection (Principal Component analysis) and Feature Scoringand Ranking.

Principal component analysis, or PCA, is a statistical technique to convert high dimensional data to low dimensional data by selecting the most important features that capture maximum information about the dataset. The features are selected on the basis of variance that they cause in the output. The feature that causes highest variance is the first principal component. The feature that is responsible for second highest variance is considered the second principal component, and so on. It is important to mention that principal components do not have any correlation with each other.

In [22]:
```
import numpy as np
import pandas as pd
```

In [23]:
```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
dataset = pd.read_csv(url, names=names)
```

In [24]:
```
dataset.head()
```

Out[24]:

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Preprocessing

The first preprocessing step is to divide the dataset into a feature set and corresponding labels. The following script performs this task:

In [25]:
```
X = dataset.drop('Class', 1)
y = dataset['Class']
```

```
<ipython-input-25-a134714d2c8b>:1: FutureWarning: In a future version of pandas al
l arguments of DataFrame.drop except for the argument 'labels' will be keyword-onl
y
  X = dataset.drop('Class', 1)
```

In [26]:
```
y
```

Out[26]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
            ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Class, Length: 150, dtype: object
```

The script above stores the feature sets into the X variable and the series of corresponding labels in to the y variable.

The next preprocessing step is to divide data into training and test sets. Execute the following script to do so:

In [27]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
```

PCA performs best with a normalized feature set. We will perform standard scalar normalization to normalize our feature set. To do this, execute the following code:

In [28]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train_s = sc.fit_transform(X_train)
X_test_s = sc.transform(X_test)
```

Applying PCA

It is only a matter of three lines of code to perform PCA using Python's Scikit-Learn library. The PCA class is used for this purpose. PCA depends only upon the feature set and not the label data. Therefore, PCA can be considered as an unsupervised machine learning technique.

Performing PCA using Scikit-Learn is a two-step process:

Initialize the PCA class by passing the number of components to the constructor. Call the fit and then transform methods by passing the feature set to these methods. The transform method returns the specified number of principal components.

In [29]:
```
from sklearn.decomposition import PCA

pca = PCA()
X_train = pca.fit_transform(X_train_s)
X_test = pca.transform(X_test_s)
```

In the code above, we create a PCA object named pca.

We did not specify the number of components in the constructor. Hence, all four of the features in the feature set will be returned for both the training and test sets.

The PCA class contains explained_variance*ratio* which returns the variance caused by each of the principal components.

Execute the following line of code to find the "explained variance ratio".

In [30]:
```
explained_variance = pca.explained_variance_ratio_
```

In [31]:
```
explained_variance
```

Out[31]:
```
array([0.72226528, 0.23974795, 0.03338117, 0.0046056 ])
```

It can be seen that first principal component is responsible for 72.22% variance. Similarly, the second principal component causes 23.9% variance in the dataset.

Collectively we can say that (72.22 + 23.9) 96.21% percent of the classification information contained in the feature set is captured by the first two principal components.

Let's first try to use 1 principal component to train our algorithm. To do so, execute the following code:

In [32]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=1)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

Training and Making Predictions

In this case we'll use random forest classification for making the predictions.

In [33]:
```
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

In [34]:
```
#Performance Evaluation

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

cm = confusion_matrix(y_test, y_pred)
print(cm)
print()
print('Accuracy', accuracy_score(y_test, y_pred))
```

```
[[11  0  0]
 [ 0 12  1]
 [ 0  1  5]]

Accuracy 0.9333333333333333
```

It can be seen from the output that with only one feature, the random forest algorithm is able to correctly predict 28 (11+12+5) out of 30 instances, resulting in 93.33% accuracy.

Now lets check, Results with 2 and 3 Principal Components

In [44]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
```

In [45]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train_s = sc.fit_transform(X_train)
X_test_s = sc.transform(X_test)
```

In [46]:
```
from sklearn.decomposition import PCA

pca = PCA()
X_train = pca.fit_transform(X_train_s)
X_test = pca.transform(X_test_s)
```

In [48]:
```
explained_variance = pca.explained_variance_ratio_
explained_variance
```

Out[48]:
```
array([0.72226528, 0.23974795, 0.03338117, 0.0046056 ])
```

In [49]:
```
# 2 components
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

In [50]:
```
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

In [51]:
```
#Performance Evaluation

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

cm = confusion_matrix(y_test, y_pred)
print(cm)
print()
print('Accuracy', accuracy_score(y_test, y_pred))
```

```
[[11  0  0]
 [ 0  9  4]
 [ 0  2  4]]

Accuracy 0.8
```

Now for 3 components same process

In [52]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
```

In [53]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train_s = sc.fit_transform(X_train)
X_test_s = sc.transform(X_test)
```

In [54]:
```
from sklearn.decomposition import PCA

pca = PCA()
X_train = pca.fit_transform(X_train_s)
X_test = pca.transform(X_test_s)
```

In [55]:
```
explained_variance = pca.explained_variance_ratio_
explained_variance
```

Out[55]:
```
array([0.72226528, 0.23974795, 0.03338117, 0.0046056 ])
```

In [56]:
```
# 3 components
from sklearn.decomposition import PCA

pca = PCA(n_components=3)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

In [57]:
```
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

In [58]:
```
#Performance Evaluation

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

cm = confusion_matrix(y_test, y_pred)
print(cm)
print()
print('Accuracy', accuracy_score(y_test, y_pred))
```

```
[[11  0  0]
 [ 0  8  5]
 [ 0  1  5]]

Accuracy 0.8
```

In [ ]:
```

```

B. For a given set of training data examples storedin a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.

Step1:Load Data set

Step2: Initialize General Hypothesis and Specific Hypothesis.

Step3: For each training example

Step4: If example is positive example

if attribute_value == hypothesis_value:

Do nothing

else:

replace attribute value with '?' (Basically generalizing it)

Step5: If example is Negative example

Make generalize hypothesis more specific.

In [1]:
```
import csv

with open("trainingexamples.csv") as f:
    csv_file = csv.reader(f)
    data = list(csv_file)

    specific = data[1][:-1]
    general = [['?' for i in range(len(specific))] for j in range(len(specific))]

    for i in data:
        if i[-1] == "Yes":
            for j in range(len(specific)):
                if i[j] != specific[j]:
                    specific[j] = "?"
                    general[j][j] = "?"

        elif i[-1] == "No":
            for j in range(len(specific)):
                if i[j] != specific[j]:
                    general[j][j] = specific[j]
                else:
                    general[j][j] = "?"

        print("\nStep " + str(data.index(i)+1) + " of Candidate Elimination Algori
        print(specific)
        print(general)

    gh = [] # gh = general Hypothesis
    for i in general:
        for j in i:
```

```
            if j != '?':
                gh.append(i)
                break
    print("\nFinal Specific hypothesis:\n", specific)
    print("\nFinal General hypothesis:\n", gh)
```

```
Step 1 of Candidate Elimination Algorithm
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?']]

Step 2 of Candidate Elimination Algorithm
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?']]

Step 3 of Candidate Elimination Algorithm
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?']]

Step 4 of Candidate Elimination Algorithm
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?',
'?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', 'Same']]

Step 5 of Candidate Elimination Algorithm
['Sunny', 'Warm', '?', 'Strong', '?', '?']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?',
'?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?']]

Final Specific hypothesis:
 ['Sunny', 'Warm', '?', 'Strong', '?', '?']

Final General hypothesis:
 [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]
```

In [ ]:

Practical 3

A. Write a program to implement the naïve Bayesian classifierfor a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
df=pd.read_csv('train.csv')
df.head()
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN

In [5]:
```
df.ndim
```

Out[5]:
```
2
```

In [6]:
```
df.shape
```

Out[6]:
```
(891, 12)
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [8]:
```
df.info()
```

```
<class  'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------------------------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [9]:
```
df.isna().sum()
```

Out[9]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [10]:
```
df.isna().any()
```

Out[10]:
```
PassengerId    False
Survived       False
Pclass         False
Name           False
Sex            False
Age             True
SibSp          False
Parch          False
Ticket         False
Fare           False
Cabin           True
Embarked        True
dtype: bool
```

In [11]:
```
df.describe().transpose()
```

Out[11]:

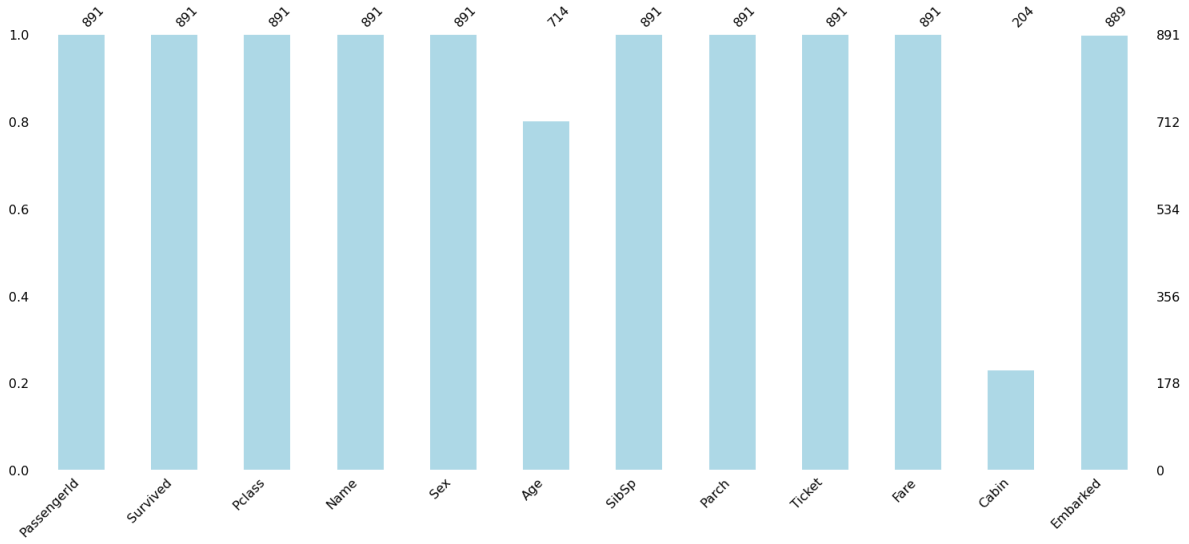
	count	mean	std	min	25%	50%	75%	max
PassengerId	891.0	446.000000	257.353842	1.00	223.5000	446.0000	668.5	891.0000
Survived	891.0	0.383838	0.486592	0.00	0.0000	0.0000	1.0	1.0000
Pclass	891.0	2.308642	0.836071	1.00	2.0000	3.0000	3.0	3.0000
Age	714.0	29.699118	14.526497	0.42	20.1250	28.0000	38.0	80.0000
SibSp	891.0	0.523008	1.102743	0.00	0.0000	0.0000	1.0	8.0000
Parch	891.0	0.381594	0.806057	0.00	0.0000	0.0000	0.0	6.0000
Fare	891.0	32.204208	49.693429	0.00	7.9104	14.4542	31.0	512.3292

In [12]:
```
null_df=pd.DataFrame()
null_df['Features']=df.isnull().sum().index
null_df['Null  values']=df.isnull().sum().values
null_df['% Null  values']=(df.isnull().sum().values  /  df.shape[0])*100
null_df.sort_values(by='% Null values',ascending=False)
```

Out[12]:

	Features	Null values	% Null values
10	Cabin	687	77.104377
5	Age	177	19.865320
11	Embarked	2	0.224467
0	PassengerId	0	0.000000
1	Survived	0	0.000000
2	Pclass	0	0.000000
3	Name	0	0.000000
4	Sex	0	0.000000
6	SibSp	0	0.000000
7	Parch	0	0.000000
8	Ticket	0	0.000000
9	Fare	0	0.000000

In [13]:
```
import missingno as no
no.bar(df,color='lightblue')
plt.show()
```



In [14]:
```
df['Age'].fillna(df.Age.median(),inplace=True)
df.Age.isna().any()
```

Out[14]:
```
False
```

In [15]:
```
df.drop(columns='Cabin' , inplace=True)
```

In [16]:
```
df.dropna(subset=['Embarked'],inplace=True)
```

In [17]:
```
df.shape
```

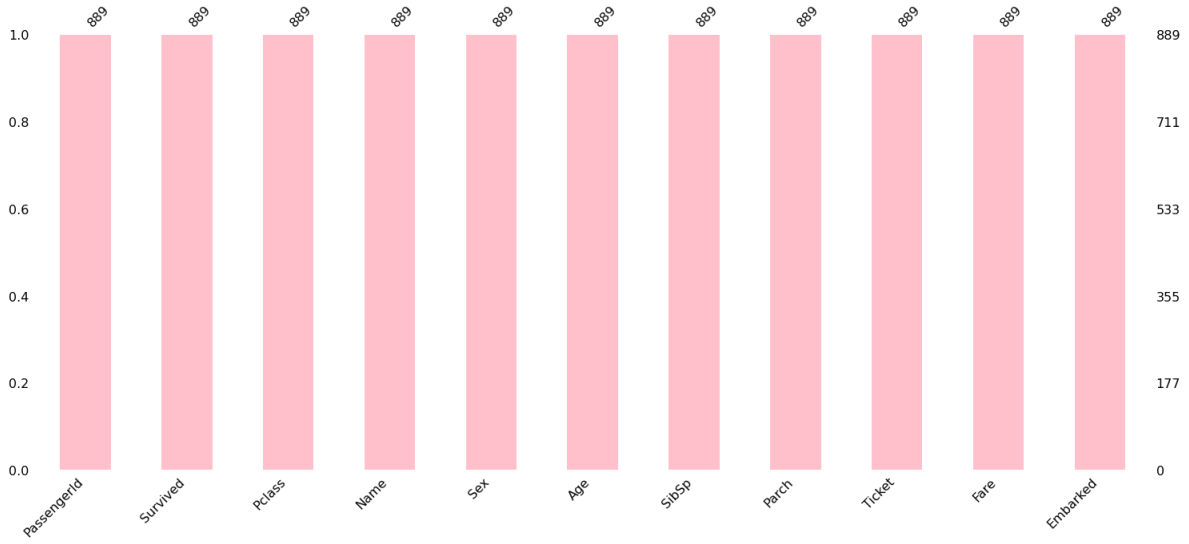
Out[17]:
```
(889, 11)
```

In [18]:
```
df.isna().sum()
```

Out[18]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

In [19]:
```
no.bar(df,color='pink')
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5228f30460>
```



Data Preprocessing

In [21]:
```
import  yellowbrick
from sklearn.preprocessing import LabelEncoder
label_Enc =LabelEncoder()
```

In [22]:
```
df.Sex  =label_Enc.fit_transform(df.Sex)
```

In [23]:
```
label_Enc.classes_
```

Out[23]:
```
array(['female',  'male'],  dtype=object)
```

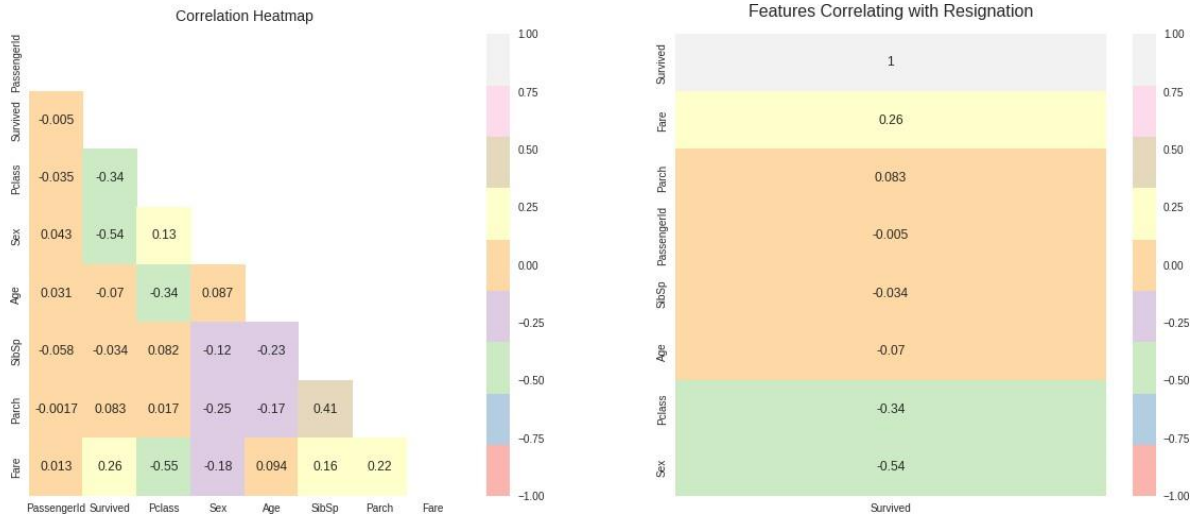
In [24]:
```
df.head()
```

Out[24]:

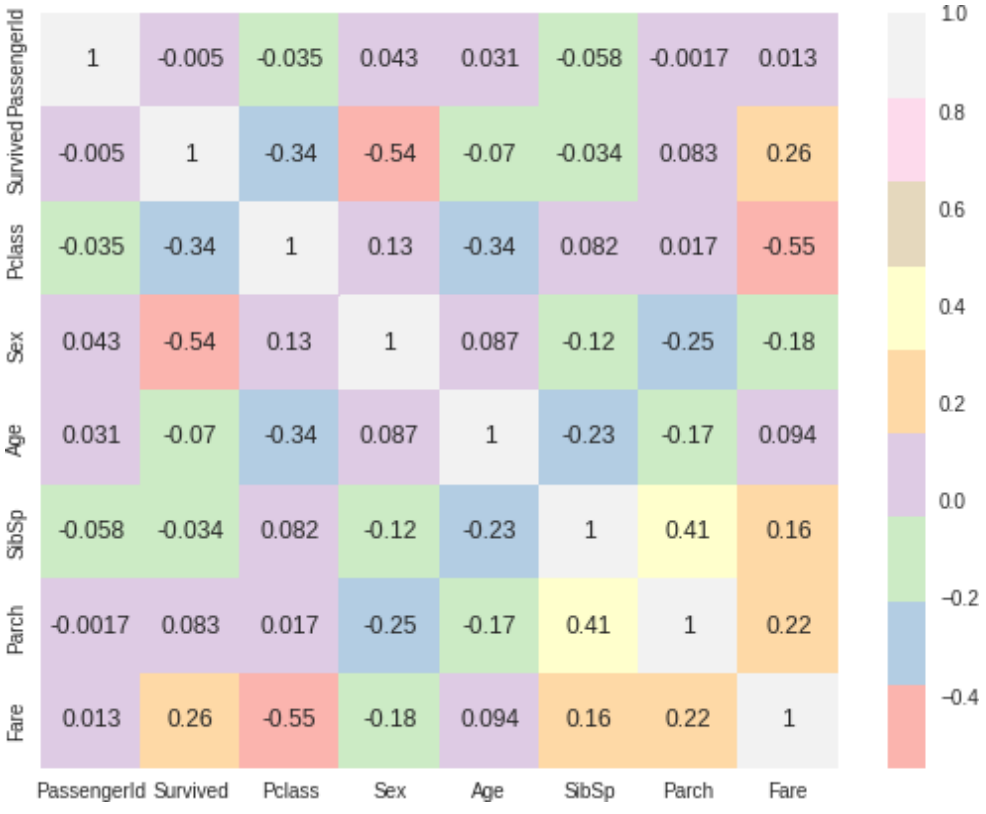
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarke
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, M ss. La na	0	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futre lle, M s. Jacq es He th (Lily May Peel)	0	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500	

In [25]:
```
fig,ax=plt.subplots(ncols=2,figsize=(20,8))
resign_corr = df.corr()
mask = np.triu(np.ones_like(resign_corr, dtype=np.bool))
cat_heatmap = sns.heatmap(df.corr(), mask=mask, vmin=-1, vmax=1,annot=True,ax=ax[0
cat_heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':14}, pad=12);
heatmap = sns.heatmap(resign_corr[['Survived']].sort_values(by='Survived',ascendin
heatmap.set_title('Features Correlating with Resignation', fontdict={'fontsize':16}
```

```
<ipython-input-25-0960bdc0efd7>:3: DeprecationWarning: `np.bool` is a deprecated a
lias for the builtin `bool`. To silence this warning, use `bool` by itself. Doing
this will not modify any behavior and is safe. If you specifically wanted the nump
y scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdoc
s/release/1.20.0-notes.html#deprecations
  mask = np.triu(np.ones_like(resign_corr, dtype=np.bool))
```

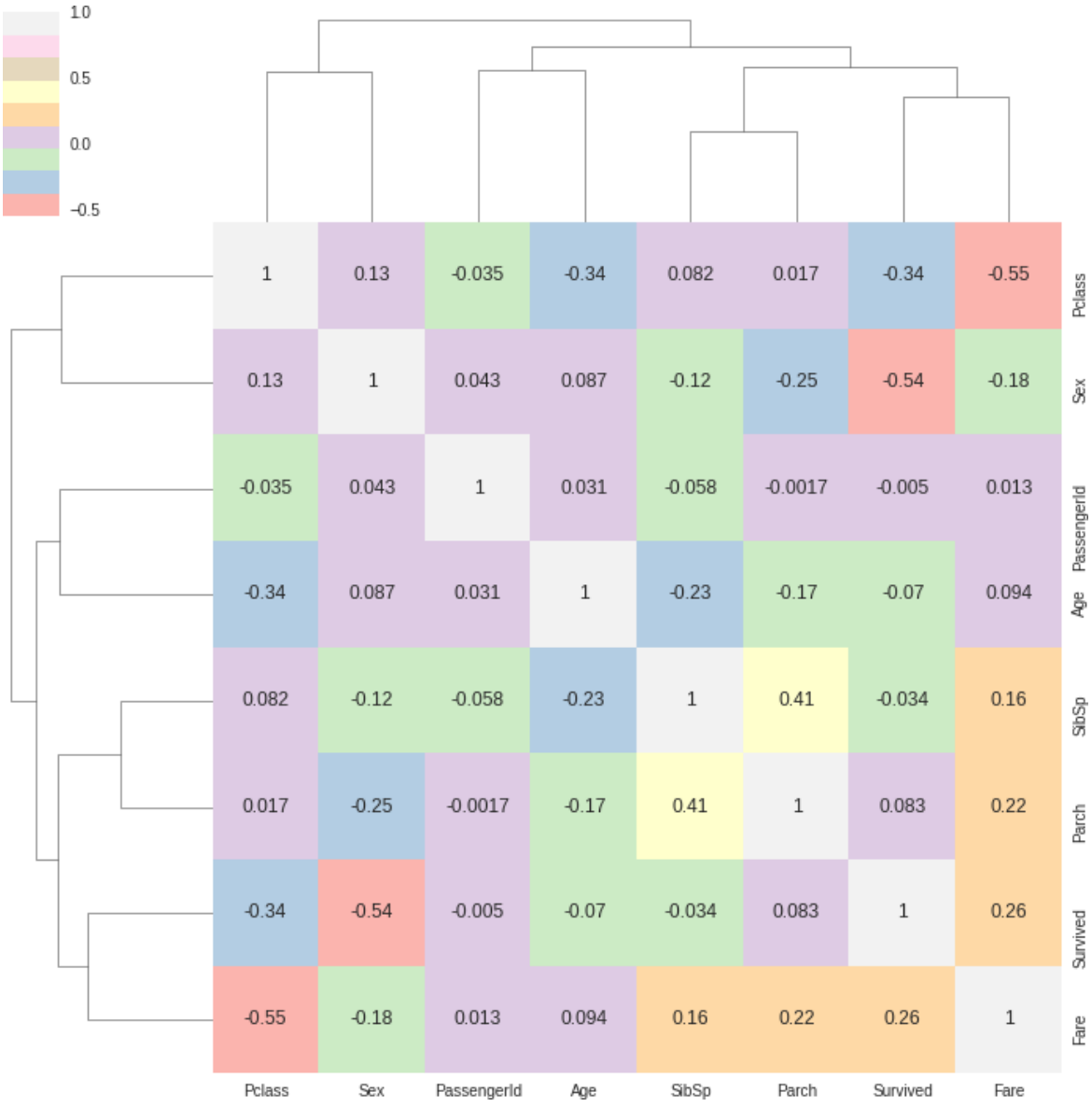


In [26]:
```
plt.figure(figsize=(9,7))
sns.heatmap(df.corr(),annot=True,cmap='Pastel1')
plt.show()
```



In [27]:
```
sns.clustermap(df.corr(),annot=True,cmap='Pastel1')
```

Out[27]:
```
<seaborn.matrix.ClusterGrid at 0x7f5224c77ca0>
```



Dataset Splitting

In [28]:
```
X=df.drop(['Survived','Name','Ticket','Embarked'],axis=1)
y=df['Survived']
```

In [29]:
```
X[:3]
```

Out[29]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare
0	1	3	1	22.0	1	0	7.2500
1	2	1	0	38.0	1	0	71.2833
2	3	3	0	26.0	0	0	7.9250

In [30]:
```
y[:5]
```

Out[30]:
```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

In [31]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test  =  train_test_split(X,y,test_size=0.2,stratify=y,rand
```

Gaussian Naive Bayes Calssifier

In [32]:
```
from sklearn.metrics import confusion_matrix, classification_report,accuracy_score
from sklearn.metrics import recall_score, precision_score, f1_score
```

In [33]:
```
from sklearn.naive_bayes import GaussianNB
```

In [34]:
```
gnb_clf=GaussianNB()
```

In [35]:
```
gnb_clf.fit(X_train, y_train)
```

Out[35]:
```
GaussianNB()
```

In [36]:
```
y_pred = gnb_clf.predict(X_test)
```

In [37]:
```
print("Accuracy Score :",accuracy_score(y_test,y_pred))
```

```
Accuracy Score : 0.7921348314606742
```

In [38]:
```
print("Recall  Score",recall_score(y_test,y_pred))
```

```
Recall  Score 0.7058823529411765
```

In [39]:
```
print("Precision Score :",precision_score(y_test,y_pred))
```

```
Precision Score : 0.7384615384615385
```

In [40]:
```
print("F1 Score :",f1_score(y_test,y_pred))
```

```
F1 Score : 0.7218045112781954
```

Confusion Matrix

In [41]:
```
confusion_matrix(y_test, y_pred)
```

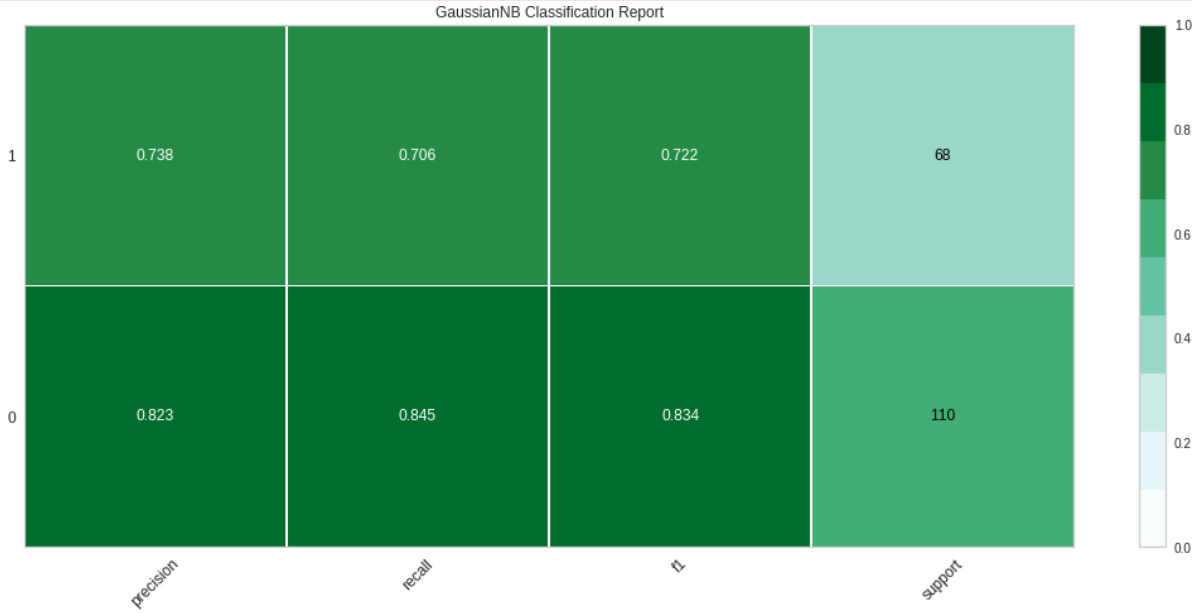
Out[41]:
```
array([[93, 17],
       [20, 48]])
```

In [42]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.85      0.83       110
           1       0.74      0.71      0.72        68

    accuracy                           0.79       178
   macro avg       0.78      0.78      0.78       178
weighted avg       0.79      0.79      0.79       178
```

In [43]:
```
import yellowbrick as yb
plt.figure(figsize=(15,7))
visualizer = yb.classifier.classification_report(gnb_clf, X_train, y_train, X_test
visualizer.show()
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does no
t have valid feature names, but GaussianNB was fitted with feature names
  warnings.warn(
```



In [ ]:

B. Write a program to implement Decision Tree and Randomforest with Prediction, Test Score and Confusion Matrix.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
```

Data Collection and Processing

In [2]:
```
# loading the csv data to a Pandas DataFrame
gold_data = pd.read_csv('/content/gld_price_data.csv')
```

In [3]:
```
# print first 5 rows in the dataframe
gold_data.head()
```

Out[3]:

	Date	SPX	GLD	USO	SLV	EUR/USD
0	1/2/2008	1447.160034	84.860001	78.470001	15.180	1.471692
1	1/3/2008	1447.160034	85.570000	78.370003	15.285	1.474491
2	1/4/2008	1411.630005	85.129997	77.309998	15.167	1.475492
3	1/7/2008	1416.180054	84.769997	75.500000	15.053	1.468299
4	1/8/2008	1390.189941	86.779999	76.059998	15.590	1.557099

In [4]:
```
# print last 5 rows of the dataframe
gold_data.tail()
```

Out[4]:

	Date	SPX	GLD	USO	SLV	EUR/USD
2285	5/8/2018	2671.919922	124.589996	14.0600	15.5100	1.186789
2286	5/9/2018	2697.790039	124.330002	14.3700	15.5300	1.184722
2287	5/10/2018	2723.070068	125.180000	14.4100	15.7400	1.191753
2288	5/14/2018	2730.129883	124.489998	14.3800	15.5600	1.193118
2289	5/16/2018	2725.780029	122.543800	14.4058	15.4542	1.182033

In [5]:
```
# number of rows and columns
gold_data.shape
```

Out[5]:
```
(2290, 6)
```

In [6]:
```
# getting some basic informations about the data
gold_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2290 entries, 0 to 2289
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Date      2290 non-null   object 
 1   SPX       2290 non-null   float64
 2   GLD       2290 non-null   float64
 3   USO       2290 non-null   float64
 4   SLV       2290 non-null   float64
 5   EUR/USD   2290 non-null   float64
dtypes: float64(5), object(1)
memory usage: 107.5+ KB
```

In [7]:
```
# checking the number of missing values
gold_data.isnull().sum()
```

Out[7]:
```
Date        0
SPX         0
GLD         0
USO         0
SLV         0
EUR/USD     0
dtype: int64
```

In [8]:
```
# getting the statistical measures of the data
gold_data.describe()
```

Out[8]:

	SPX	GLD	USO	SLV	EUR/USD
count	2290.000000	2290.000000	2290.000000	2290.000000	2290.000000
mean	1654.315776	122.732875	31.842221	20.084997	1.283653
std	519.111540	23.283346	19.523517	7.092566	0.131547
min	676.530029	70.000000	7.960000	8.850000	1.039047
25%	1239.874969	109.725000	14.380000	15.570000	1.171313
50%	1551.434998	120.580002	33.869999	17.268500	1.303297
75%	2073.010070	132.840004	37.827501	22.882500	1.369971
max	2872.870117	184.589996	117.480003	47.259998	1.598798

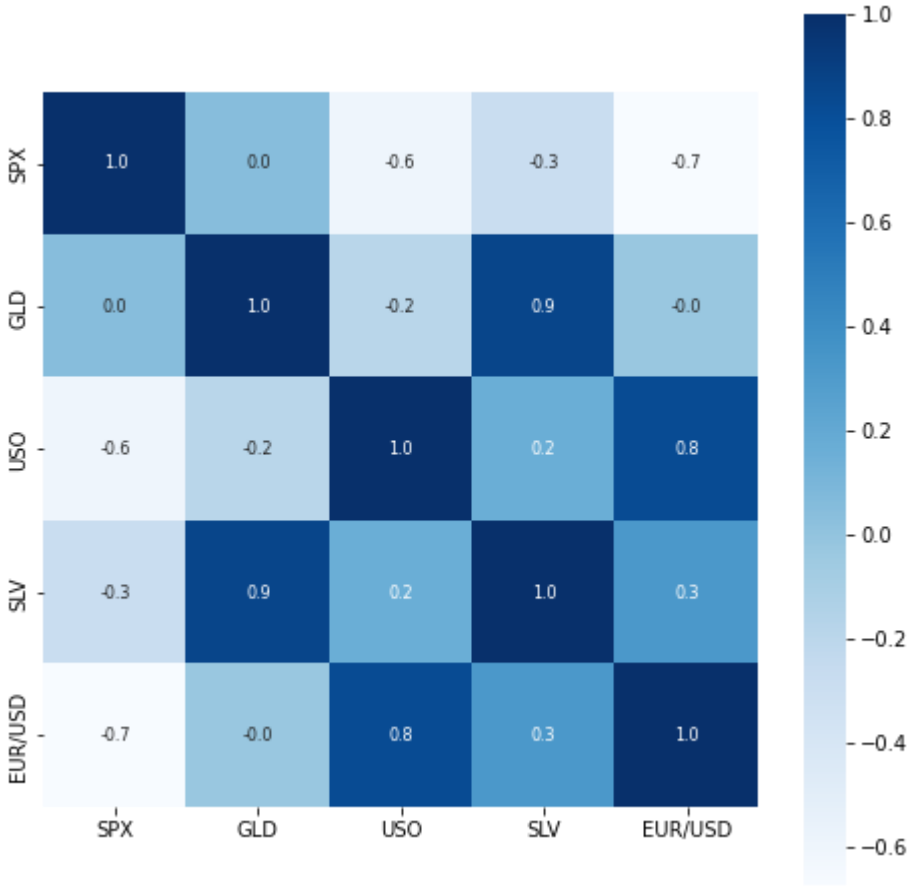
Correlation:

1. Positive Correlation
2. Negative Correlation

In [9]:
```
correlation = gold_data.corr()
```

In [10]:
```
# constructing a heatmap to understand the correlatiom
plt.figure(figsize = (8,8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f',annot=True, annot_kws={
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9593846100>
```



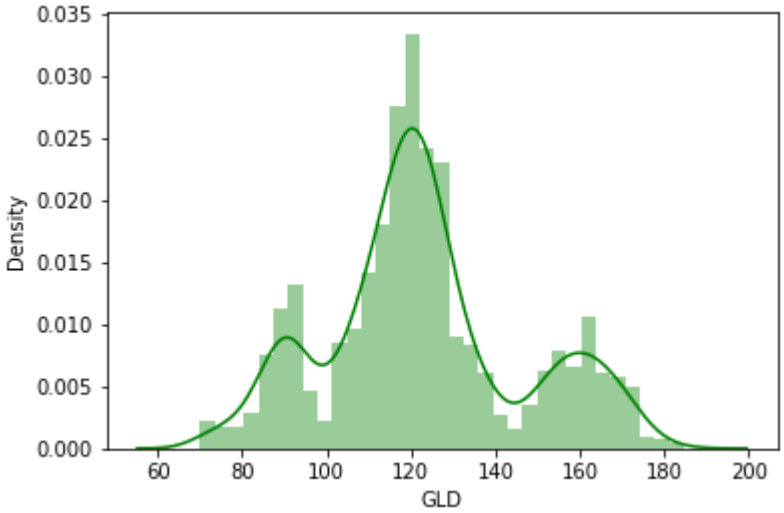
In [11]:
```
# correlation values of GLD
print(correlation['GLD'])
```

```
SPX        0.049345
GLD        1.000000
USO       -0.186360
SLV        0.866632
EUR/USD   -0.024375
Name: GLD, dtype: float64
```

In [12]:
```
# checking the distribution of the GLD Price
sns.distplot(gold_data['GLD'],color='green')
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619:  FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg,  FutureWarning)
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot  at  0x7f959388e700>
```



Splitting the Features and Target

In [13]:
```
X = gold_data.drop(['Date','GLD'],axis=1)
Y = gold_data['GLD']
```

In [14]:
```
print(X)
```

```
              SPX        USO      SLV   EUR/USD
0     1447.160034  78.470001  15.1800  1.471692
1     1447.160034  78.370003  15.2850  1.474491
2     1411.630005  77.309998  15.1670  1.475492
3     1416.180054  75.500000  15.0530  1.468299
4     1390.189941  76.059998  15.5900  1.557099
...           ...        ...      ...       ...
2285  2671.919922  14.060000  15.5100  1.186789
2286  2697.790039  14.370000  15.5300  1.184722
2287  2723.070068  14.410000  15.7400  1.191753
2288  2730.129883  14.380000  15.5600  1.193118
2289  2725.780029  14.405800  15.4542  1.182033

[2290 rows x 4 columns]
```

In [15]:
```
print(Y)
```

```
0        84.860001
1        85.570000
2        85.129997
3        84.769997
4        86.779999
           ...
2285    124.589996
2286    124.330002
2287    125.180000
2288    124.489998
2289    122.543800
Name: GLD, Length: 2290, dtype: float64
```

Splitting into Training data and Test Data

In [16]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_
```

Model Training: Random Forest Regressor

In [17]:
```
regressor = RandomForestRegressor(n_estimators=100)
```

In [18]:
```
# training the model
regressor.fit(X_train,Y_train)
```

Out[18]:
```
RandomForestRegressor()
```

Model Evaluation

In [19]:
```
# prediction on Test Data
test_data_prediction = regressor.predict(X_test)
```

In [20]:
```
print(test_data_prediction)
```

```
[168.8024997   81.76539994 116.12460019 127.62850073 120.48470156
 154.66519818 150.24759836 125.84670103 117.63599858 126.1703004
 116.85650075 172.02970087 141.59679814 167.85919873 115.10210022
 117.82960026 138.57680361 169.86320092 159.40210318 161.05689992
 154.91850014 125.44000014 175.49340004 157.50830401 125.1320003
  93.66869953  78.02370047 120.52970031 119.13349971 167.45739924
  88.04280098 125.24589994  91.14940071 117.66750035 121.2089994
 135.98950078 115.694701   115.04590048 146.44709943 106.99790111
 104.82810278  87.04569765 126.48040045 118.10090015 152.68889877
 119.61880008 108.28900029 108.03599864  93.27370055 127.08999789
  75.02340024 113.68889914 121.17329996 111.22789913 118.92419886
 120.5121995  160.02059918 168.33770156 147.09079678  85.83389886
  94.36280034  86.79519902  90.50050024 119.04540061 126.49490095
 127.72149976 171.15230023 122.4069992  117.51719915  98.86540018
 167.9912021  143.15119877 132.31180264 121.23400213 121.05899936
 119.94710073 114.55480113 118.34530054 106.96280105 128.01270153
 113.95429968 107.48039984 116.68990078 119.65489935  88.99480029
  88.26269898 146.4706018  127.41390019 113.40880057 110.5933987
 108.00199893  77.71559894 169.88760205 114.04949917 121.74349888
 127.43100145 154.88539843  91.87309932 135.033001   159.02880316
 125.05030046 125.63790047 130.52790141 114.99450071 119.80359947
  92.04919968 110.24699897 167.74819956 156.49949951 114.1226994
 106.6976014  79.26499986 113.33470035 125.84690105 107.2521991
 119.56700101 155.6990033  159.7409986  119.96249984 134.27220299
 101.39339988 117.32829809 119.38200035 112.91350054 102.7760988
 160.50779769  99.36290041 147.62240009 125.3281011  169.61439948
 125.59379886 127.37199737 127.57160148 113.86039913 113.08560077
 123.54299896 102.1041991   88.95689982 124.7889997  101.74339927
 107.01119913 113.6225001  117.55810039  99.19359956 121.67850059
 163.340599    87.33899855 106.7294999  117.2687009  127.77440158
 124.01900058  80.77579923 120.57860031 156.91729846  87.98909965
 110.27619958 119.03359907 172.69829885 102.98369909 105.86200059
 122.34990023 156.93189786  87.64069835  93.04940034 112.69390038
 177.21080023 114.45490017 119.43210034  94.66300117 125.65990073
 165.93120159 115.08220066 116.62860131  88.28939885 149.1347014
 120.4637994  89.55709972 112.78680013 117.15080065 118.7462013
  88.20489995  94.05830011 117.15969981 118.50690153 120.33750005
 126.66489866 121.92029971 148.99210006 165.73610109 118.56489973
 120.48780117 150.76999997 118.35839864 172.53419942 105.24789919
 104.968201   149.43500155 114.01530066 124.81940116 147.6377997
 119.51870136 115.36640055 112.68490007 113.46520213 140.84430107
 117.9862976  102.99780036 115.891301   104.04320214  98.71730056
 117.41470054  90.71890031  91.57620083 153.45039876 102.69129976
 154.46740059 114.34300187 138.92410158  90.10079834 115.48689965
 114.68779993 123.29140081 121.83640046 165.45400081  92.95649931
 135.85150099 121.29709917 120.67990072 104.71530006 141.99780292
 121.78439933 116.78350033 113.30420113 126.92689804 122.36689966
 125.76879951 121.1925002   86.81459914 132.35280162 145.49390165
  92.57609963 157.42719961 159.22610235 126.34419887 165.00649977
 108.79369928 110.51460074 103.81319825  94.44030088 127.45920238
 106.85330056 160.44280034 121.61150058 132.07840053 130.55650141
 160.3821997  90.11689866 175.32800211 127.7070003  126.8019989
  86.36999907 124.59159927 150.31859705  89.73030008 106.9375
 109.01819982  84.51399903 135.98869973 154.97860217 138.48350371
  73.99380007 151.70510101 126.09000092 126.82530024 127.55489919
 108.71409963 156.1334008  114.77340088 116.90080167 125.46199968
 153.99840094 121.46409979 156.44939912  92.90840041 125.55970153
 126.02880037  87.78300066  92.25049951 126.39029925 128.60650335
 113.18820028 117.55219758 120.7733002  127.02999853 119.59820125
 135.96440047  93.80519931 119.75460029 113.22380113  94.25969941
 108.92519934  87.20519881 108.94899918  89.62599972  92.5899002
 131.4442032  162.36510047  89.35050028 119.52930064 133.46600222
 123.85570026 128.31960197 102.02209867  88.87959882 131.31090094
 119.73370041 108.6379002  167.83540179 115.12450037  86.6030987
```

```
118.81820093  91.06069994 162.07819985 116.63010042 121.49199974
160.18019772 120.12209912 112.79689913 108.42099855 126.74099978
 76.11180024 102.97489994 127.08480207 121.75219942  92.58969992
131.91230033 118.03380151 116.19869968 154.46590275 160.02860093
110.24209944 156.64859782 119.32080091 160.58870059 118.61570049
158.23189948 115.24159956 116.4455005  149.64229906 114.91860058
125.655599   165.52139973 117.64900015 124.85989946 153.17650362
153.39210264 132.03060023 114.90310047 121.2380019  125.08720072
 89.70290063 123.42919984 154.73330242 111.52600018 106.62440001
162.10980135 118.53599955 165.63710018 134.38040104 115.23589997
152.99509915 168.62709947 114.55610007 114.09480121 158.15769901
 85.16059911 127.11650106 127.94040045 129.01919953 124.28710069
123.93290094  90.54930076 153.17020012  97.15369954 136.83959992
 89.07429904 107.41179991 114.99410044 112.81100059 124.2679991
 91.37529878 125.42780132 162.44889908 120.00079866 165.15160049
126.71939854 112.1334999  127.57619926  95.02019889  91.00459956
103.35769885 120.71540035  83.0787994  126.29580012 160.13240427
117.35450088 118.37429985 119.88980003 122.66529963 120.13190125
121.58870012 118.07590071 106.90810018 148.45260034 126.15509888
115.62890103  73.66800009 127.81950124 153.95840036 121.86300034
125.5668005   88.86170017 104.08609876 125.06970061 120.16880036
 73.33340079 152.03700024 121.0907005  104.63079986  86.35239777
115.26989943 172.17669834 119.98030027 160.31919831 113.16560009
121.10420029 118.27880149  95.89909985 118.37110011 125.85890034
118.50799956  96.23690073 154.07480154 122.10479965 147.46699974
159.31840212 113.98780014 122.5353993  148.34009846 127.62480039
165.58640094 135.0120001  119.92969911 167.21709849 108.24659974
121.75059849 138.82410125 107.41269918]
```

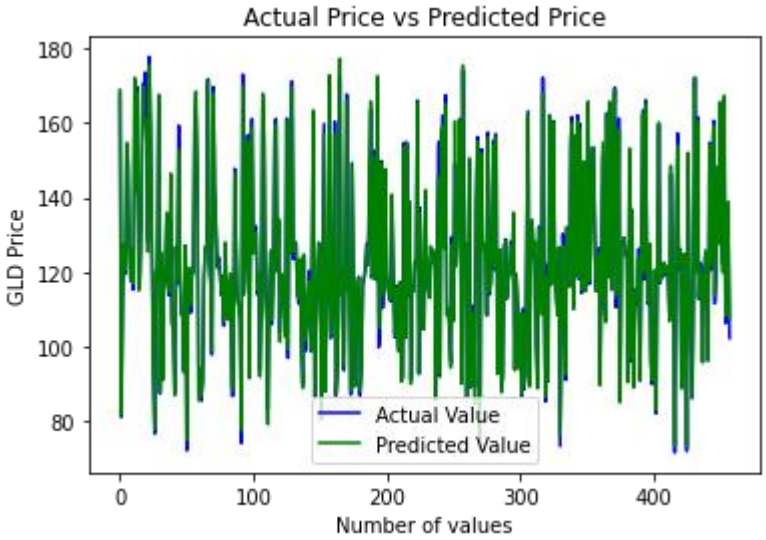
In [21]:
```
# R squared error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R squared error : ", error_score)
```

```
R squared error :  0.9897511997931183
```

Compare the Actual Values and Predicted Values in a Plot

In [22]:
```
Y_test = list(Y_test)
```

In [23]:
```
plt.plot(Y_test, color='blue', label = 'Actual Value')
plt.plot(test_data_prediction, color='green', label='Predicted Value')
plt.title('Actual Price vs Predicted Price')
plt.xlabel('Number of values')
plt.ylabel('GLD Price')
plt.legend()
plt.show()
```



In [24]:
```
# Decision Tree

# import the regressor
from sklearn.tree import DecisionTreeRegressor

# create a regressor object
regressor = DecisionTreeRegressor(random_state = 0)

# fit the regressor with X and Y data
regressor.fit(X_train,Y_train)
```

Out[24]:
```
DecisionTreeRegressor(random_state=0)
```

In [31]:
```
# predicting a new value

# test the output
y_pred = regressor.predict(X_test)
y_pred
```

Out[31]:
```
array([168.970001,  86.089996, 114.769997, 127.550003, 121.730003,
       155.669998, 149.149994, 126.809998, 117.389999, 125.620003,
       117.959999, 174.580002, 141.919998, 167.179993, 115.839996,
       116.730003, 134.100006, 168.5     , 159.570007, 138.220001,
       155.360001, 126.610001, 177.210007, 157.339996, 125.459999,
        93.720001,  73.080002, 122.290001, 119.220001, 167.990005,
        87.370003, 124.769997,  91.730003, 117.919998, 121.300003,
       135.410004, 114.57    , 115.800003, 134.119995, 105.720001,
       104.370003,  87.239998, 127.489998, 118.360001, 157.779999,
       119.959999, 108.419998, 107.839996,  93.459999, 128.539993,
        72.510002, 113.260002, 120.110001, 109.860001, 118.919998,
       120.730003, 161.320007, 161.520004, 146.869995,  85.199997,
        93.040001,  86.879997,  90.949997, 119.800003, 126.139999,
       127.400002, 173.490005, 122.970001, 116.209999,  97.550003,
       170.130005, 142.050003, 132.490005, 120.910004, 122.879997,
       119.190002, 113.910004, 118.82    , 106.260002, 127.660004,
       114.769997, 108.470001, 115.57    , 119.699997,  89.910004,
        87.989998, 142.380005, 127.150002, 114.209999, 110.239998,
       108.279999,  73.080002, 173.020004, 113.639999, 121.209999,
       128.5     , 155.139999,  91.769997, 134.410004, 161.320007,
       124.43    , 126.18    , 130.369995, 113.910004, 121.160004,
        92.059998, 111.459999, 173.490005, 160.289993, 113.639999,
       107.040001,  81.989998, 113.669998, 125.540001, 108.089996,
       118.82    , 157.320007, 160.559998, 120.139999, 132.850006,
       101.849998, 116.029999, 118.959999, 112.440002, 102.309998,
       160.559998,  97.43    , 139.350006, 125.290001, 171.509995,
       123.730003, 128.539993, 128.380005, 115.199997, 112.139999,
       124.269997, 101.790001,  89.5     , 125.959999,  99.669998,
       107.040001, 115.849998, 116.730003,  97.699997, 121.650002,
       165.279999,  87.089996, 107.669998, 118.559998, 127.660004,
       125.18    ,  79.790001, 121.470001, 160.619995,  89.5     ,
       109.760002, 118.279999, 174.399994, 102.459999, 104.860001,
       122.599998, 160.619995,  86.610001,  92.760002, 111.57    ,
       177.210007, 114.720001, 118.959999,  93.860001, 124.910004,
       166.630005, 113.910004, 117.260002,  88.419998, 150.410004,
       119.75    ,  89.440002, 112.650002, 118.360001, 118.860001,
        87.690002,  94.599998, 114.459999, 118.230003, 120.839996,
       126.160004, 122.080002, 138.      , 165.649994, 119.32    ,
       120.730003, 151.440002, 120.559998, 174.399994, 107.43    ,
       105.169998, 150.410004, 114.769997, 124.360001, 146.589996,
       119.730003, 113.639999, 112.650002, 112.220001, 132.690002,
       118.099998, 103.150002, 115.940002, 104.370003,  98.360001,
       118.360001,  89.440002,  91.980003, 154.449997, 102.459999,
       154.470001, 115.050003, 137.660004,  90.589996, 115.139999,
       108.860001, 119.510002, 122.709999, 166.490005,  93.389999,
       137.660004, 121.110001, 120.160004, 105.68    , 135.020004,
       122.169998, 116.470001, 113.5     , 128.789993, 123.32    ,
       125.699997, 121.110001,  87.239998, 132.070007, 135.380005,
        91.5     , 161.520004, 161.320007, 126.949997, 168.110001,
       110.290001, 108.550003, 103.019997,  94.730003, 129.740005,
       107.040001, 161.520004, 121.650002, 131.740005, 130.369995,
       157.210007,  90.610001, 177.210007, 129.860001, 126.160004,
        84.769997, 124.019997, 150.479996,  89.440002, 107.669998,
       108.470001,  88.32    , 134.300003, 156.710007, 135.020004,
        72.5     , 151.440002, 126.809998, 127.169998, 127.970001,
       108.470001, 157.160004, 113.910004, 116.730003, 123.389999,
       154.      , 122.760002, 156.479996,  92.93    , 125.379997,
       124.239998,  88.580002,  91.610001, 125.620003, 129.710007,
       112.589996, 116.839996, 120.779999, 128.789993, 120.730003,
       136.050003,  92.949997, 118.809998, 112.940002,  94.599998,
       109.129997,  88.32    , 108.949997,  90.809998,  93.389999,
       130.369995, 161.419998,  88.32    , 120.139999, 134.25    ,
       124.389999, 129.710007, 101.419998,  91.309998, 130.800003,
       123.32    , 108.220001, 173.199997, 113.290001,  86.510002,
```

```
 118.540001,  91.25    , 161.589996, 118.360001, 123.620003,
159.429993, 120.559998, 114.269997, 108.309998, 126.800003,
 75.650002, 103.18    , 129.740005, 121.589996,  92.730003,
132.199997, 118.559998, 117.099998, 154.470001, 161.220001,
110.809998, 137.380005, 119.330002, 160.559998, 117.449997,
159.050003, 114.860001, 116.5     , 147.179993, 115.43    ,
127.779999, 167.339996, 117.139999, 124.779999, 152.300003,
153.050003, 132.009995, 115.43    , 121.559998, 129.740005,
 90.279999, 125.32    , 157.320007, 110.459999, 107.669998,
162.300003, 118.769997, 165.800003, 134.75    , 113.269997,
152.589996, 169.610001, 117.769997, 114.720001, 161.449997,
 84.459999, 127.120003, 128.270004, 128.110001, 124.529999,
124.400002,  90.800003, 151.910004,  96.5     , 136.050003,
 86.449997, 108.470001, 115.      , 111.75    , 124.959999,
 91.610001, 124.589996, 161.419998, 121.239998, 167.580002,
126.160004, 112.970001, 127.599998,  95.449997,  94.349998,
102.279999, 120.739998,  84.279999, 126.589996, 160.460007,
116.610001, 118.68    , 117.610001, 124.389999, 120.720001,
120.940002, 117.650002, 107.040001, 146.869995, 123.730003,
115.32    ,  74.      , 127.349998, 157.779999, 118.      ,
125.540001,  89.269997, 102.199997, 123.389999, 121.      ,
 73.790001, 151.050003, 120.110001, 103.93    ,  88.949997,
114.769997, 172.289993, 121.470001, 160.649994, 113.669998,
121.790001, 118.559998,  96.5     , 119.169998, 126.07    ,
118.239998,  97.800003, 157.779999, 123.620003, 146.      ,
159.300003, 114.730003, 123.239998, 145.729996, 128.669998,
166.130005, 132.690002, 120.980003, 173.020004, 109.199997,
122.019997, 135.410004, 103.419998])
```

In [33]:
```
# R squared error
error_score = metrics.r2_score(Y_test, test_data_prediction)

print("R squared error : ", error_score)
```

```
R squared error :  0.9897511997931183
```

In [34]:
```
regressor.score(X_test, Y_test)
```

Out[34]:
```
0.9854190298916252
```

In [ ]:
```

```

Practical 4

A. For a given set of training data examples stored in a .CSVfile implement Least Square Regression algorithm.

In [1]:
```
# Import Libary
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
# read data
df=pd.read_csv('headbrain.csv')
```

In [3]:
```
df.head()
```

Out[3]:

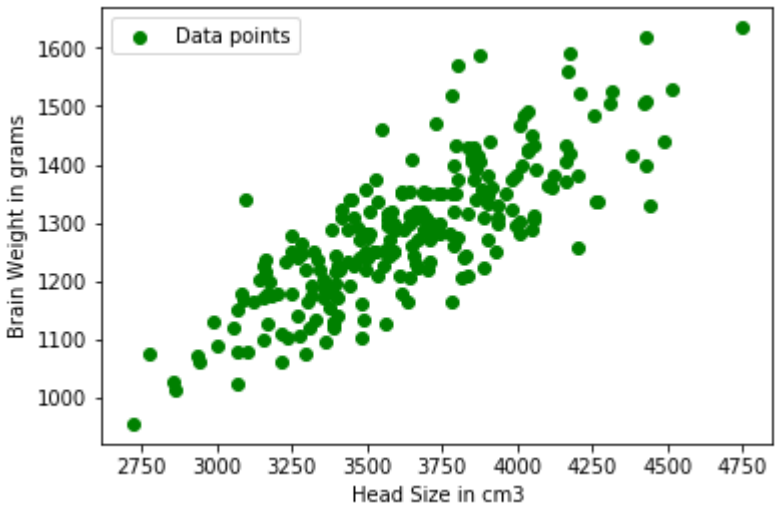
	Gender	Age Range	Head Size(cm^3)	Brain Weight(grams)
0	1	1	4512	1530
1	1	1	3738	1297
2	1	1	4261	1335
3	1	1	3777	1282
4	1	1	4177	1590

In [4]:
```
# Declare dependent variable(Y) and independent variable(X)

X=df['Head  Size(cm^3)'].values
Y = df['Brain Weight(grams)'].values
```

In [5]:
```
np.corrcoef(X, Y)
```

Out[5]:
```
array([[1.        , 0.79956971],
       [0.79956971, 1.        ]])
```

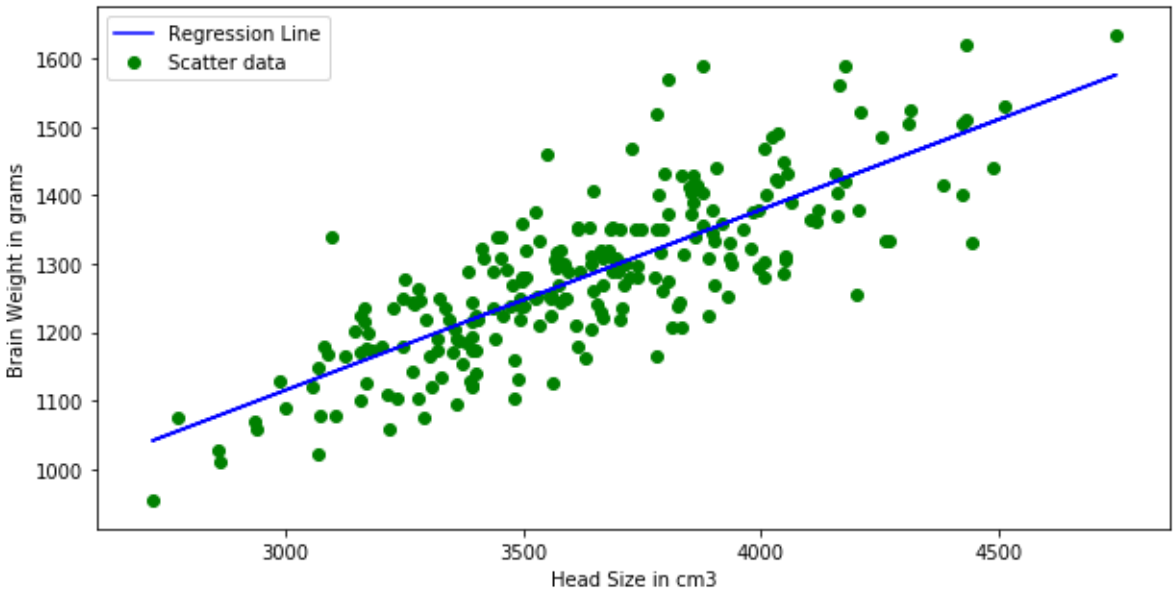
In [6]:
```
# Plot the Input Data
plt.scatter(X, Y, c='green', label='Data points')
plt.xlabel('Head Size in cm3')
plt.ylabel('Brain Weight in grams')
plt.legend()
plt.show()
```



In [7]:
```
# Calculating coefficient

# Mean X and Y
mean_x = np.mean(X)
mean_y = np.mean(Y)

# Total number of values
n = len(X)

# Using the formula to calculate theta1 and theta2
numer = 0
denom = 0
for i in range(n):
    numer += (X[i] - mean_x) * (Y[i] - mean_y)
    denom += (X[i] - mean_x) ** 2
b1 = numer / denom
b0 = mean_y - (b1 * mean_x)

# Printing coefficients
print("coefficients for regression",b1, b0)
```

```
coefficients for regression 0.26342933948939945 325.57342104944223
```

In [8]:
```
# Plotting Values and Regression Line
%matplotlib inline

plt.rcParams['figure.figsize'] = (10.0, 5.0)
# max_x = np.max(X) + 100
# min_x = np.min(X) - 100

y = b0 + b1 * X

# Ploting Line
plt.plot(X, y, color='blue', label='Regression Line')
# Ploting Scatter Points
plt.scatter(X, Y, c='green', label='Scatter data')

plt.xlabel('Head Size in cm3')
plt.ylabel('Brain Weight in grams')
plt.legend()
plt.show()
```



In [9]:
```
# Calculating Root Mean Squares Error
rmse = 0
for i in range(n):
    y_pred = b0 + b1 * X[i]
    rmse += (Y[i] - y_pred) ** 2

rmse = np.sqrt(rmse/n)
print("Root Mean Square Error is",rmse)
```

```
Root Mean Square Error is 72.1206213783709
```

In [10]:
```
# Calculating R2 Score
ss_tot = 0
ss_res = 0
for i in range(n):
    y_pred = b0 + b1 * X[i]
    ss_tot += (Y[i] - mean_y) ** 2
    ss_res += (Y[i] - y_pred) ** 2
r2 = 1 - (ss_res/ss_tot)
print("R2 Score",r2)
```

```
R2 Score 0.6393117199570003
```

In [ ]:

B. For a given set of training data examples storedin a .CSV file implement Logistic Regression algorithm.

In [12]:
```
#import pandas
import pandas as pd
#col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'd
# load dataset
pima = pd.read_csv("diabetes.csv")
```

In [13]:
```
pima.head()
```

Out[13]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6	0.627	
1	1	85	66	29	0	26.6	0.351	
2	8	183	64	0	0	23.3	0.672	
3	1	89	66	23	94	28.1	0.167	
4	0	137	40	35	168	43.1	2.288	

In [15]:
```
#split dataset in features and target variable
feature_cols = ['Pregnancies', 'Insulin', 'BMI', 'Age','Glucose','BloodPressure','
X = pima[feature_cols] # Features
y = pima.Outcome # Target variable
```

In [17]:
```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)
```

Model Development and Prediction

First, import the Logistic Regression module and create a Logistic Regression classifier object using LogisticRegression() function.

Then, fit your model on the train set using fit() and perform prediction on the test set using predict().

In [23]:
```
# import the class
from sklearn.linear_model import LogisticRegression
```

In [24]:
```
# instantiate the model (using the default parameters)
logreg = LogisticRegression()

# fit the model with data
logreg.fit(X_train,y_train)
```

```
C:\Users\h103196\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:94
0: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[24]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [25]:
```
# predict the model
y_pred=logreg.predict(X_test)
```

In [ ]:
```

```

Model Evaluation using Confusion Matrix

A confusion matrix is a table that is used to evaluate the performance of a classification model. You can also visualize the performance of an algorithm.

The fundamental of a confusion matrix is the number of correct and incorrect predictions are summed up class-wise.

In [26]:
```
# import the metrics class
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[26]:
```
array([[117,  13],
       [ 24,  38]], dtype=int64)
```

Here, you can see the confusion matrix in the form of the array object. The dimension of this matrix is 2*2 because this model is binary classification.

You have two classes 0 and 1. Diagonal values represent accurate predictions, while non-diagonal elements are inaccurate predictions.

In the output, 117 and 38 are actual predictions, and 24 and 13 are incorrect predictions.

Visualizing Confusion Matrix using Heatmap

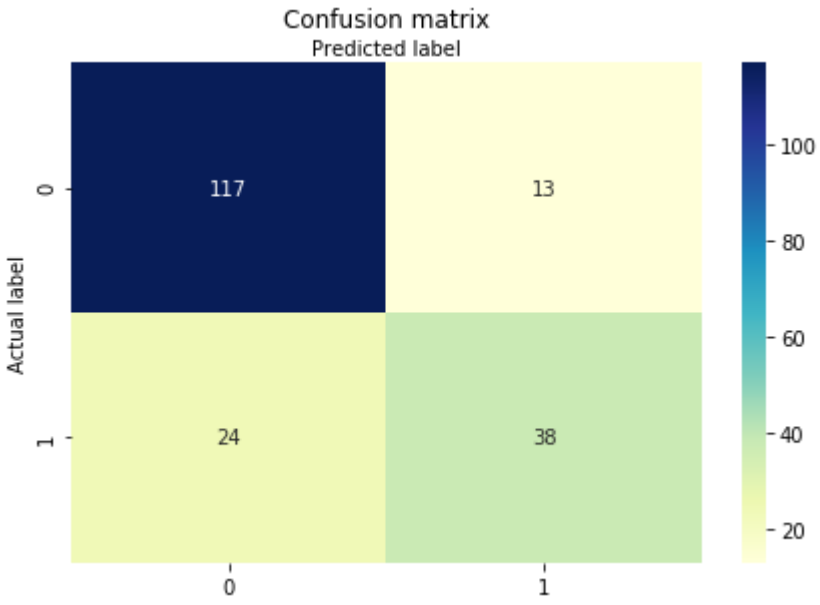
Let's visualize the results of the model in the form of a confusion matrix using matplotlib and seaborn.

In [27]:
```
# import required modules
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [21]:
```
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
```

```
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[21]:
```
Text(0.5, 257.44, 'Predicted label')
```



Confusion Matrix Evaluation Metrics

Let's evaluate the model using model evaluation metrics such as accuracy, precision, and recall.

In [22]:
```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.8072916666666666
Precision: 0.7450980392156863
Recall: 0.6129032258064516
```

Well, you got a classification rate of 80%, considered as good accuracy.

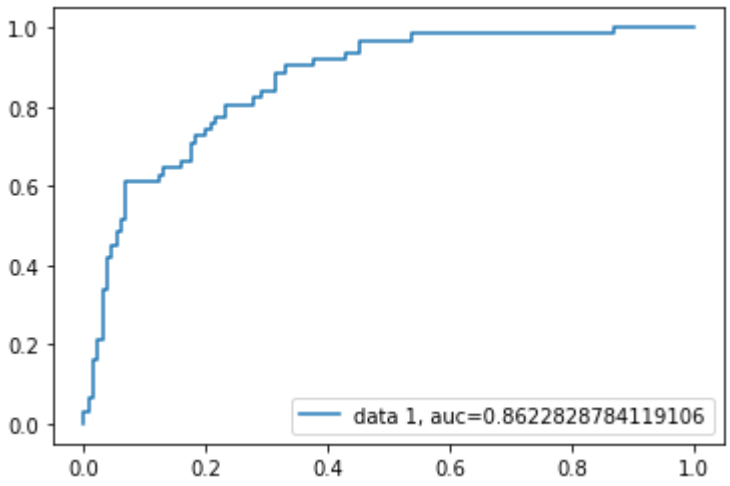
Precision: Precision is about being precise, i.e., how accurate your model is. In other words, you can say, when a model makes a prediction, how often it is correct. In your prediction case, when your Logistic Regression model predicted patients are going to suffer from diabetes, that patients have 74% of the time.

Recall: If there are patients who have diabetes in the test set and your Logistic Regression model can identify it 61% of the time.

ROC Curve

Receiver Operating Characteristic(ROC) curve is a plot of the true positive rate against the false positive rate. It shows the tradeoff between sensitivity and specificity.

In [28]:
```
y_pred_proba  =  logreg.predict_proba(X_test)[::,1]

fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)

auc = metrics.roc_auc_score(y_test, y_pred_proba)

plt.plot(fpr,tpr,label="data  1,  auc="+str(auc))
plt.legend(loc=4)
plt.show()
```



AUC score for the case is 0.86. AUC score 1 represents perfect classifier, and 0.5 represents a worthless classifier.

In [ ]:

Practical 5

A. Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data setfor building the decision tree and apply this knowledge to classify a new sample.

In [17]:
```
#Importing important libraries

import pandas as pd

from pandas import DataFrame

#Reading Dataset

df_tennis  =  pd.read_csv('/content/PlayTennis.csv')

print( df_tennis)
```

	Unnamed: 0	PlayTennis	Outlook	Temperature	Humidity	Wind
0	0	No	Sunny	Hot	High	Weak
1	1	No	Sunny	Hot	High	Strong
2	2	Yes	Overcast	Hot	High	Weak
3	3	Yes	Rain	Mild	High	Weak
4	4	Yes	Rain	Cool	Normal	Weak
5	5	No	Rain	Cool	Normal	Strong
6	6	Yes	Overcast	Cool	Normal	Strong
7	7	No	Sunny	Mild	High	Weak
8	8	Yes	Sunny	Cool	Normal	Weak
9	9	Yes	Rain	Mild	Normal	Weak
10	10	Yes	Sunny	Mild	Normal	Strong
11	11	Yes	Overcast	Mild	High	Strong
12	12	Yes	Overcast	Hot	Normal	Weak
13	13	No	Rain	Mild	High	Strong

In [18]:
```
def entropy(probs):
    import math
    return sum( [-prob*math.log(prob, 2) for prob in probs] )

#Function to calulate the entropy of the given Data Sets/List with respect to targe
def entropy_of_list(a_list):
    #print("A-list",a_list)
    from collections import Counter
    cnt = Counter(x for x in a_list)   # Counter calculates the propotion of class
    num_instances = len(a_list)*1.0   # = 14
    print("\n Number of Instances of the Current Sub Class is {0}:".format(num_inst
    probs = [x / num_instances for x in cnt.values()]  # x means no of YES/NO
    print("\n  Classes:",min(cnt),max(cnt))
    print(" \n Probabilities of Class {0} is {1}:".format(min(cnt),min(probs)))
    print(" \n Probabilities of Class {0} is {1}:".format(max(cnt),max(probs)))
    return entropy(probs) # Call Entropy :

# The initial entropy of the YES/NO attribute for our dataset.
print("\n  INPUT DATA SET FOR ENTROPY CALCULATION:\n", df_tennis['PlayTennis'])

total_entropy = entropy_of_list(df_tennis['PlayTennis'])

print("\n Total Entropy of PlayTennis Data Set:",total_entropy)
```

```
  INPUT DATA SET FOR ENTROPY CALCULATION:
 0      No
1       No
2      Yes
3      Yes
4      Yes
5       No
6      Yes
7       No
8      Yes
9      Yes
10     Yes
11     Yes
12     Yes
13      No
Name: PlayTennis, dtype: object

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:

 Total Entropy of PlayTennis Data Set: 0.9402859586706309
```

In [19]:

```python
def information_gain(df, split_attribute_name, target_attribute_name, trace=0):
    print("Information Gain Calculation of ",split_attribute_name)
    '''
    Takes a DataFrame of attributes, and quantifies the entropy of a target
    attribute after performing a split along the values of another attribute.
    '''
    # Split Data by Possible Vals of Attribute:
    df_split = df.groupby(split_attribute_name)

    # Calculate Entropy for Target Attribute, as well as
    # Proportion of Obs in Each Data-Split
    nobs = len(df.index) * 1.0

    df_agg_ent = df_split.agg({target_attribute_name : [entropy_of_list, lambda x:

    df_agg_ent.columns = ['Entropy', 'PropObservations']
    #if trace: # helps understand what fxn is doing:
     #   print(df_agg_ent)


    # Calculate Information Gain:
    new_entropy = sum( df_agg_ent['Entropy'] * df_agg_ent['PropObservations'] )
    old_entropy = entropy_of_list(df[target_attribute_name])
    return old_entropy - new_entropy


print('Info-gain for Outlook is :'+str( information_gain(df_tennis, 'Outlook', 'Pla
print('\n Info-gain for Humidity is: ' + str( information_gain(df_tennis, 'Humidity
print('\n Info-gain for Wind is:' + str( information_gain(df_tennis, 'Wind', 'PlayT
print('\n Info-gain for Temperature is:' + str( information_gain(df_tennis, 'Temper
```

```
Information Gain Calculation of  Outlook

 Number of Instances of the Current Sub Class is 4.0:

 Classes: Yes Yes

 Probabilities of Class Yes is 1.0:

 Probabilities of Class Yes is 1.0:

 Number of Instances of the Current Sub Class is 5.0:

 Classes: No Yes

 Probabilities of Class No is 0.4:

 Probabilities of Class Yes is 0.6:

 Number of Instances of the Current Sub Class is 5.0:

 Classes: No Yes

 Probabilities of Class No is 0.4:

 Probabilities of Class Yes is 0.6:

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:
Info-gain for Outlook is :0.2467498197744391

Information Gain Calculation of  Humidity

 Number of Instances of the Current Sub Class is 7.0:

 Classes: No Yes

 Probabilities of Class No is 0.42857142857142855:

 Probabilities of Class Yes is 0.5714285714285714:

 Number of Instances of the Current Sub Class is 7.0:

 Classes: No Yes

 Probabilities of Class No is 0.14285714285714285:

 Probabilities of Class Yes is 0.8571428571428571:

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:

 Info-gain for Humidity is: 0.15183550136234136

Information Gain Calculation of  Wind
```

Number of Instances of the Current Sub Class is 6.0:

Classes: No Yes

Probabilities of Class No is 0.5:

Probabilities of Class Yes is 0.5:

Number of Instances of the Current Sub Class is 8.0:

Classes: No Yes

Probabilities of Class No is 0.25:

Probabilities of Class Yes is 0.75:

Number of Instances of the Current Sub Class is 14.0:

Classes: No Yes

Probabilities of Class No is 0.35714285714285715:

Probabilities of Class Yes is 0.6428571428571429:

Info-gain for Wind is:0.04812703040826927

Information Gain Calculation of Temperature

Number of Instances of the Current Sub Class is 4.0:

Classes: No Yes

Probabilities of Class No is 0.25:

Probabilities of Class Yes is 0.75:

Number of Instances of the Current Sub Class is 4.0:

Classes: No Yes

Probabilities of Class No is 0.5:

Probabilities of Class Yes is 0.5:

Number of Instances of the Current Sub Class is 6.0:

Classes: No Yes

Probabilities of Class No is 0.3333333333333333:

Probabilities of Class Yes is 0.6666666666666666:

Number of Instances of the Current Sub Class is 14.0:

Classes: No Yes

Probabilities of Class No is 0.35714285714285715:

Probabilities of Class Yes is 0.6428571428571429:

Info-gain for Temperature is:0.029222565658954647

In [20]:
```
def id3(df, target_attribute_name, attribute_names, default_class=None):

    ## Tally target attribute:
    from collections import Counter
    cnt = Counter(x for x in df[target_attribute_name])# class of YES /NO

    ## First check: Is this split of the dataset homogeneous?
    if len(cnt) == 1:
        return next(iter(cnt))  # next input data set, or raises StopIteration when

    ## Second check: Is this split of the dataset empty?
    # if yes, return a default value
    elif df.empty or (not attribute_names):
        return default_class  # Return None for Empty Data Set

    ## Otherwise: This dataset is ready to be devied up!
    else:
        # Get Default Value for next recursive call of this function:
        default_class = max(cnt.keys()) #No of YES and NO Class
        # Compute the Information Gain of the attributes:
        gainz = [information_gain(df, attr, target_attribute_name) for attr in att
        index_of_max = gainz.index(max(gainz)) # Index of Best Attribute
        # Choose Best Attribute to split on:
        best_attr = attribute_names[index_of_max]

        # Create an empty tree, to be populated in a moment
        tree = {best_attr:{}} # Iniiate the tree with best attribute as a node
        remaining_attribute_names = [i for i in attribute_names if i != best_attr]

        # Split dataset
        # On each split, recursively call this algorithm.
        # populate the empty tree with subtrees, which
        # are the result of the recursive call
        for attr_val, data_subset in df.groupby(best_attr):
            subtree = id3(data_subset,
                          target_attribute_name,
                          remaining_attribute_names,
                          default_class)
            tree[best_attr][attr_val] = subtree
        return tree
```

Predicting from all Attributes given

In [21]:
```
# Get Predictor Names (all but 'class')
attribute_names = list(df_tennis.columns)
print("List of Attributes:", attribute_names)
attribute_names.remove('PlayTennis') #Remove the class attribute
print("Predicting Attributes:", attribute_names)
```

```
List of Attributes: ['Unnamed: 0', 'PlayTennis', 'Outlook', 'Temperature', 'Humidi
ty', 'Wind']
Predicting Attributes: ['Unnamed: 0', 'Outlook', 'Temperature', 'Humidity', 'Win
d']
```

In [22]:
```
# Run Algorithm:
from pprint import pprint
tree = id3(df_tennis,'PlayTennis',attribute_names)
print("\n\nThe Resultant Decision Tree is :\n")
#print(tree)
pprint(tree)
attribute = next(iter(tree))
print("Best Attribute :\n",attribute)
print("Tree Keys:\n",tree[attribute].keys())
```

nformation Gain Calculation of Unnamed: 0

Number of Instances of the Current Sub Class is 1.0:

Classes: No No

Probabilities of Class No is 1.0:

Probabilities of Class No is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: No No

Probabilities of Class No is 1.0:

Probabilities of Class No is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: No No

Probabilities of Class No is 1.0:

Probabilities of Class No is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: No No

Probabilities of Class No is 1.0:

Probabilities of Class No is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: No No

Probabilities of Class No is 1.0:

Probabilities of Class No is 1.0:

Number of Instances of the Current Sub Class is 14.0:

Classes: No Yes

Probabilities of Class No is 0.35714285714285715:

Probabilities of Class Yes is 0.6428571428571429:
Information Gain Calculation of Outlook

Number of Instances of the Current Sub Class is 4.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 5.0:

Classes: No Yes

Probabilities of Class No is 0.4:

Probabilities of Class Yes is 0.6:

Number of Instances of the Current Sub Class is 5.0:

Classes: No Yes

Probabilities of Class No is 0.4:

Probabilities of Class Yes is 0.6:

Number of Instances of the Current Sub Class is 14.0:

Classes: No Yes

Probabilities of Class No is 0.35714285714285715:

Probabilities of Class Yes is 0.6428571428571429:

Information Gain Calculation of Temperature

Number of Instances of the Current Sub Class is 4.0:

Classes: No Yes

Probabilities of Class No is 0.25:

Probabilities of Class Yes is 0.75:

Number of Instances of the Current Sub Class is 4.0:

Classes: No Yes

Probabilities of Class No is 0.5:

Probabilities of Class Yes is 0.5:

Number of Instances of the Current Sub Class is 6.0:

Classes: No Yes

Probabilities of Class No is 0.3333333333333333:

Probabilities of Class Yes is 0.6666666666666666:

Number of Instances of the Current Sub Class is 14.0:

Classes: No Yes

Probabilities of Class No is 0.35714285714285715:

Probabilities of Class Yes is 0.6428571428571429:

Information Gain Calculation of Humidity

Number of Instances of the Current Sub Class is 7.0:

Classes: No Yes

```
 Probabilities of Class No is 0.42857142857142855:

 Probabilities of Class Yes is 0.5714285714285714:

 Number of Instances of the Current Sub Class is 7.0:

 Classes: No Yes

 Probabilities of Class No is 0.14285714285714285:

 Probabilities of Class Yes is 0.8571428571428571:

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:
Information Gain Calculation of  Wind

 Number of Instances of the Current Sub Class is 6.0:

 Classes: No Yes

 Probabilities of Class No is 0.5:

 Probabilities of Class Yes is 0.5:

 Number of Instances of the Current Sub Class is 8.0:

 Classes: No Yes

 Probabilities of Class No is 0.25:

 Probabilities of Class Yes is 0.75:

 Number of Instances of the Current Sub Class is 14.0:

 Classes: No Yes

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:


The Resultant Decision Tree is :

{'Unnamed: 0': {0: 'No',
                1: 'No',
                2: 'Yes',
                3: 'Yes',
                4: 'Yes',
                5: 'No',
                6: 'Yes',
                7: 'No',
                8: 'Yes',
                9: 'Yes',
                10: 'Yes',
                11: 'Yes',
                12: 'Yes',
                13: 'No'}}
Best Attribute :
```

```
 Unnamed: 0
Tree Keys:
 dict_keys([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13])
```

Check Accuracy for the same

Classification Accuracy for Training and Testing Set

In [23]:
```
training_data = df_tennis.iloc[1:-4] # all but last four instances
test_data  = df_tennis.iloc[-4:] # just the last four
train_tree  =  id3(training_data,  'PlayTennis',  attribute_names)

test_data['predicted2'] = test_data.apply(classify, # <----  test_data  source
                                          axis=1,
                                          args=(train_tree,'Yes') ) # <------ train_d


print ('\n\n Accuracy is : ' + str( sum(test_data['PlayTennis']==test_data['predic
```

Information Gain Calculation of Unnamed: 0

Number of Instances of the Current Sub Class is 1.0:

Classes: No No

Probabilities of Class No is 1.0:

Probabilities of Class No is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: No No

Probabilities of Class No is 1.0:

Probabilities of Class No is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: No No

Probabilities of Class No is 1.0:

Probabilities of Class No is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 1.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 9.0:

Classes: No Yes

Probabilities of Class No is 0.3333333333333333:

Probabilities of Class Yes is 0.6666666666666666:

Information Gain Calculation of Outlook

Number of Instances of the Current Sub Class is 2.0:

Classes: Yes Yes

Probabilities of Class Yes is 1.0:

Probabilities of Class Yes is 1.0:

Number of Instances of the Current Sub Class is 4.0:

Classes: No Yes

Probabilities of Class No is 0.25:

Probabilities of Class Yes is 0.75:

Number of Instances of the Current Sub Class is 3.0:

Classes: No Yes

Probabilities of Class No is 0.3333333333333333:

Probabilities of Class Yes is 0.6666666666666666:

Number of Instances of the Current Sub Class is 9.0:

Classes: No Yes

Probabilities of Class No is 0.3333333333333333:

Probabilities of Class Yes is 0.6666666666666666:

Information Gain Calculation of Temperature

Number of Instances of the Current Sub Class is 4.0:

Classes: No Yes

Probabilities of Class No is 0.25:

Probabilities of Class Yes is 0.75:

Number of Instances of the Current Sub Class is 2.0:

Classes: No Yes

Probabilities of Class No is 0.5:

Probabilities of Class Yes is 0.5:

Number of Instances of the Current Sub Class is 3.0:

Classes: No Yes

Probabilities of Class No is 0.3333333333333333:

Probabilities of Class Yes is 0.6666666666666666:

Number of Instances of the Current Sub Class is 9.0:

Classes: No Yes

Probabilities of Class No is 0.3333333333333333:

Probabilities of Class Yes is 0.6666666666666666:

Information Gain Calculation of Humidity

Number of Instances of the Current Sub Class is 4.0:

Classes: No Yes

Probabilities of Class No is 0.5:

Probabilities of Class Yes is 0.5:

Number of Instances of the Current Sub Class is 5.0:

Classes: No Yes

Probabilities of Class No is 0.2:

Probabilities of Class Yes is 0.8:

Number of Instances of the Current Sub Class is 9.0:

Classes: No Yes

Probabilities of Class No is 0.3333333333333333:

Probabilities of Class Yes is 0.6666666666666666:

Information Gain Calculation of Wind

Number of Instances of the Current Sub Class is 3.0:

Classes: No Yes

Probabilities of Class No is 0.3333333333333333:

Probabilities of Class Yes is 0.6666666666666666:

Number of Instances of the Current Sub Class is 6.0:

Classes: No Yes

Probabilities of Class No is 0.16666666666666666:

Probabilities of Class Yes is 0.8333333333333334:

Number of Instances of the Current Sub Class is 9.0:

```
 Classes: No Yes

 Probabilities of Class No is 0.3333333333333333:

 Probabilities of Class Yes is 0.6666666666666666:
Key: dict_keys(['Unnamed: 0'])
Attribute: Unnamed: 0
Key: dict_keys(['Unnamed: 0'])
Attribute: Unnamed: 0
Key: dict_keys(['Unnamed: 0'])
Attribute: Unnamed: 0
Key: dict_keys(['Unnamed: 0'])
Attribute: Unnamed: 0


 Accuracy is : 0.75
<ipython-input-23-8576c8cc24da>:5:  SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_data['predicted2'] = test_data.apply(classify, # <---- test_data source
```

The Accuracy is 0.75 ie, 75%

In [23]:

B. Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set

In [38]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [39]:
```
url  =  "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# Assign colum names to the dataset
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']

# Read dataset to pandas dataframe
dataset = pd.read_csv(url, names=names)
```

In [40]:
```
dataset.head()
```

Out[40]:

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [41]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal-length  150 non-null    float64
 1   sepal-width   150 non-null    float64
 2   petal-length  150 non-null    float64
 3   petal-width   150 non-null    float64
 4   Class         150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [42]:
```
dataset.describe()
```

Out[42]:

	sepal-length	sepal-width	petal-length	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [43]:
```
dataset.isnull().sum()
```

Out[43]:
```
sepal-length    0
sepal-width     0
petal-length    0
petal-width     0
Class           0
dtype: int64
```

In [44]:
```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values
```

The X variable contains the first four columns of the dataset (i.e. attributes) while y contains the labels.

Train Test Split

In [45]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

Feature Scaling

In [46]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [47]:
```
X_train
```

Out[47]:
```
array([[-0.39871148,  1.08700609, -1.40772509, -1.30672051],
       [-0.28028233, -0.13664054,  0.17738091,  0.12708753],
       [-0.87242809,  1.82119407, -1.29450323, -1.17637432],
       [ 0.90400919, -0.38136987,  0.46043555,  0.12708753],
       [-1.10928639,  0.10808879, -1.29450323, -1.43706669],
       [-0.04342402, -0.87082852,  0.74349019,  0.90916464],
       [-1.583003  , -1.84974582, -1.40772509, -1.17637432],
       [-0.75399893,  2.55538205, -1.29450323, -1.43706669],
       [ 0.19343428, -0.13664054,  0.57365741,  0.77881846],
       [-1.70143215,  0.35281811, -1.40772509, -1.30672051],
       [-0.28028233, -0.13664054,  0.40382462,  0.3877799 ],
       [-0.99085724, -0.13664054, -1.2378923 , -1.30672051],
       [ 0.19343428, -2.09447515,  0.68687927,  0.3877799 ],
       [ 0.19343428, -0.38136987,  0.40382462,  0.3877799 ],
       [-0.04342402, -0.87082852,  0.17738091, -0.26395103],
       [-1.22771554,  0.10808879, -1.2378923 , -1.30672051],
       [ 1.25929665,  0.10808879,  0.74349019,  1.43054938],
       [-0.87242809,  1.82119407, -1.06805952, -1.04602814],
       [-0.39871148, -1.84974582,  0.12076998,  0.12708753],
       [ 1.49615495, -0.13664054,  1.19637762,  1.16985701],
       [ 1.02243834,  0.10808879,  0.51704648,  0.3877799 ],
       [-0.99085724,  1.08700609, -1.40772509, -1.17637432],
       [-0.99085724, -2.5839338 , -0.16228466, -0.26395103],
       [-0.75399893,  0.84227676, -1.35111416, -1.30672051],
       [ 1.6145841 ,  1.33173542,  1.30959948,  1.69124175],
       [-0.63556978,  1.57646474, -1.29450323, -1.30672051],
       [ 1.02243834,  0.59754744,  1.08315577,  1.69124175],
       [ 0.31186343, -0.38136987,  0.51704648,  0.25743372],
       [-1.22771554, -0.13664054, -1.35111416, -1.43706669],
       [ 2.44358817,  1.82119407,  1.47943226,  1.03951083],
       [-1.46457384,  0.84227676, -1.35111416, -1.17637432],
       [ 0.54872174, -0.87082852,  0.63026834,  0.77881846],
       [-1.10928639, -1.36028717,  0.40382462,  0.64847227],
       [-0.99085724,  1.08700609, -1.2378923 , -0.78533577],
       [ 1.73301326, -0.38136987,  1.42282134,  0.77881846],
       [-0.99085724,  0.84227676, -1.2378923 , -1.04602814],
       [-1.22771554,  0.84227676, -1.2378923 , -1.30672051],
       [-0.16185317, -1.11555785, -0.16228466, -0.26395103],
       [-0.51714063,  2.0659234 , -1.40772509, -1.04602814],
       [-0.87242809,  0.59754744, -1.18128137, -0.91568195],
       [-1.8198613 , -0.13664054, -1.52094695, -1.43706669],
       [ 0.54872174,  0.59754744,  1.25298855,  1.69124175],
       [ 1.02243834, -0.13664054,  0.80010112,  1.43054938],
       [-1.34614469,  0.35281811, -1.40772509, -1.30672051],
       [ 1.6145841 , -0.13664054,  1.13976669,  0.51812609],
       [ 0.43029259,  0.84227676,  0.91332298,  1.43054938],
       [ 1.02243834,  0.59754744,  1.08315577,  1.16985701],
       [-0.75399893, -0.87082852,  0.06415905,  0.25743372],
       [-0.39871148, -1.36028717,  0.12076998,  0.12708753],
       [ 0.78558004, -0.62609919,  0.46043555,  0.3877799 ],
       [-0.51714063, -0.13664054,  0.40382462,  0.3877799 ],
       [ 1.6145841 ,  0.35281811,  1.25298855,  0.77881846],
       [-0.51714063,  1.57646474, -1.29450323, -1.30672051],
       [ 1.85144241, -0.62609919,  1.30959948,  0.90916464],
       [ 0.31186343, -0.62609919,  0.12076998,  0.12708753],
       [-0.16185317, -1.36028717,  0.68687927,  1.03951083],
       [-0.16185317, -0.13664054,  0.23399184, -0.00325865],
       [-1.46457384,  0.35281811, -1.35111416, -1.30672051],
       [-1.10928639,  0.10808879, -1.29450323, -1.43706669],
       [ 1.02243834, -1.36028717,  1.13976669,  0.77881846],
       [ 1.25929665,  0.10808879,  0.91332298,  1.16985701],
       [ 0.78558004, -0.13664054,  0.96993391,  0.77881846],
       [ 0.31186343, -0.13664054,  0.46043555,  0.25743372],
       [-0.04342402, -0.87082852,  0.06415905, -0.00325865],
```

```
       [ 0.66715089,  0.35281811,  0.85671205,  1.43054938],
       [-0.16185317, -0.62609919,  0.40382462,  0.12708753],
       [ 0.66715089, -0.87082852,  0.85671205,  0.90916464],
       [-0.39871148, -1.6050165 ,  0.00754812, -0.13360484],
       [ 0.43029259, -0.38136987,  0.29060277,  0.12708753],
       [-0.75399893,  1.08700609, -1.29450323, -1.30672051],
       [ 0.78558004, -0.13664054,  0.80010112,  1.03951083],
       [-0.99085724,  1.33173542, -1.35111416, -1.30672051],
       [ 0.54872174,  0.84227676,  1.02654484,  1.56089557],
       [ 1.1408675 , -0.62609919,  0.57365741,  0.25743372],
       [ 0.66715089, -0.38136987,  0.29060277,  0.12708753],
       [ 0.31186343, -1.11555785,  1.02654484,  0.25743372],
       [ 0.19343428,  0.84227676,  0.40382462,  0.51812609],
       [-0.87242809,  1.57646474, -1.29450323, -1.04602814],
       [ 2.20672986,  1.82119407,  1.64926505,  1.3002032 ],
       [ 0.66715089,  0.10808879,  0.96993391,  0.77881846],
       [-0.99085724, -1.84974582, -0.27550652, -0.26395103],
       [ 2.20672986, -1.11555785,  1.76248691,  1.43054938],
       [ 0.54872174, -1.36028717,  0.63026834,  0.3877799 ],
       [-0.87242809,  1.82119407, -1.2378923 , -1.30672051],
       [ 0.19343428, -0.87082852,  0.74349019,  0.51812609],
       [ 0.43029259, -0.62609919,  0.57365741,  0.77881846],
       [-0.99085724,  0.84227676, -1.29450323, -1.30672051],
       [-1.10928639, -0.13664054, -1.35111416, -1.30672051],
       [-0.28028233, -0.62609919,  0.63026834,  1.03951083],
       [ 1.3777258 ,  0.35281811,  0.51704648,  0.25743372],
       [-1.22771554,  0.84227676, -1.06805952, -1.30672051],
       [-1.10928639,  0.10808879, -1.29450323, -1.43706669],
       [-1.46457384,  0.10808879, -1.29450323, -1.30672051],
       [-1.70143215, -0.38136987, -1.35111416, -1.30672051],
       [ 0.66715089, -0.62609919,  1.02654484,  1.3002032 ],
       [-0.87242809,  0.84227676, -1.29450323, -1.30672051],
       [-0.39871148,  2.80011138, -1.35111416, -1.30672051],
       [-0.87242809, -1.36028717, -0.4453393 , -0.13360484],
       [ 0.54872174,  0.59754744,  0.51704648,  0.51812609],
       [ 0.54872174, -0.38136987,  1.02654484,  0.77881846],
       [ 0.54872174, -0.62609919,  0.74349019,  0.3877799 ],
       [-0.99085724,  0.59754744, -1.35111416, -1.30672051],
       [-0.28028233, -0.38136987, -0.10567373,  0.12708753],
       [ 0.90400919, -0.13664054,  0.34721369,  0.25743372],
       [-0.87242809,  1.08700609, -1.35111416, -1.30672051],
       [ 1.25929665,  0.35281811,  1.08315577,  1.43054938],
       [ 2.08830071, -0.13664054,  1.59265412,  1.16985701],
       [ 0.66715089, -0.62609919,  1.02654484,  1.16985701],
       [ 1.1408675 ,  0.35281811,  1.19637762,  1.43054938],
       [-1.34614469,  0.35281811, -1.2378923 , -1.30672051],
       [ 0.54872174, -1.84974582,  0.34721369,  0.12708753],
       [-0.04342402, -0.87082852,  0.74349019,  0.90916464],
       [ 0.66715089,  0.35281811,  0.40382462,  0.3877799 ],
       [-0.39871148, -1.11555785,  0.34721369, -0.00325865],
       [ 2.20672986, -0.13664054,  1.30959948,  1.43054938],
       [-0.04342402, -1.11555785,  0.12076998, -0.00325865],
       [-0.51714063,  0.84227676, -1.18128137, -1.30672051],
       [ 0.07500513, -0.13664054,  0.23399184,  0.3877799 ],
       [-0.16185317, -0.38136987,  0.23399184,  0.12708753],
       [ 0.07500513,  0.35281811,  0.57365741,  0.77881846]])
```

In [48]:
```
X_test
```

Out[48]:
```
array([[-2.80282326e-01, -8.70828519e-01,  2.33991838e-01,
         1.27087531e-01],
       [-1.70143215e+00, -1.36640541e-01, -1.40772509e+00,
        -1.30672051e+00],
       [-4.34240223e-02, -6.26099193e-01,  7.43490194e-01,
         1.56089557e+00],
       [ 1.02243834e+00,  1.08088786e-01,  3.47213695e-01,
         2.57433716e-01],
       [ 3.11863433e-01, -1.36640541e-01,  6.30268337e-01,
         7.78818457e-01],
       [-1.61853174e-01,  3.28957003e+00, -1.29450323e+00,
        -1.04602814e+00],
       [-1.61853174e-01,  1.82119407e+00, -1.18128137e+00,
        -1.17637432e+00],
       [ 1.25929665e+00,  1.08088786e-01,  6.30268337e-01,
         3.87779901e-01],
       [ 1.14086750e+00, -1.36640541e-01,  9.69933908e-01,
         1.16985701e+00],
       [-1.10928639e+00, -1.60501650e+00, -2.75506519e-01,
        -2.63951025e-01],
       [ 3.11863433e-01, -6.26099193e-01,  5.17046480e-01,
        -3.25865463e-03],
       [-5.17140630e-01,  2.06592340e+00, -1.18128137e+00,
        -1.04602814e+00],
       [-3.98711478e-01, -1.60501650e+00, -4.90628047e-02,
        -2.63951025e-01],
       [ 5.48721737e-01, -1.36028717e+00,  6.86879266e-01,
         9.09164643e-01],
       [-8.72428085e-01,  1.08700609e+00, -1.35111416e+00,
        -1.17637432e+00],
       [ 2.20672986e+00, -6.26099193e-01,  1.64926505e+00,
         1.03951083e+00],
       [-4.34240223e-02,  2.31065272e+00, -1.46433602e+00,
        -1.30672051e+00],
       [ 1.02243834e+00, -1.36640541e-01,  6.86879266e-01,
         6.48472272e-01],
       [ 7.85580041e-01, -1.36640541e-01,  1.13976669e+00,
         1.30020320e+00],
       [-5.17140630e-01,  8.42276765e-01, -1.29450323e+00,
        -1.04602814e+00],
       [ 1.02243834e+00,  1.08088786e-01,  1.02654484e+00,
         1.56089557e+00],
       [-1.61853174e-01, -6.26099193e-01,  1.77380909e-01,
         1.27087531e-01],
       [ 1.93434281e-01, -2.09447515e+00,  1.20769981e-01,
        -2.63951025e-01],
       [-9.90857237e-01,  3.52818112e-01, -1.46433602e+00,
        -1.30672051e+00],
       [ 4.30292585e-01, -2.09447515e+00,  4.03824623e-01,
         3.87779901e-01],
       [ 7.50051295e-02, -1.36640541e-01,  7.43490194e-01,
         7.78818457e-01],
       [-1.46457384e+00,  1.33173542e+00, -1.57755787e+00,
        -1.30672051e+00],
       [ 7.85580041e-01,  3.52818112e-01,  7.43490194e-01,
         1.03951083e+00],
       [-2.80282326e-01, -1.36028717e+00,  6.41590523e-02,
        -1.33604840e-01],
       [-1.22771554e+00, -1.36640541e-01, -1.35111416e+00,
        -1.17637432e+00]])
```

Training and Predictions

In [49]:
```
from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
```

Out[49]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

The first step is to import the KNeighborsClassifier class from the sklearn.neighbors library.

In the second line, this class is initialized with one parameter, i.e. n_neigbours.

This is basically the value for the K.

There is no ideal value for K and it is selected after testing and evaluation, however to start out, 5 seems to be the most commonly used value for KNN algorithm.

The final step is to make predictions on our test data.

In [50]:
```
y_pred = classifier.predict(X_test)
```

In [51]:
```
y_pred
```

Out[51]:
```
array(['Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa'], dtype=object)
```

Evaluating the Algorithm

In [52]:
```
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, y_pred))
print()
print("*****************************************************************")
print()
print(classification_report(y_test, y_pred))
```

```
[[10  0  0]
 [ 0 10  1]
 [ 0  1  8]]

*****************************************************************

                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       0.91      0.91      0.91        11
 Iris-virginica       0.89      0.89      0.89         9

       accuracy                           0.93        30
      macro avg       0.93      0.93      0.93        30
   weighted avg       0.93      0.93      0.93        30
```

Comparing Error Rate with the K Value

In [53]:
```
error = []

# Calculating error for K values between 1 and 40
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error.append(np.mean(pred_i != y_test))
```

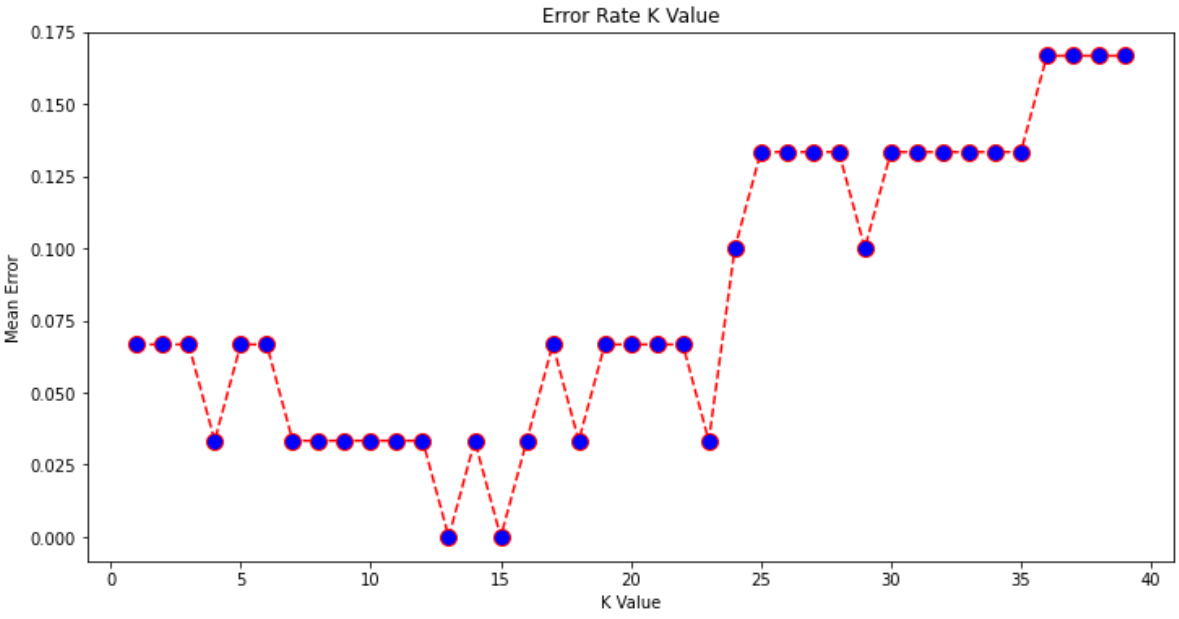
we will plot the mean error for the predicted values of test set for all the K values between 1 and 40.

The above script executes a loop from 1 to 40. In each iteration the mean error for predicted values of test set is calculated and the result is appended to the error list.

The next step is to plot the error values against K values. Execute the following script to create the plot:

In [54]:
```
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

Out[54]:
```
Text(0, 0.5, 'Mean Error')
```



In [54]:

Practical 6

A. Implement the different Distance methods (Euclidean) with Prediction, Test Score and Confusion Matrix.

In [1]:
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:
```
# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, -1].values
```

In [3]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random
```

In [4]:
```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [5]:
```
# Training the K-NN model on the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
```

Out[5]:
```
KNeighborsClassifier()
```

We are using 3 parameters in the model creation. n_neighbors is setting as 5, which means 5 neighborhood points are required for classifying a given point. The distance metric we are using is Minkowski, the equation for it is given below

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

As per the equation, we have to select the p-value also.

p = 1 , Manhattan Distance

p = 2 , Euclidean Distance

p = infinity , Cheybchev Distance

In our problem, we are choosing the p as 2 (also u can choose the metric as “euclidean”) Our Model is created, now we have to predict the output for the test set

In [6]:
```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

In [10]:
```
y_pred
```

Out[10]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1])
```

In [7]:
```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac = accuracy_score(y_test, y_pred)
```

In [8]:
```
cm
```

Out[8]:
```
array([[55,  3],
       [ 1, 21]])
```

In [9]:
```
ac
```

Out[9]:
```
0.95
```

In [ ]:

B. Implement the classification model using clustering for the following techniques with K means clustering with Prediction, Test Score andConfusion Matrix.

In [2]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
# Importing the dataset
dataset = pd.read_csv('/content/Mall_Customers.csv')
```

In [4]:
```
dataset.head()
```

Out[4]:

	CustomerID	Genre	Age	Annual_Income_(k$)	Spending_Score
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [5]:
```
dataset.tail()
```

Out[5]:

	CustomerID	Genre	Age	Annual_Income_(k$)	Spending_Score
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

In [6]:
```
dataset.shape
```

Out[6]:
```
(200, 5)
```

In [7]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Genre                   200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual_Income_(k$)      200 non-null    int64
 4   Spending_Score          200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

In [8]:
```
dataset.describe()
```

Out[8]:

	CustomerID	Age	Annual_Income_(k$)	Spending_Score
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

In [9]:
```
dataset.isnull().sum()
```

Out[9]:
```
CustomerID              0
Genre                   0
Age                     0
Annual_Income_(k$)      0
Spending_Score          0
dtype: int64
```

In [10]:
```
dataset.columns
```

Out[10]:
```
Index(['CustomerID', 'Genre', 'Age', 'Annual_Income_(k$)', 'Spending_Score'], dtype='object')
```

No Nans found! Great

In [11]:
```
data=dataset[['Annual_Income_(k$)','Spending_Score']]
```

In [12]:
```
data
```

Out[12]:

	Annual_Income_(k$)	Spending_Score
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40
...	...	...
195	120	79
196	126	28
197	126	74
198	137	18
199	137	83

200 rows × 2 columns

In [13]:
```
data.shape
```

Out[13]:
```
(200, 2)
```

In [14]:
```
data.values
```

Out[14]:
```
array([[ 15,  39],
       [ 15,  81],
       [ 16,   6],
       [ 16,  77],
       [ 17,  40],
       [ 17,  76],
       [ 18,   6],
       [ 18,  94],
       [ 19,   3],
       [ 19,  72],
       [ 19,  14],
       [ 19,  99],
       [ 20,  15],
       [ 20,  77],
       [ 20,  13],
       [ 20,  79],
       [ 21,  35],
       [ 21,  66],
       [ 23,  29],
       [ 23,  98],
       [ 24,  35],
       [ 24,  73],
       [ 25,   5],
       [ 25,  73],
       [ 28,  14],
       [ 28,  82],
       [ 28,  32],
       [ 28,  61],
       [ 29,  31],
       [ 29,  87],
       [ 30,   4],
       [ 30,  73],
       [ 33,   4],
       [ 33,  92],
       [ 33,  14],
       [ 33,  81],
       [ 34,  17],
       [ 34,  73],
       [ 37,  26],
       [ 37,  75],
       [ 38,  35],
       [ 38,  92],
       [ 39,  36],
       [ 39,  61],
       [ 39,  28],
       [ 39,  65],
       [ 40,  55],
       [ 40,  47],
       [ 40,  42],
       [ 40,  42],
       [ 42,  52],
       [ 42,  60],
       [ 43,  54],
       [ 43,  60],
       [ 43,  45],
       [ 43,  41],
       [ 44,  50],
       [ 44,  46],
       [ 46,  51],
       [ 46,  46],
       [ 46,  56],
       [ 46,  55],
       [ 47,  52],
       [ 47,  59],
```

```
[ 48,  51],
[ 48,  59],
[ 48,  50],
[ 48,  48],
[ 48,  59],
[ 48,  47],
[ 49,  55],
[ 49,  42],
[ 50,  49],
[ 50,  56],
[ 54,  47],
[ 54,  54],
[ 54,  53],
[ 54,  48],
[ 54,  52],
[ 54,  42],
[ 54,  51],
[ 54,  55],
[ 54,  41],
[ 54,  44],
[ 54,  57],
[ 54,  46],
[ 57,  58],
[ 57,  55],
[ 58,  60],
[ 58,  46],
[ 59,  55],
[ 59,  41],
[ 60,  49],
[ 60,  40],
[ 60,  42],
[ 60,  52],
[ 60,  47],
[ 60,  50],
[ 61,  42],
[ 61,  49],
[ 62,  41],
[ 62,  48],
[ 62,  59],
[ 62,  55],
[ 62,  56],
[ 62,  42],
[ 63,  50],
[ 63,  46],
[ 63,  43],
[ 63,  48],
[ 63,  52],
[ 63,  54],
[ 64,  42],
[ 64,  46],
[ 65,  48],
[ 65,  50],
[ 65,  43],
[ 65,  59],
[ 67,  43],
[ 67,  57],
[ 67,  56],
[ 67,  40],
[ 69,  58],
[ 69,  91],
[ 70,  29],
[ 70,  77],
[ 71,  35],
[ 71,  95],
```

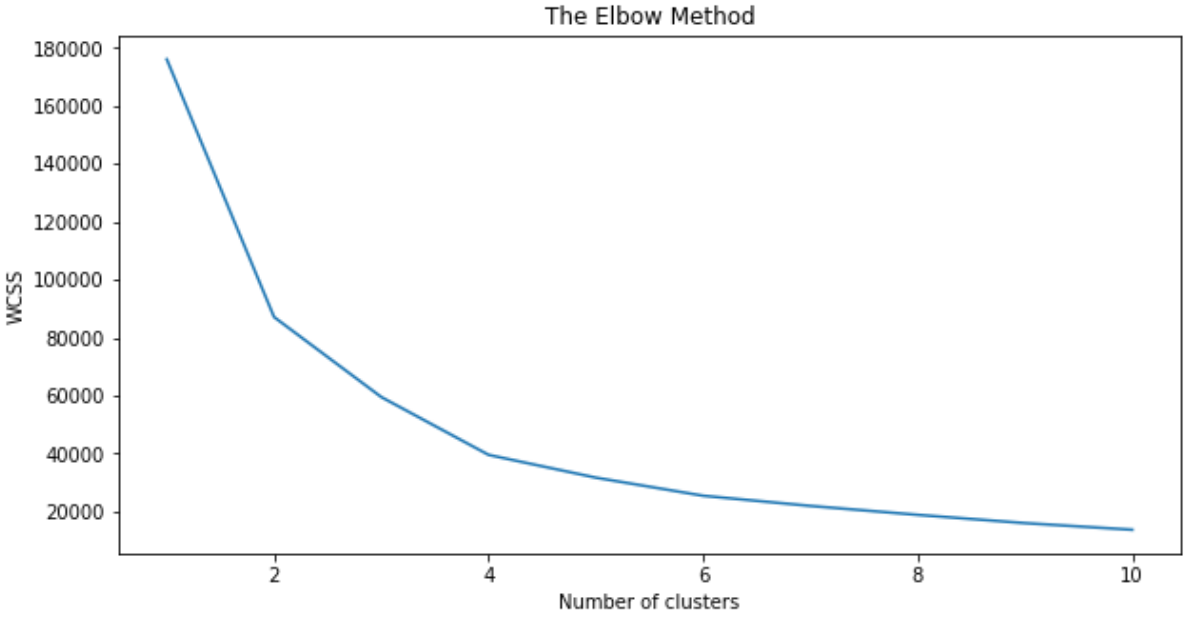
```
[ 71,  11],
[ 71,  75],
[ 71,   9],
[ 71,  75],
[ 72,  34],
[ 72,  71],
[ 73,   5],
[ 73,  88],
[ 73,   7],
[ 73,  73],
[ 74,  10],
[ 74,  72],
[ 75,   5],
[ 75,  93],
[ 76,  40],
[ 76,  87],
[ 77,  12],
[ 77,  97],
[ 77,  36],
[ 77,  74],
[ 78,  22],
[ 78,  90],
[ 78,  17],
[ 78,  88],
[ 78,  20],
[ 78,  76],
[ 78,  16],
[ 78,  89],
[ 78,   1],
[ 78,  78],
[ 78,   1],
[ 78,  73],
[ 79,  35],
[ 79,  83],
[ 81,   5],
[ 81,  93],
[ 85,  26],
[ 85,  75],
[ 86,  20],
[ 86,  95],
[ 87,  27],
[ 87,  63],
[ 87,  13],
[ 87,  75],
[ 87,  10],
[ 87,  92],
[ 88,  13],
[ 88,  86],
[ 88,  15],
[ 88,  69],
[ 93,  14],
[ 93,  90],
[ 97,  32],
[ 97,  86],
[ 98,  15],
[ 98,  88],
[ 99,  39],
[ 99,  97],
[101,  24],
[101,  68],
[103,  17],
[103,  85],
[103,  23],
[103,  69],
```

```
       [113,   8],
       [113,  91],
       [120,  16],
       [120,  79],
       [126,  28],
       [126,  74],
       [137,  18],
       [137,  83]])
```

In [17]:
```
#we always assume the max number of cluster would be 10
#you can judge the number of clusters by doing averaging
###Static code to get max no of clusters
X = dataset.iloc[:, [2, 3]].values
```

In [18]:
```
# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wc_ss = []
for i in range(1, 11):
    kmeans_clu = KMeans(n_clusters = i, random_state = 56)
    kmeans_clu.fit(X)
    # inertia method returns wcss for that model
    wc_ss.append(kmeans_clu.inertia_)
```

In [19]:
```
plt.figure(figsize=(10,5))
plt.plot(range(1,11), wc_ss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

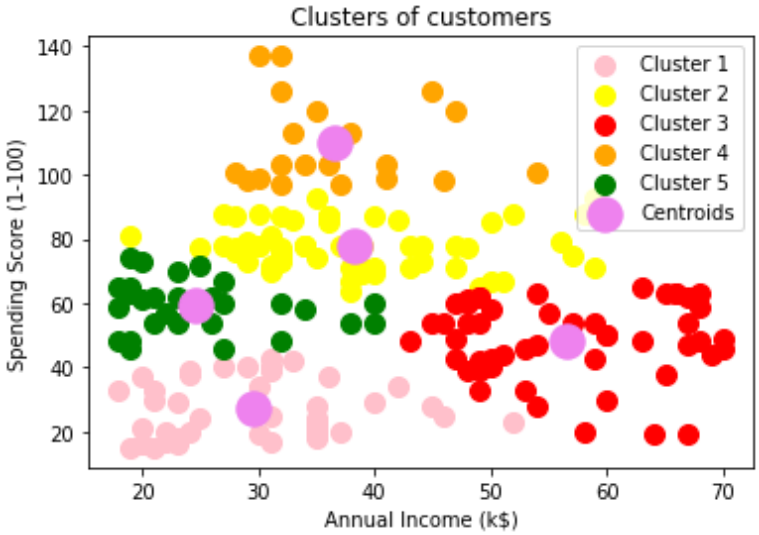


this curve is telling you that last elbow comes at k=5 no matter what range we select ex-(1,21) also i will see the same behaviour but if we chose higher range it is little difficult to visualize the ELBOW that is why we usually prefer range (1,11) Finally we got that k=5

In [20]:
```
# Fitting K-Means to the dataset
kmeans = KMeans(n_clusters = 5, random_state = 56)
y_kmeans = kmeans.fit_predict(X)
```

7. Visualisation

In [21]:
```
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'pink', label =
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'yellow', label
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'red', label =
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'orange', label
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'green', label
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300,
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



Practical 7

Implement the classification model using clusteringfor the following techniques with hierarchical clustering with Prediction, Test Score and ConfusionMatrix

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [3]:
```
dataset = pd.read_csv('Mall_Customers.csv')
```

In [7]:
```
dataset.head()
```

Out[7]:

	CustomerID	Genre	Age	Annual_Income_(k$)	Spending_Score
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Our goal is to cluster the customers based on their Spending score. That's why CustomerID and Genre are useless. So we remove both columns.

In [8]:
```
X = dataset.iloc[:, [3, 4]].values
```

Now, we have only Annual Income and Spending Score Column.

In [9]:
```
X
```

Out[9]:
```
array([[ 15,  39],
       [ 15,  81],
       [ 16,   6],
       [ 16,  77],
       [ 17,  40],
       [ 17,  76],
       [ 18,   6],
       [ 18,  94],
       [ 19,   3],
       [ 19,  72],
       [ 19,  14],
       [ 19,  99],
       [ 20,  15],
       [ 20,  77],
       [ 20,  13],
       [ 20,  79],
       [ 21,  35],
       [ 21,  66],
       [ 23,  29],
       [ 23,  98],
       [ 24,  35],
       [ 24,  73],
       [ 25,   5],
       [ 25,  73],
       [ 28,  14],
       [ 28,  82],
       [ 28,  32],
       [ 28,  61],
       [ 29,  31],
       [ 29,  87],
       [ 30,   4],
       [ 30,  73],
       [ 33,   4],
       [ 33,  92],
       [ 33,  14],
       [ 33,  81],
       [ 34,  17],
       [ 34,  73],
       [ 37,  26],
       [ 37,  75],
       [ 38,  35],
       [ 38,  92],
       [ 39,  36],
       [ 39,  61],
       [ 39,  28],
       [ 39,  65],
       [ 40,  55],
       [ 40,  47],
       [ 40,  42],
       [ 40,  42],
       [ 42,  52],
       [ 42,  60],
       [ 43,  54],
       [ 43,  60],
       [ 43,  45],
       [ 43,  41],
       [ 44,  50],
       [ 44,  46],
       [ 46,  51],
       [ 46,  46],
       [ 46,  56],
       [ 46,  55],
       [ 47,  52],
       [ 47,  59],
```

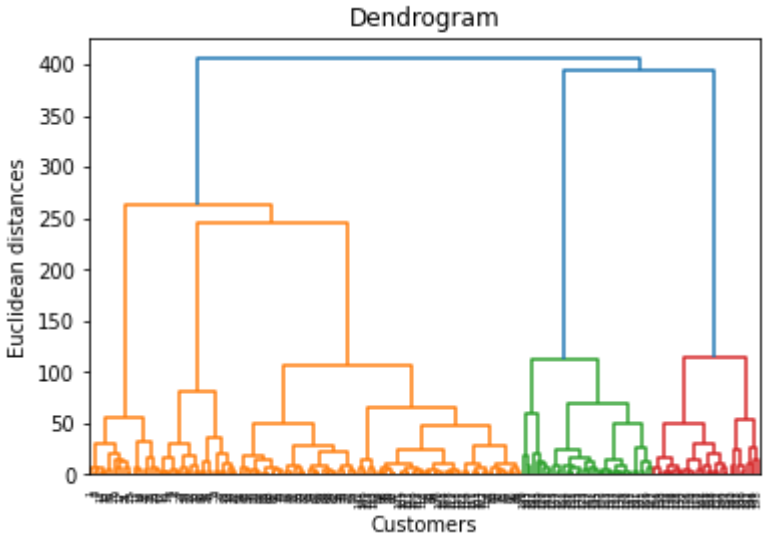
```
[ 48,  51],
[ 48,  59],
[ 48,  50],
[ 48,  48],
[ 48,  59],
[ 48,  47],
[ 49,  55],
[ 49,  42],
[ 50,  49],
[ 50,  56],
[ 54,  47],
[ 54,  54],
[ 54,  53],
[ 54,  48],
[ 54,  52],
[ 54,  42],
[ 54,  51],
[ 54,  55],
[ 54,  41],
[ 54,  44],
[ 54,  57],
[ 54,  46],
[ 57,  58],
[ 57,  55],
[ 58,  60],
[ 58,  46],
[ 59,  55],
[ 59,  41],
[ 60,  49],
[ 60,  40],
[ 60,  42],
[ 60,  52],
[ 60,  47],
[ 60,  50],
[ 61,  42],
[ 61,  49],
[ 62,  41],
[ 62,  48],
[ 62,  59],
[ 62,  55],
[ 62,  56],
[ 62,  42],
[ 63,  50],
[ 63,  46],
[ 63,  43],
[ 63,  48],
[ 63,  52],
[ 63,  54],
[ 64,  42],
[ 64,  46],
[ 65,  48],
[ 65,  50],
[ 65,  43],
[ 65,  59],
[ 67,  43],
[ 67,  57],
[ 67,  56],
[ 67,  40],
[ 69,  58],
[ 69,  91],
[ 70,  29],
[ 70,  77],
[ 71,  35],
[ 71,  95],
```

```
[ 71,  11],
[ 71,  75],
[ 71,   9],
[ 71,  75],
[ 72,  34],
[ 72,  71],
[ 73,   5],
[ 73,  88],
[ 73,   7],
[ 73,  73],
[ 74,  10],
[ 74,  72],
[ 75,   5],
[ 75,  93],
[ 76,  40],
[ 76,  87],
[ 77,  12],
[ 77,  97],
[ 77,  36],
[ 77,  74],
[ 78,  22],
[ 78,  90],
[ 78,  17],
[ 78,  88],
[ 78,  20],
[ 78,  76],
[ 78,  16],
[ 78,  89],
[ 78,   1],
[ 78,  78],
[ 78,   1],
[ 78,  73],
[ 79,  35],
[ 79,  83],
[ 81,   5],
[ 81,  93],
[ 85,  26],
[ 85,  75],
[ 86,  20],
[ 86,  95],
[ 87,  27],
[ 87,  63],
[ 87,  13],
[ 87,  75],
[ 87,  10],
[ 87,  92],
[ 88,  13],
[ 88,  86],
[ 88,  15],
[ 88,  69],
[ 93,  14],
[ 93,  90],
[ 97,  32],
[ 97,  86],
[ 98,  15],
[ 98,  88],
[ 99,  39],
[ 99,  97],
[101,  24],
[101,  68],
[103,  17],
[103,  85],
[103,  23],
[103,  69],
```

```
       [113,   8],
       [113,  91],
       [120,  16],
       [120,  79],
       [126,  28],
       [126,  74],
       [137,  18],
       [137,  83]])
```

We have loaded dataset. Now its time to find the optimal number of clusters. And for that we need to create a Dendrogram.

In [10]:
```
# Create Dendrogram to find the Optimal Number of Clusters

import scipy.cluster.hierarchy as sch
dendro = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean  distances')
plt.show()
```



Here in the code “sch” is the short code for scipy.cluster.hierarchy.”

“dendro” is the variable name. It may be anything. And “Dendrogram” is the function name.

So, after implementing this code, we will get our Dendrogram.

As I discussed that cut the horizontal line with longest line that traverses maximum distance up and down without intersecting the merging points.

In that dendrogram, the optimal number of clusters are 5.

Now let’s fit our Agglomerative model with 5 clusters.

In [11]:
```
# Fitting Agglomerative Hierarchical Clustering to the dataset

from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'wa
y_hc = hc.fit_predict(X)
```

Now our model has been trained. If you want to see different clusters, you can do it by simply writing print.

In [12]:
```
print(y_hc)
```

```
[4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4
 3 4 3 4 3 4 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 0 2 0 2 1 2 0 2 0 2 0 2 1 2 0 2 1 2
 0 2 0 2 0 2 0 2 0 2 0 2 1 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0
 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2]
```

Now, its time to visualize the clusters.

In [13]:
```
# Visualise the clusters


plt.scatter(X[y_hc  == 0, 0], X[y_hc == 0, 1], s = 100, c = 'red', label = 'Cluster
plt.scatter(X[y_hc  == 1, 0], X[y_hc == 1, 1], s = 100, c = 'blue', label = 'Cluste
plt.scatter(X[y_hc  == 2, 0], X[y_hc == 2, 1], s = 100, c = 'green', label = 'Clust
plt.scatter(X[y_hc  == 3, 0], X[y_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluste
plt.scatter(X[y_hc  == 4, 0], X[y_hc == 4, 1], s = 100, c = 'magenta', label = 'Clu
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

In [ ]:

Practical 8

A. Exploratory Data Analysis (EDA)

Objective to achieve, understand and implement following topics

– Handle Missing value

– Removing duplicates

– Outlier Treatment

– Normalizing and Scaling(Numerical Variables)

– Encoding Categorical variables(Dummy Variables)

– Bivariate Analysis

In [61]:
```
# Importing all the necessary libraries
import numpy as np   # numerial python
import pandas as pd  # dataframe working
import matplotlib.pyplot as plt #  plotting
import seaborn as sn # advance plotting
%matplotlib inline
```

In [22]:
```
df_car = pd.read_excel("/content/EDA Cars.xlsx")
df_car.head(30)
```

Out[22]:

	INDEX	INCOME	MARITAL STATUS	SEX	EDUCATION	JOB	TRAVEL TIME	USE	MILES CLOCKED
0	1	125301	No	F	Bachelors	Blue Collar	45.703013	Commercial	17430.0
1	2	50815.4	No	M	High School	NaN	20.591628	Private	18930.0
2	3	62977.8	NaN	F	Bachelors	Clerical	33.639949	Private	NaN
3	4	77100	No	F	NaN	Lawyer	15.415676	NaN	18300.0
4	5	130795	No	M	High School	NaN	NaN	Commercial	28340.0
5	6	NaN	NaN	F	High School	Home Maker	48.360191	NaN	6000.0
6	7	87460.1	No	M	High School	Manager	45.000488	NaN	15420.0
7	8	NaN	Yes	F	High School	Blue Collar	15.665947	NaN	11290.0
8	9	@@	NaN	F	NaN	Clerical	26.392961	NaN	10030.0
9	10	@@	Yes	M	High School	Blue Collar	27.490749	NaN	NaN
10	11	16988.7	No	M	High School	Blue Collar	20.450016	NaN	NaN
11	12	@@	No	F	High School	Blue Collar	61.603208	Commercial	9360.0
12	13	16352	Yes	F	NaN	Home Maker	NaN	Private	10520.0
13	14	63952.2	Yes	F	Masters	Lawyer	49.576895	Private	32340.0
14	15	24059.7	No	M	Bachelors	Clerical	24.430198	Private	14220.0
15	16	20821.8	Yes	F	High School	Home Maker	45.907713	Private	6880.0
16	17	@@	No	F	High School	Blue Collar	42.500815	Private	NaN
17	18	137103	NaN	F	NaN	Manager	28.800357	Private	19510.0
18	19	102393	No	M	Bachelors	Professional	24.867970	NaN	1500.0
19	20	40656.4	Yes	F	High School	NaN	57.091375	Private	9170.0
20	21	@@	Yes	F	Bachelors	Blue Collar	NaN	Commercial	5200.0
21	22	31773.1	Yes	M	High School	Clerical	14.459323	Commercial	9640.0
22	23	38035.6	Yes	F	High School	Blue Collar	32.180285	Commercial	13140.0
23	24	0	Yes	F	High School	Home Maker	24.520001	Private	11740.0
24	25	0	NaN	M	High School	Student	13.662860	Commercial	15090.0
25	26	NaN	Yes	F	Bachelors	Student	48.143336	Commercial	12850.0
26	27	44705.2	Yes	F	Bachelors	Professional	50.088976	Private	12840.0
27	28	31738.4	Yes	M	High School	Blue Collar	23.286841	Commercial	NaN

	INDEX	INCOME	MARITAL STATUS	SEX	EDUCATION	JOB	TRAVEL TIME	USE	MILES CLOCKED
28	29	64013.8	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0
29	30	53244.4	No	M	Bachelors	Professional	27.551402	NaN	20920.0

In [23]:
```
df_car.shape
```

Out[23]:
```
(303, 13)
```

In [24]:
```
df_car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   INDEX           303 non-null    int64  
 1   INCOME          265 non-null    object 
 2   MARITAL STATUS  275 non-null    object 
 3   SEX             297 non-null    object 
 4   EDUCATION       259 non-null    object 
 5   JOB             257 non-null    object 
 6   TRAVEL TIME     262 non-null    float64
 7   USE             250 non-null    object 
 8   MILES CLOCKED   278 non-null    float64
 9   CAR TYPE        293 non-null    object 
 10  CAR AGE         283 non-null    float64
 11  CITY            297 non-null    object 
 12  POSTAL CODE     300 non-null    float64
dtypes: float64(4), int64(1), object(8)
memory usage: 30.9+ KB
```

In [16]:
```
df_car.describe()
```

Out[16]:

	INDEX	TRAVEL TIME	MILES CLOCKED	CAR AGE	POSTAL CODE
count	303.000000	262.000000	278.000000	283.000000	300.000000
mean	139.640264	34.282098	13591.978417	6.265018	50712.196667
std	85.178422	14.910178	7167.328655	5.111218	24141.029290
min	1.000000	5.000000	1500.000000	1.000000	11435.000000
25%	62.500000	24.449874	7900.000000	1.000000	42420.000000
50%	138.000000	33.564757	12065.000000	6.000000	47150.000000
75%	213.500000	43.907339	18240.000000	10.000000	61701.000000
max	289.000000	83.617643	38000.000000	20.000000	90049.000000

In [25]:
```
df_car.describe(include=['object',  'int','float'])
```

Out[25]:

	INDEX	INCOME	MARITAL STATUS	SEX	EDUCATION	JOB	TRAVEL TIME	USE	MIL CLOCKE
count	303.000000	265.0	275	297	259	257	262.000000	250	278.0000
unique	NaN	222.0	2	2	4	8	NaN	2	Na
top	NaN	0.0	No	F	High School	Blue Collar	NaN	Private	Na
freq	NaN	25.0	151	165	161	96	NaN	133	Na
mean	139.640264	NaN	NaN	NaN	NaN	NaN	34.282098	NaN	13591.9784
std	85.178422	NaN	NaN	NaN	NaN	NaN	14.910178	NaN	7167.3286
min	1.000000	NaN	NaN	NaN	NaN	NaN	5.000000	NaN	1500.0000
25%	62.500000	NaN	NaN	NaN	NaN	NaN	24.449874	NaN	7900.0000
50%	138.000000	NaN	NaN	NaN	NaN	NaN	33.564757	NaN	12065.0000
75%	213.500000	NaN	NaN	NaN	NaN	NaN	43.907339	NaN	18240.0000
max	289.000000	NaN	NaN	NaN	NaN	NaN	83.617643	NaN	38000.0000

In [26]:
```
df_car.INCOME
```

Out[26]:
```
0       125301
1      50815.4
2      62977.8
3        77100
4       130795
        ...
298    15251.5
299    18408.4
300        NaN
301        NaN
302          0
Name: INCOME, Length: 303, dtype: object
```

In [27]:
```
df_car[df_car['INCOME'] == '@@']
```

Out[27]:

	INDEX	INCOME	MARITAL STATUS	SEX	EDUCATION	JOB	TRAVEL TIME	USE	MILES CLOCKED	CA TY
8	9	@@	NaN	F	NaN	Clerical	26.392961	NaN	10030.0	SU
9	10	@@	Yes	M	High School	Blue Collar	27.490749	NaN	NaN	Pan Tru
11	12	@@	No	F	High School	Blue Collar	61.603208	Commercial	9360.0	Spo C
16	17	@@	No	F	High School	Blue Collar	42.500815	Private	NaN	SU
20	21	@@	Yes	F	Bachelors	Blue Collar	NaN	Commercial	5200.0	SU

In [19]:
```
```

In [29]:
```
df_car['INCOME'] = df_car['INCOME'].replace(to_replace = '@@', value= np.nan)
```

```
df_car['INCOME'] = df_car['INCOME'].astype(float)
```

In [30]:
```
df_car[df_car['INCOME'] == '@@']
```

Out[30]:

INDEX	INCOME	MARITAL STATUS	SEX	EDUCATION	JOB	TRAVEL TIME	USE	MILES CLOCKED	CAR TYPE	CAR AGE	CITY

In [31]:
```
#df_car['TRAVEL TIME'] = df_car['TRAVEL TIME'].replace(to_replace = '*****', value
```

In [32]:
```
df_car['TRAVEL TIME'] = df_car['TRAVEL TIME'].astype(float)
```

In [35]:
```
#df_car['MILES CLOCKED'] = df_car['MILES CLOCKED'].replace(to_replace = 'Na', value
#df_car['MILES CLOCKED'] = df_car['MILES CLOCKED'].replace(to_replace = '*****', va
#df_car['MILES CLOCKED'] = df_car['MILES CLOCKED'].astype(float)
```

In [34]:
```
#df_car.replace(to_replace='*****', value= np.nan, inplace = True)
```

In [33]:
```
df_car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   INDEX           303 non-null    int64  
 1   INCOME          260 non-null    float64
 2   MARITAL STATUS  275 non-null    object 
 3   SEX             297 non-null    object 
 4   EDUCATION       259 non-null    object 
 5   JOB             257 non-null    object 
 6   TRAVEL TIME     262 non-null    float64
 7   USE             250 non-null    object 
 8   MILES CLOCKED   278 non-null    float64
 9   CAR TYPE        293 non-null    object 
 10  CAR AGE         283 non-null    float64
 11  CITY            297 non-null    object 
 12  POSTAL CODE     300 non-null    float64
dtypes: float64(5), int64(1), object(7)
memory usage: 30.9+ KB
```

In [36]:
```
# Check for missing values in any column or Handling missing values in dataset

df_car.isnull().sum()
```

Out[36]:
```
INDEX             0
INCOME           43
MARITAL STATUS   28
SEX               6
EDUCATION        44
JOB              46
TRAVEL TIME      41
USE              53
MILES CLOCKED    25
CAR TYPE         10
CAR AGE          20
CITY              6
POSTAL CODE       3
dtype: int64
```

We can see that we have various missing values in the respective columns. There are various ways of treating your missing values in the data set. And which technique to use when is actually dependent on the type of data you are dealing with.

- Drop the missing values: In this case, we drop the missing values from those variables. In case there are very few missing values you can drop those values.
- Impute with mean value: For the numerical column, you can replace the missing values with mean values. Before replacing with mean value, it is advisable to check that the variable shouldn't have extreme values .i.e. outliers.
- Impute with median value: For the numerical column, you can also replace the missing values with median values. In case you have extreme values such as outliers it is advisable to use the median approach.
- Impute with mode value: For the categorical column, you can replace the missing values with mode values i.e the frequent ones.

In [37]:
```
# In this exercise, we will replace the numerical columns with median values and fc


#Replacing the NULL Values in numerical columns using MEDIAN
median_income = df_car['INCOME'].median()
median_travel_time = df_car['TRAVEL TIME'].median()
median_miles_clocked = df_car['MILES CLOCKED'].median()
median_car_age = df_car['CAR AGE'].median()
median_postal_code = df_car['POSTAL CODE'].median()


df_car['INCOME'].replace(np.nan, median_income, inplace = True)
df_car['TRAVEL TIME'].replace(np.nan, median_travel_time, inplace = True)
df_car['MILES CLOCKED'].replace(np.nan, median_miles_clocked, inplace = True)
df_car['CAR AGE'].replace(np.nan, median_car_age, inplace = True)
df_car['POSTAL CODE'].replace(np.nan, median_postal_code, inplace = True)
```

In [38]:
```
# Replacing the NULL values in  categorical columns using MODE


mode_sex = df_car['SEX'].mode().values[0]
mode_martial_status = df_car['MARITAL STATUS'].mode().values[0]
mode_education = df_car['EDUCATION'].mode().values[0]
mode_job = df_car['JOB'].mode().values[0]
mode_use = df_car['USE'].mode().values[0]
mode_city = df_car['CITY'].mode().values[0]
mode_car_type = df_car['CAR TYPE'].mode().values[0]


df_car['SEX']= df_car['SEX'].replace(np.nan, mode_sex)
df_car['MARITAL STATUS']= df_car['MARITAL STATUS'].replace(np.nan, mode_martial_sta
df_car['EDUCATION']= df_car['EDUCATION'].replace(np.nan, mode_education)
df_car['JOB']= df_car['JOB'].replace(np.nan, mode_job)
df_car['USE']= df_car['USE'].replace(np.nan, mode_use)
df_car['CITY']= df_car['CITY'].replace(np.nan, mode_city)
df_car['CAR TYPE']= df_car['CAR TYPE'].replace(np.nan, mode_car_type)
```

In [39]:
```
df_car.isnull().sum()
```

Out[39]:
```
INDEX             0
INCOME            0
MARITAL STATUS    0
SEX               0
EDUCATION         0
JOB               0
TRAVEL TIME       0
USE               0
MILES CLOCKED     0
CAR TYPE          0
CAR AGE           0
CITY              0
POSTAL CODE       0
dtype: int64
```

In [40]:
```
# Handling Duplicate Records
duplicate = df_car.duplicated()
print(duplicate.sum())
df_car[duplicate]
```

```
14
```

Out[40]:

	INDEX	INCOME	MARITAL STATUS	SEX	EDUCATION	JOB	TRAVEL TIME	USE	MILES CLOCKED	T
69	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi
70	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi
71	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi
72	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi
73	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi
74	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi
75	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi
76	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi
77	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi
78	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi
79	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi
80	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi
81	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi
82	29	64013.81632	Yes	M	High School	Blue Collar	32.717234	Commercial	7900.0	Pi

Since we have 14 duplicate records in the data, we will remove this from the data set so that we get only distinct records.

Post removing the duplicate, we will check whether the duplicates have been removed from the data set or not.

In [41]:
```
# code to drop the duplicate
df_car.drop_duplicates(inplace=True)
```

In [42]:
```
dup = df_car.duplicated()
dup.sum()
```

Out[42]:
```
0
```

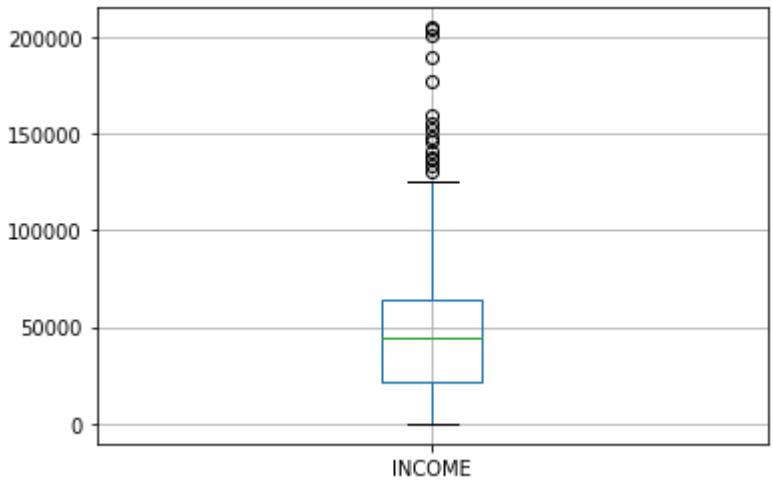
Handling Outlier

Outliers, being the most extreme observations, may include the sample maximum or sample minimum, or both, depending on whether they are extremely high or low.

However, the sample maximum and minimum are not always outliers because they may not be unusually far from other observations.

We Generally identify outliers with the help of boxplot, so here box plot shows some of the data points outside the range of the data.

In [43]:
```
df_car.boxplot(column=['INCOME'])
plt.show()
```



Looking at the box plot, it seems that the variables INCOME, have outlier present in the variables. These outliers value needs to be teated and there are several ways of treating them:

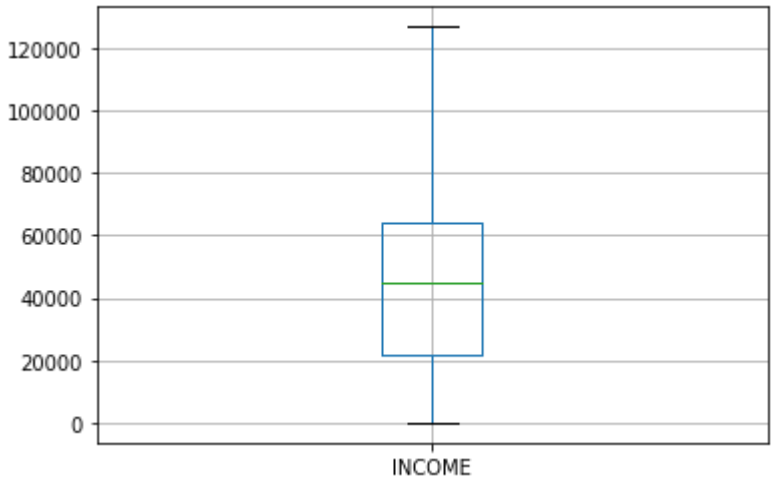
- Drop the outlier value
- Replace the outlier value using the IQR

In [44]:
```
# creating a user defined function called remove_outlier for getting the threshold

def remove_outlier(col):
  sorted(col)
  Q1, Q3 = col.quantile([0.25, 0.75])
  IQR = Q3-Q1
  lower_range = Q1-(1.5 * IQR)
  upper_range = Q3+(1.5 * IQR)
  return lower_range, upper_range
```

In [45]:
```
lower_income, upper_income = remove_outlier(df_car['INCOME'])

df_car['INCOME'] = np.where(df_car['INCOME'] > upper_income, upper_income, df_car[

df_car['INCOME'] = np.where(df_car['INCOME'] < lower_income, lower_income, df_car[
```

After removing outlier, let us check it with boxplot

In [46]:
```
df_car.boxplot(column=['INCOME'])
plt.show()
```



Bivariate Analysis

When we talk about bivariate analysis, it means **analyzing 2 variables**. Since we know there are numerical and categorical variables, there is a way of analyzing these variables as shown below:

Numerical vs. Numerical

1. Scatterplot
2. Line plot
3. Heatmap for correlation
4. Joint plot

Categorical vs. Numerical

1. Bar chart
2. Violin plot
3. Categorical box plot 4.Swarm plot

Two Categorical Variables

1. Bar chart
2. Grouped bar chart
3. Point plot

In [47]:
```
# if we need to find the correlation
df_car.corr()
```

Out[47]:

	INDEX	INCOME	TRAVEL TIME	MILES CLOCKED	CAR AGE	POSTAL CODE
INDEX	1.000000	-0.044968	0.016710	0.042880	-0.027206	-0.244783
INCOME	-0.044968	1.000000	0.062594	0.342164	0.267087	0.034204
TRAVEL TIME	0.016710	0.062594	1.000000	0.026915	0.140511	0.021390
MILES CLOCKED	0.042880	0.342164	0.026915	1.000000	0.127137	-0.111283
CAR AGE	-0.027206	0.267087	0.140511	0.127137	1.000000	-0.099449
POSTAL CODE	-0.244783	0.034204	0.021390	-0.111283	-0.099449	1.000000

Normalizing and Scaling

Often the variables of the data set **are of different scales i.e. one variable is in millions and others in only 100.**

For e.g. in our data set Income is having values in thousands and age in just two digits. Since the data in these variables are of different scales, it is tough to compare these variables.

Feature scaling (also known as data normalization) is the method used to standardize the range of features of data.

Since the range of values of data may vary widely, it becomes a necessary step in data preprocessing while using machine learning algorithms.

In this method, we convert variables with different scales of measurements into a single scale.

StandardScaler normalizes the data using the formula (x-mean)/standard deviation.

So we will be doing this only for the numerical variables.

In [48]:
```
# we use sklearn preprocessing using the function Standard Scaler

from sklearn.preprocessing import StandardScaler
std_scale = StandardScaler()
std_scale
```

Out[48]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [49]:
```
df_car['INCOME'] = std_scale.fit_transform(df_car[['INCOME']])
df_car['TRAVEL TIME'] = std_scale.fit_transform(df_car[['TRAVEL TIME']])
df_car['CAR AGE'] = std_scale.fit_transform(df_car[['CAR AGE']])
df_car['POSTAL CODE'] = std_scale.fit_transform(df_car[['POSTAL CODE']])
df_car['MILES CLOCKED'] = std_scale.fit_transform(df_car[['MILES CLOCKED']])
```

In [50]:
```
df_car.head()
```

Out[50]:

	INDEX	INCOME	MARITAL STATUS	SEX	EDUCATION	JOB	TRAVEL TIME	USE	MILES CLOCKED	CA TY
0	1	2.330448	No	F	Bachelors	Blue Collar	0.807947	Commercial	0.534077	Spo C
1	2	0.120293	No	M	High School	Blue Collar	-0.964473	Private	0.750925	Miniv
2	3	0.481177	No	F	Bachelors	Clerical	-0.043492	Private	-0.241513	SU
3	4	0.900212	No	F	High School	Lawyer	-1.329803	Private	0.659849	Spo C
4	5	2.377524	No	M	High School	Blue Collar	-0.048799	Commercial	2.111280	Pan Tru

ENCODING

One-Hot-Encoding is used to create dummy variables to replace the categories in a categorical variable into features of each category and represent it using 1 or 0 based on the presence or absence of the categorical value in the record.

This is required to do since the machine learning algorithms only work on the numerical data.

That is why there is a need to convert the categorical column into a numerical one.

get_dummies is the method that creates a dummy variable for each categorical variable.

In [54]:
```
dummies = pd.get_dummies(df_car[['MARITAL STATUS', 'SEX', 'EDUCATION', 'JOB', 'USE
                         columns = ['MARITAL STATUS', 'SEX', 'EDUCATION', 'JOB', '
                         prefix = ['married', 'sex', 'education', 'job', 'use', 'c
```

In [55]:
```
dummies.head()
```

Out[55]:

	married_Yes	sex_M	education_High School	education_Masters	education_PhD	job_Clerical	job_Docto
0	0	0	0	0	0	0	
1	0	1	1	0	0	0	
2	0	0	0	0	0	1	
3	0	0	1	0	0	0	
4	0	1	1	0	0	0	

In [59]:
```
columns = ["MARITAL STATUS", "SEX", "EDUCATION","JOB","USE", "CAR TYPE", "CITY"]
df_car = pd.concat([df_car, dummies], axis=1)

# drop original column 'fuel type' from df

df_car.drop(columns, axis= 1, inplace=True )
```

In [60]:
```
df_car.head()
```

Out[60]:

	INDEX	INCOME	TRAVEL TIME	MILES CLOCKED	CAR AGE	POSTAL CODE	married_Yes	sex_M	education_Hig Scho
0	1	2.330448	0.807947	0.534077	0.137267	-0.277291	0.0	0.0	0
1	2	0.120293	-0.964473	0.750925	-1.052842	-0.277291	0.0	1.0	1
2	3	0.481177	-0.043492	-0.241513	-1.052842	-0.277291	0.0	0.0	0
3	4	0.900212	-1.329803	0.659849	0.930674	-0.277291	0.0	0.0	1
4	5	2.377524	-0.048799	2.111280	0.732322	-0.277291	0.0	1.0	1

5 rows × 90 columns

In [ ]:

B.Time Series Analysis

Getting Google Trends data of keywords such as 'diet' and 'gym' and see how they vary over time while learning about trends and seasonality in time series data.

We will go through the code that we put together during the session step by step. You're not going to do much mathematics but you are going to do the following:

- Source your data
- Wrangle your data
- Exploratory Data Analysis
- Trends and seasonality in time series data *Identifying Trends*Seasonal patterns *First Order Differencing*Periodicity and Autocorrelation

In [1]:
```
# Import packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set()
```

In [4]:
```
df = pd.read_csv('/content/multiTimeline.csv', skiprows=1)
df.head()
```

Out[4]:

	Month	diet: (Worldwide)	gym: (Worldwide)	finance: (Worldwide)
0	2004-01	100	31	48
1	2004-02	75	26	49
2	2004-03	67	24	47
3	2004-04	70	22	48
4	2004-05	72	22	43

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168 entries, 0 to 167
Data columns (total 4 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   Month                 168 non-null    object
 1   diet: (Worldwide)     168 non-null    int64 
 2   gym: (Worldwide)      168 non-null    int64 
 3   finance: (Worldwide)  168 non-null    int64 
dtypes: int64(3), object(1)
memory usage: 5.4+ KB
```

Wrangle Your Data

The first thing that you want to do is rename the columns of your DataFrame df so that they have no whitespaces in them. There are multiple ways to do this, but for now, you'll reassign

to df.columns a list of what you want the columns to be called.

Double check the result of your reassignment by calling df.head():

In [6]:
```
df.columns = ['month', 'diet', 'gym', 'finance']
df.head()
```

Out[6]:

	month	diet	gym	finance
0	2004-01	100	31	48
1	2004-02	75	26	49
2	2004-03	67	24	47
3	2004-04	70	22	48
4	2004-05	72	22	43

Next, you'll turn the **'month' column** into a DateTime data type and make it the index of the DataFrame.

Note that you do this because you saw in the result of the .**info()** method that the **'Month' column** was actually an of data type object. Now, that generic data type encapsulates everything from strings to integers, etc. That's not exactly what you want when you want to be looking at time series data. That's why you'll use **.to_datetime()** to convert the **'month' column** in your DataFrame to a DateTime.

Be careful! Make sure to include the inplace argument when you're setting the index of the DataFrame df so that you actually alter the original index and set it to the 'month' column.

In [7]:
```
df.month = pd.to_datetime(df.month)
df.set_index('month', inplace=True)
```

In [8]:
```
df.head()
```

Out[8]:

month	diet	gym	finance
2004-01-01	100	31	48
2004-02-01	75	26	49
2004-03-01	67	24	47
2004-04-01	70	22	48
2004-05-01	72	22	43

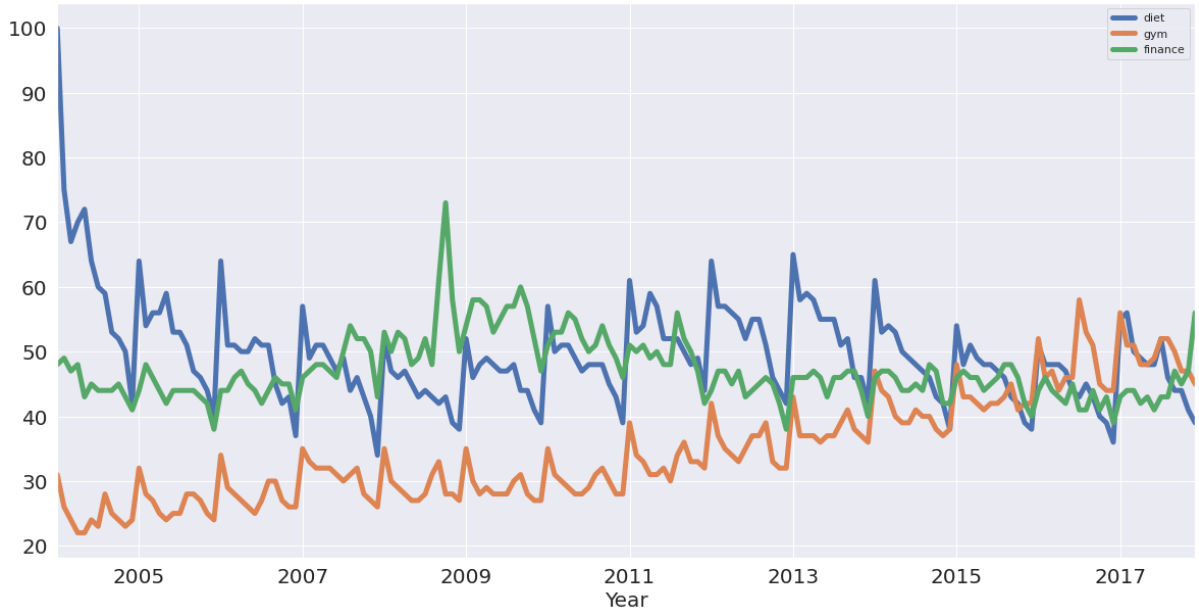
A bit of Exploratory Data Analysis (EDA)

You can use a built-in pandas visualization method .plot() to plot your data as 3 line plots on a single figure (one for each column, namely, **'diet', 'gym', and 'finance'**).

Note that you can also specify some arguments to this method, such as **figsize, linewidthand fontsize** to set the figure size, line width and font size of the plot, respectively.

Additionally, you'll see that what you see on the x-axis is not the months, as the default label suggests, but the years. To make your plot a bit more accurate, you'll specify the label on the x-axis to 'Year' and also set the font size to 20.

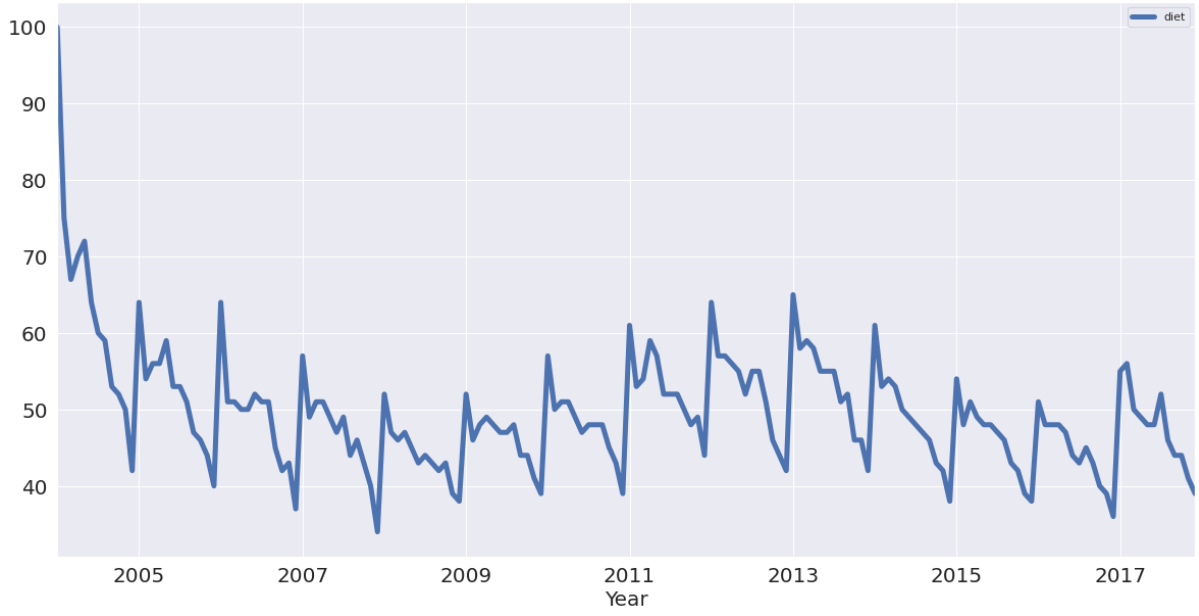
In [9]:
```
df.plot(figsize=(20,10), linewidth=5, fontsize=20)
plt.xlabel('Year', fontsize=20);
```



Numbers represent search interest relative to the highest point on the chart for the given region and time. A value of 100 is the peak popularity for the term. A value of 50 means that the term is half as popular. Likewise a score of 0 means the term was less than 1% as popular as the peak.

If you want, you can also plot the 'diet' column by itself as a time series:

In [10]:
```
df[['diet']].plot(figsize=(20,10), linewidth=5, fontsize=20)
plt.xlabel('Year', fontsize=20);
```



Note: the first thing to notice is that there is seasonality: each January, there's a big jump. Also, there seems to be a trend: it seems to go slightly up, then down, back up and then

back down. In other words, it looks like there are trends and seasonal components to these time series.

With this in mind, you'll learn how to identify trends in your time series!

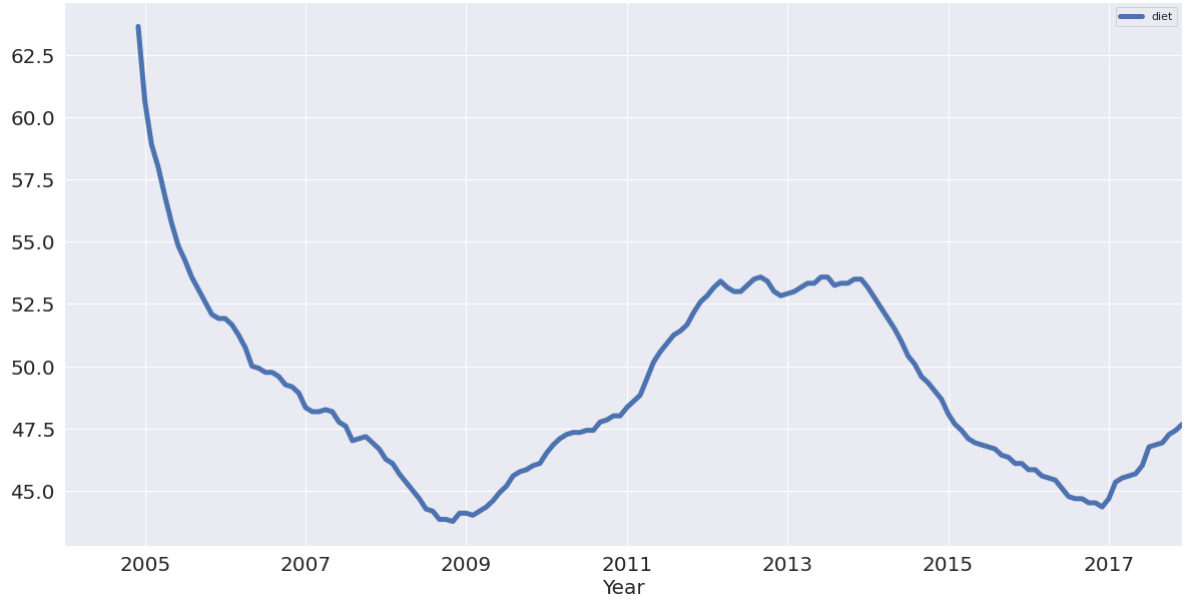
Identifying Trends in Time Series

There are several ways to think about identifying trends in time series. One popular way is by taking a rolling average, which means that, for each time point, you take the average of the points on either side of it. Note that the number of points is specified by a window size, which you need to choose.

What happens then because you take the average is it tends to smooth out noise and seasonality. You'll see an example of that right now. Check out this rolling average of 'diet' using the built-in pandas methods.

When it comes to determining the window size, here, it makes sense to first try out one of twelve months, as you're talking about yearly seasonality.

In [11]:
```
diet = df[['diet']]
diet.rolling(12).mean().plot(figsize=(20,10),  linewidth=5,  fontsize=20)
plt.xlabel('Year', fontsize=20);
```



Note that in the code chunk above you used two sets of squared brackets to extract the 'diet' column as a DataFrame;

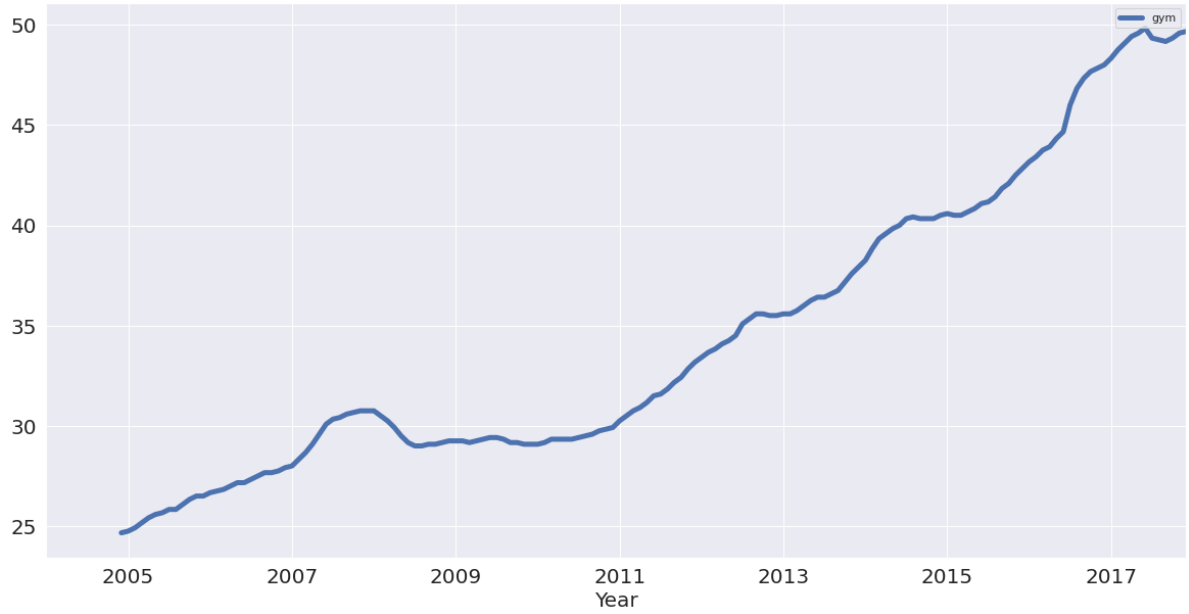
If you would have used one set, like df['diet'], you would have created a pandas Series.

In the code chunk above, you also chained methods: you called methods on an object one after another. Method chaining is pretty popular and pandas is one of the packages that really allows you to use that style of programming to the max!

Now you have the trend that you're looking for! You have removed most of the seasonality compared to the previous plot.

You can also plot the rolling average of 'gym' using built-in pandas methods with the same window size as you took for the 'diet' data:

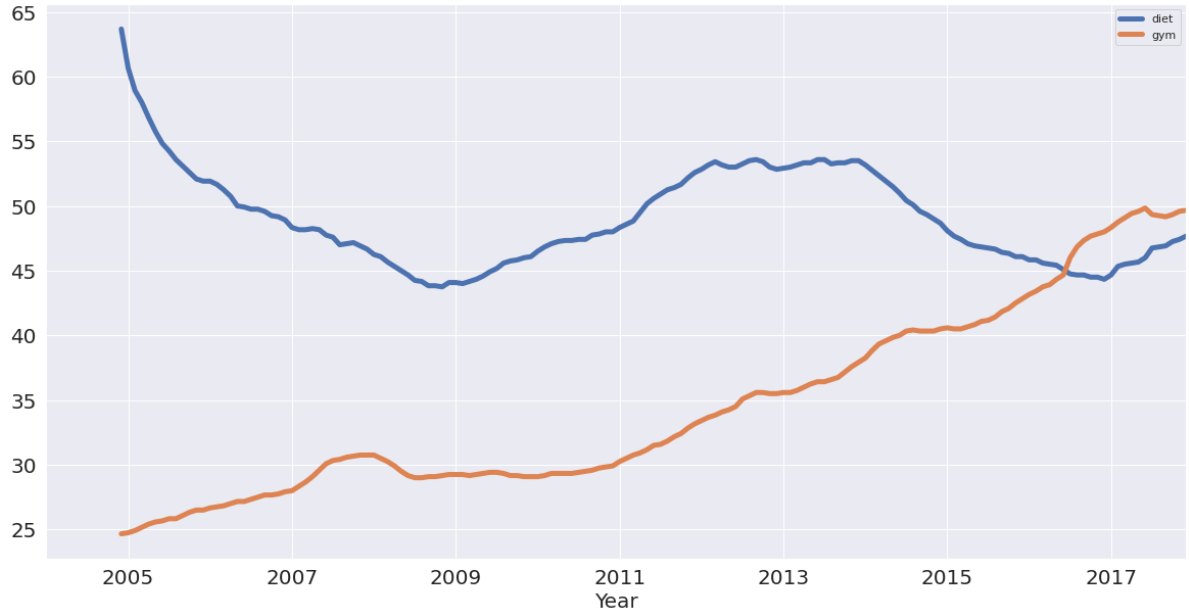
In [12]:
```
gym = df[['gym']]
gym.rolling(12).mean().plot(figsize=(20,10),  linewidth=5,  fontsize=20)
plt.xlabel('Year', fontsize=20);
```



You have successfully removed the seasonality and you see an upward trend for "gym"! But how do these two search terms compare?

You can figure this out by plotting the trends of 'gym' and 'diet' on a single figure:

In [13]:
```
df_rm  =  pd.concat([diet.rolling(12).mean(),  gym.rolling(12).mean()],  axis=1)
df_rm.plot(figsize=(20,10), linewidth=5, fontsize=20)
plt.xlabel('Year',  fontsize=20);
```



You created a new DataFrame df_rm that has two columns with the rolling average of 'diet' and 'gym'. You used the pd.concat() function, which takes a list of the columns as a first argument and, since you want to concatenate them as columns, you also added the axis argument, which you set to 1.

Next, you plotted the DataFrame with the plot() method, just like you did before! So now, removing the seasonality, you see that diet potentially has some form of seasonality, whereas gym is actually increasing!

With the trends in the data identified, it's time to think about seasonality, which is the repetitive nature of your time series. As you saw in the beginning of this tutorial, it looked like there were trends and seasonal components to the time series of the data.

Seasonal Patterns in Time Series Data

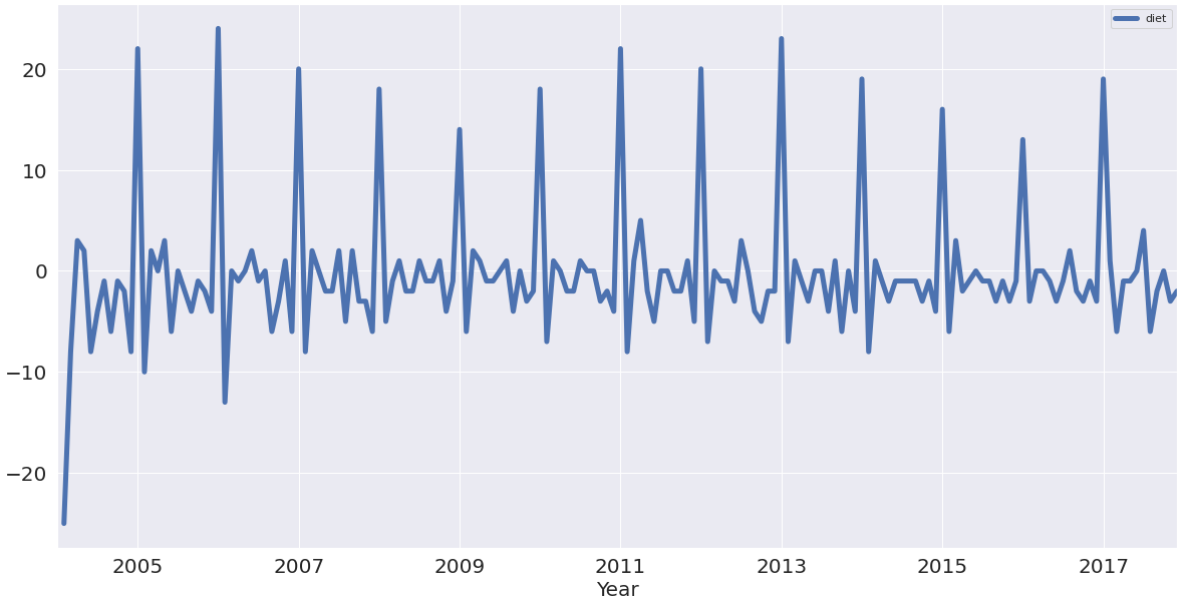
One way to think about the seasonal components to the time series of your data is to remove the trend from a time series, so that you can more easily investigate seasonality. To remove the trend, you can subtract the trend you computed above (rolling mean) from the original signal. This, however, will be dependent on how many data points you averaged over.

Another way to remove the trend is called "differencing", where you look at the difference between successive data points (called "first-order differencing", because you're only looking at the difference between one data point and the one before it).

First-order differencing

You can use pandas and the diff() and plot() methods to compute and plot the first order difference of the 'diet' Series:

In [14]:
```
diet.diff().plot(figsize=(20,10),  linewidth=5,  fontsize=20)
plt.xlabel('Year', fontsize=20);
```



See that you have removed much of the trend and you can really see the peaks in January every year. Each January, there is a huge spike of 20 or more percent on the highest search item you've seen!

Periodicity and Autocorrelation

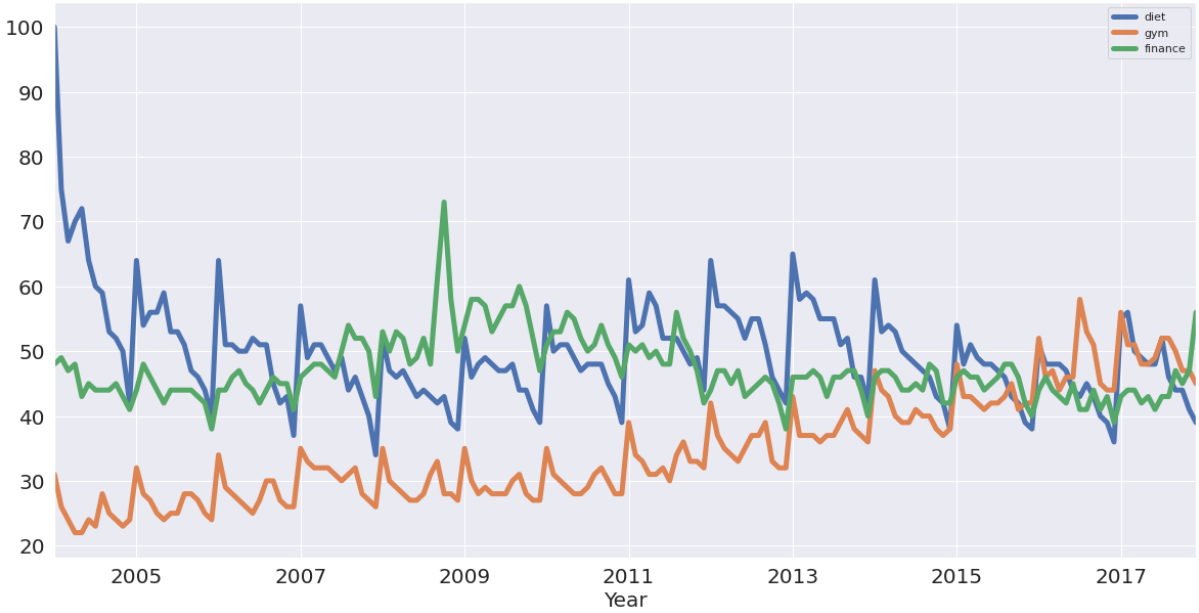
A time series is periodic if it repeats itself at equally spaced intervals, say, every 12 months.

Another way to think of this is that if the time series has a peak somewhere, then it will have a peak 12 months after that and, if it has a trough somewhere, it will also have a trough 12 months after that.

Yet another way of thinking about this is that the time series is correlated with itself shifted by 12 months. That means that, if you took the time series and moved it 12 months backwards or forwards, it would map onto itself in some way.

To start off, plot all your time series again to remind yourself of what they look like:

In [15]:
```
df.plot(figsize=(20,10), linewidth=5, fontsize=20)
plt.xlabel('Year', fontsize=20);
```



In [16]:
```
df.corr()
```

Out[16]:

	diet	gym	finance
diet	1.000000	-0.100764	-0.034639
gym	-0.100764	1.000000	-0.284279
finance	-0.034639	-0.284279	1.000000

Now, what does this above tell you?

Let's focus on 'diet' and 'gym';

They are negatively correlated.

That's very interesting! Remember that you have a seasonal and a trend component.

From the correlation coefficient, 'diet' and 'gym' are negatively correlated.

However, from looking at the times series, it looks as though their seasonal components would be positively correlated and their trends negatively correlated.

The actual correlation coefficient is actually capturing both of those.

What you want to do now is plot the first-order differences of these time series and then compute the correlation of those because that will be the correlation of the seasonal components, approximately.

Remember that removing the trend may reveal correlation in seasonality.

Start off by plotting the first-order differences with the help of .diff() and .plot():

In [17]:
```
df.diff().plot(figsize=(20,10),  linewidth=5,  fontsize=20)
plt.xlabel('Year', fontsize=20);
```

You see that 'diet' and 'gym' are incredibly correlated once you remove the trend. Now, you'll compute the correlation coefficients of the first-order differences of these time series:

In [18]:
```
df.diff().corr()
```

Out[18]:

	diet	gym	finance
diet	1.000000	0.758707	0.373828
gym	0.758707	1.000000	0.301111
finance	0.373828	0.301111	1.000000

Note that once again, there was a slight negative correlation when you were thinking about the trend and the seasonal component. Now, you can see that with the seasonal component, 'diet' and 'gym' are highly correlated, with a coefficient of 0.76.

Autocorrelation

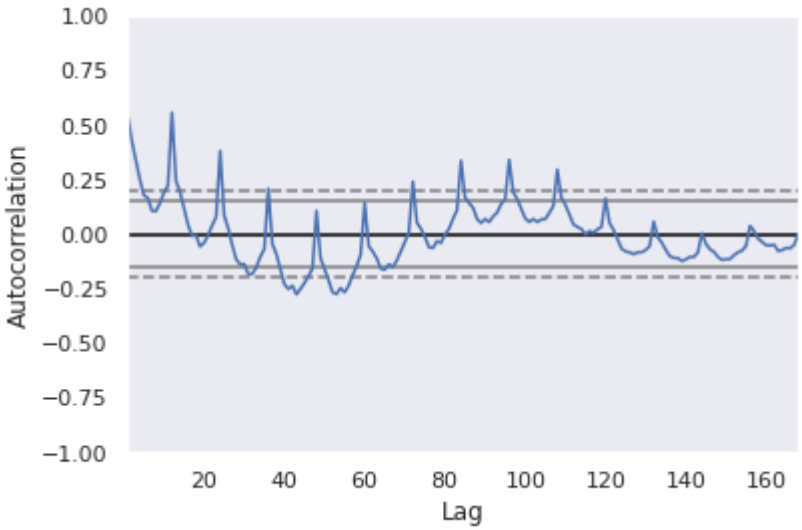
Now you've taken a dive into correlation of variables and correlation of time series, it's time to plot the autocorrelation of the 'diet' series: on the x-axis, you have the lag and on the y-axis, you have how correlated the time series is with itself at that lag.

So, this means that if the original time series repeats itself every two days, you would expect to see a spike in the autocorrelation function at 2 days.

Here, you'll look at the plot and what you should expect to see here is a spike in the autocorrelation function at 12 months: the time series is correlated with itself shifted by twelve months.

Use the plotting interface of pandas, which has the autocorrelation_plot() function. You can use this function to plot the time series 'diet':

In [19]:
```
pd.plotting.autocorrelation_plot(diet);
```



If you included more lags in your axes, you'd see that it is 12 months at which you have this huge peak in correlation.

You have another peak at a 24 month interval, where it's also correlated with itself.

You have another peak at 36, but as you move further away, there's less and less of a correlation.

Of course, you have a correlation of itself with itself at a lag of 0.

The dotted lines in the above plot actually tell you about the statistical significance of the correlation.

In this case, you can say that the 'diet' series is genuinely autocorrelated with a lag of twelve months.

You have identified the seasonality of this 12 month repetition!

In [ ]:
```
```

Practical 9

Aim: Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model toperform this task.

Code

In [10]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
```

In [11]:
```
msg=pd.read_csv('naivetext.csv',names=['message','label'])
```

In [12]:
```
print('The dimensions of the dataset',msg.shape)
```

```
The dimensions of the dataset (18, 2)
```

In [13]:
```
msg['labelnum']=msg.label.map({'pos':1,'neg':0})
X=msg.message
y=msg.labelnum
```

In [14]:
```
#splitting the dataset into train and test data
xtrain,xtest,ytrain,ytest=train_test_split(X,y)
print ('\n the total number of Training Data :',ytrain.shape)
print ('\n the total number of Test Data :',ytest.shape)
```

```
 the total number of Training Data : (13,)

 the total number of Test Data : (5,)
```

In [15]:
```
#output the words or Tokens in the text documents
cv = CountVectorizer()
xtrain_dtm = cv.fit_transform(xtrain)
xtest_dtm=cv.transform(xtest)
print('\n The words or Tokens in the text documents \n')
print(cv.get_feature_names())
df=pd.DataFrame(xtrain_dtm.toarray(),columns=cv.get_feature_names())
```

```
 The words or Tokens in the text documents

['about', 'am', 'amazing', 'an', 'and', 'awesome', 'bad', 'beers', 'boss', 'can',
'deal', 'enemy', 'feel', 'good', 'great', 'he', 'holiday', 'horrible', 'house', 'i
s', 'locality', 'love', 'my', 'of', 'place', 'sandwich', 'sick', 'stay', 'stuff',
'sworn', 'that', 'these', 'this', 'tired', 'to', 'today', 'very', 'view', 'went',
'what', 'with']
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarn
ing: Function get_feature_names is deprecated; get_feature_names is deprecated in
1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [16]:
```
# Training Naive Bayes (NB) classifier on training data.
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)
```

In [17]:
```
#printing accuracy, Confusion matrix, Precision and Recall
print('\n Accuracy of the classifier is',metrics.accuracy_score(ytest,predicted))
print('\n Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))
print('\n The value of Precision', metrics.precision_score(ytest,predicted))
print('\n The value of Recall', metrics.recall_score(ytest,predicted))
```

```
 Accuracy of the classifier is 0.6

 Confusion matrix
[[2 0]
 [2 1]]

 The value of Precision 1.0

 The value of Recall 0.3333333333333333
```

In [17]:

Practical 10

Aim: Perform Text pre-processing, Text clustering,classification with Prediction, Test Score and Confusion Matrix

In [7]:
```
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

Out[7]:
```
True
```

In [1]:
```
import numpy as np
import re
import nltk
from sklearn.datasets import load_files
nltk.download('stopwords')
import pickle
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

In [2]:
```
movie_data  =  load_files(r"/content/sample_data/txt_sentoken")
X, y = movie_data.data, movie_data.target
```

In [8]:
```
documents = []

from nltk.stem import WordNetLemmatizer

stemmer = WordNetLemmatizer()

for sen in range(0, len(X)):
    # Remove all the special characters
    document = re.sub(r'\W', ' ', str(X[sen]))

    # remove all single characters
    document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

    # Remove single characters from the start
    document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

    # Substituting multiple spaces with single space
    document = re.sub(r'\s+', ' ', document, flags=re.I)

    # Removing prefixed 'b'
    document = re.sub(r'^b\s+', '', document)

    # Converting to Lowercase
    document = document.lower()

    # Lemmatization
    document = document.split()
```

```
    document = [stemmer.lemmatize(word) for word in document]
    document = ' '.join(document)

    documents.append(document)
```

In [9]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features=1500, min_df=5, max_df=0.7, stop_words=s
X = vectorizer.fit_transform(documents).toarray()
```

In [12]:
```
from sklearn.feature_extraction.text import TfidfTransformer
tfidfconverter = TfidfTransformer()
X = tfidfconverter.fit_transform(X).toarray()
```

In [13]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidfconverter = TfidfVectorizer(max_features=1500, min_df=5, max_df=0.7, stop_wor
X = tfidfconverter.fit_transform(documents).toarray()
```

In [14]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
```

In [16]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=1000, random_state=0)
classifier.fit(X_train, y_train)
```

Out[16]:
```
RandomForestClassifier(n_estimators=1000, random_state=0)
```

In [17]:
```
y_pred = classifier.predict(X_test)
```

In [18]:
```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

```
[[197   0]
 [ 76   9]]
              precision    recall  f1-score   support

           1       0.72      1.00      0.84       197
           2       1.00      0.11      0.19        85

    accuracy                           0.73       282
   macro avg       0.86      0.55      0.51       282
weighted avg       0.81      0.73      0.64       282

0.7304964539007093
```

In [ ]:
```
```

ROBOTIC PROCESS AUTOMATION

INDEX

Sr No.	Title	Date	Sign
1.	a. Create a simple sequence based project. b. Create a flowchart-based project. c. Create an UiPath Robot which can empty a folder in Gmail solely on basis ofrecording		
2.	a. Automate UiPath Number Calculation (Subtraction, Multiplication, Divisionof numbers). b.Create an automation UiPath project using different types of variables (number, datetime, Boolean, generic, array, data table)		
3.	a. Create an automation UiPath Project using decision statements b. Create an automation UiPath Project using looping statements		
4.	a. Automate any process using basic recording. b. Automate any process using desktop recording. c. Automate any process using web recording		
5.	Consider an array of names. We have to find out how many of them start withthe letter "a". Create an automation where the number of names starting with "a"is counted and the result is displayed.		
6.	a. Create an application automating the read, write and append operation on excel file.		

	b. Automate the process to extract data from an excel file into a data table andvice versa		
7.	a. Implement the attach window activity. b. Find different controls using UiPath. c. Demonstrate the following activities in UiPath: 1. Mouse (click, double click and hover) 2. Type into 3. Type Secure text		
8.	a. Demonstrate the following events in UiPath: 1. Element triggering event 2. Image triggering event 3. System Triggering Event b.Automate the following screen scraping methods using UiPath 1. Full Test 2. Native 3. OCR c. Install and automate any process using UiPath with the following plug-ins: 1.Java Plugin 2.Mail Plugin 3.PDF Plugin 4.Web Integration 5.Excel Plugin 6.Word Plugin d. Credential Management		
9.	a. Automate the process of send mail event (on any email). b. Automate the process of launching an assistant bot on a keyboard event. c. Demonstrate the Exception handing in UiPath. d. Demonstrate the use of config files in UiPath		

10.	a. Automate the process of logging and taking screenshots in UiPath. b.Automate any process using State Machine in UiPath c. Demonstrate the use of publish utility. d. Create and provision Robot using Orchestrator.		

Practical 1A

Create a simple sequence based project.

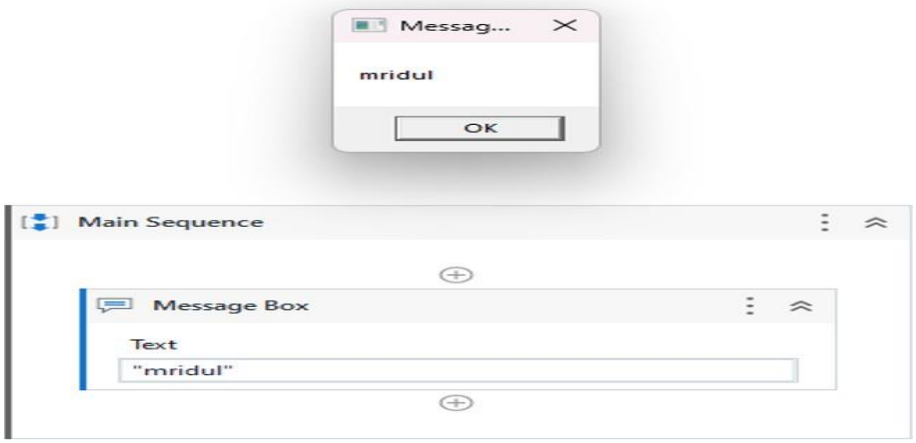
Theory:

A sequence is a basic automation process in RPA. It is used when the process is divided into steps and sequential. It is simpler and easier to use.

Steps:

1. Open UI path studio.
2. In uipath studio start page, under new project click on process
3. A dialog box appears, give a name to the process and select the language as C# and click on create.
4. Once a process is created, go to activity panel and add new sequence
5. Add name to the sequence and click on create. A sequence is created.
6. Once the sequence is created, add a message box.
7. Add the text you want to display and click on debug at the top ribbon.
8. This is how we create a simple sequence.

Output:



Practical 1B

Create a flowchart-based project.

Theory:

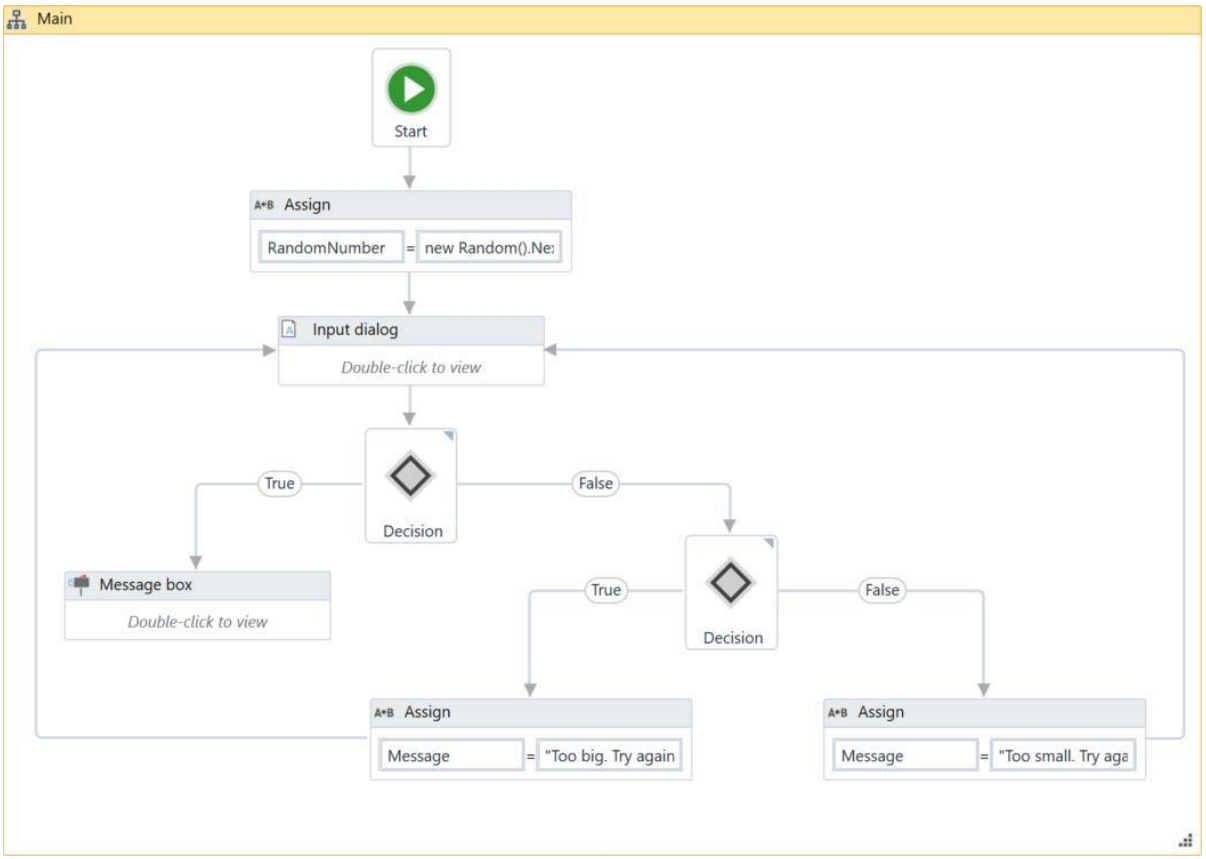
Flowcharts can be used in a variety of settings, from large jobs to small projects that you can reuse in other projects. The most important aspect of flowcharts is that, unlike sequences, they present multiple branching logical operators, that enable you to create complex business processes and connect activities in multiple ways.

Steps:

1. **Create a blank process and from the Design tab, in the File group, select New > Flowchart. The New Flowchart window is displayed.**
2. In the **Name** field type a name for the automation, such as "First Flowchart", and leave the default project location or add a subfolder. Click **Create**. The **Designer** panel is updated accordingly.
3. Create two **Int32** variables (RandomNumber, GuessNumber) and a **String** one (Message).
4. Set the default value of the Message variable to "Guess a number from 1 to 999." The RandomNumber stores a random number between 1 and 999, GuessNumber stores the user's guess and Message stores the message that is going to be displayed to prompt the user.
5. Add an **Assign** activity to the **Designer** panel, and connect it to the **Start** node.
6. In the **Properties** panel, in the **To** field add the RandomNumber variable.
7. In the **Value** field, type new Random().Next(1,999).
8. Add an **Input Dialog** activity to the **Designer** panel and connect it to the **Assign** one.
9. In the **Properties** panel, in the **Label** field, add the Message variable.
10. In the **Result** field, add the GuessNumber variable. This activity asks and stores the user's guesses in the GuessNumber variable.
11. Add a **Flow Decision** activity and connect it to the **Input Dialog**. This activity enables you to tell the user if he correctly guessed the number or not.
12. In the **Properties** panel, in the **Condition** field, type GuessNumber = RandomNumber. This enables you to verify if the number added by the user is the same as the randomly-generated one.
13. Add a **Message Box** activity and connect it to the **True** branch of the **Flow Decision**.
14. In the **Properties** panel, in the **Text** field, type "Congratulations! You guessed correctly! The number was " + RandomNumber.ToString + ".". This is the message that is going to be displayed if the user correctly guessed the number.
15. Add a new **Flow Decision** activity and connect it to the **False** branch of the previously added **Flow** Decision.
16. In the **Properties** panel, in the **Condition** field, type GuessNumber > RandomNumber. This activity enables you to check if the number the user added is bigger than the randomly-generated one.
17. In the **DisplayName** field, type **Comparison**. This enables you to easily to tell the difference between the two **Flow Decisions** used.
18. Add an **Assign** activity and connect it to the **True** branch of the **Comparison** activity.
19. In the **To** field, type the Message variable, and in the **Value** field, type a message indicating that the guess was too high, such as "Too big. Try again.".
20. Select the **Assign** activity and press Ctrl+C. The entire activity and its properties are copied to the Clipboard.
21. Press Ctrl + V. A duplicate of the previous **Assign** activity is displayed.

22. Connect it to the **False** branch of the **Comparison** activity and, in the **Properties** panel, in the **Value** field, type "Too small. Try again.".
23. Connect the **Assign** activities created at steps 18-22 to the **Input Dialog**. A loop is created, asking the user to type a smaller or bigger number, until he guesses correctly.
The final project should look as in the screenshot below.

OUTPUT :



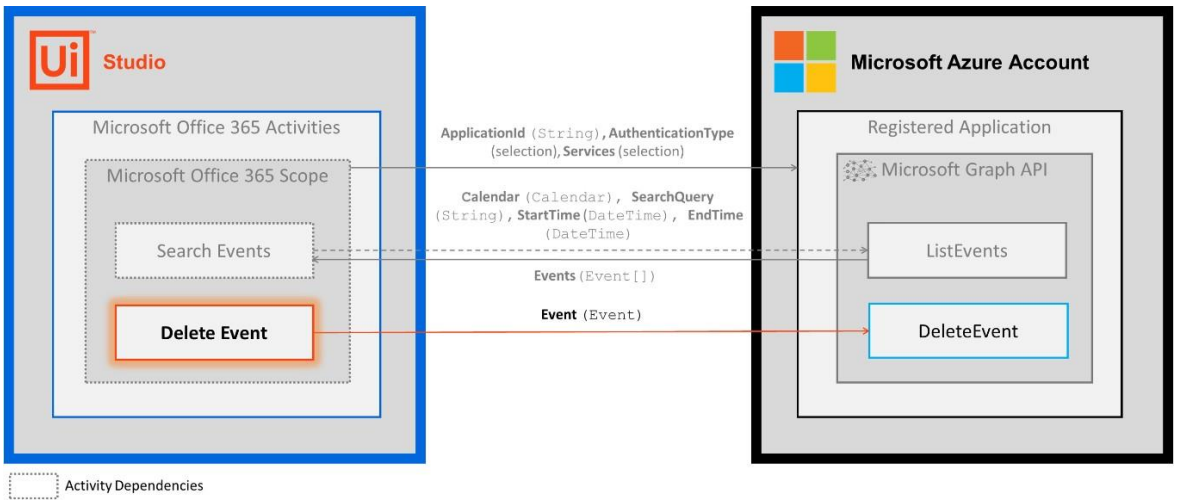
Practical 1C

An UiPath Robot which can empty a folder in Gmail solely on basis of recording.

STEPS :

1. Complete the Setup steps.
2. Add the Microsoft Office 365 Scope activity to your project.
3. Add an activity or run an external process that outputs a `Event` object (e.g., Search Events).
4. Add the **Delete Event** activity inside the **Microsoft Office 365 Scope** activity.
5. Enter values for the Input properties.
6. Run the activity.
   - Your input property values are sent to the DeleteEvent API.

OUTPUT :



Practical 2A

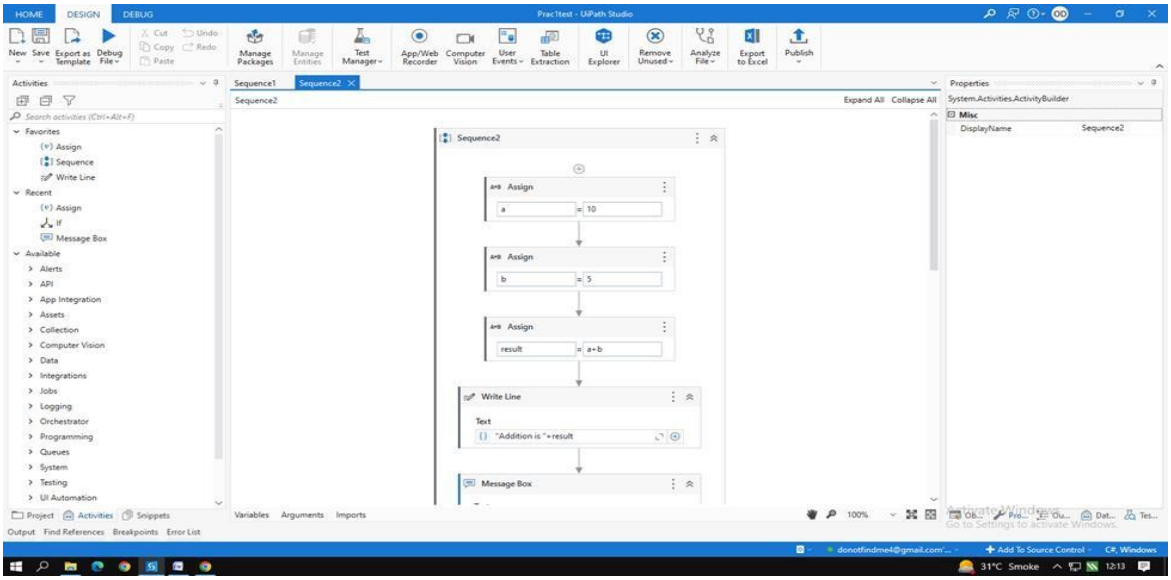
Automate UiPath Number Calculation (Subtraction, Multiplication, Division ofnumbers).

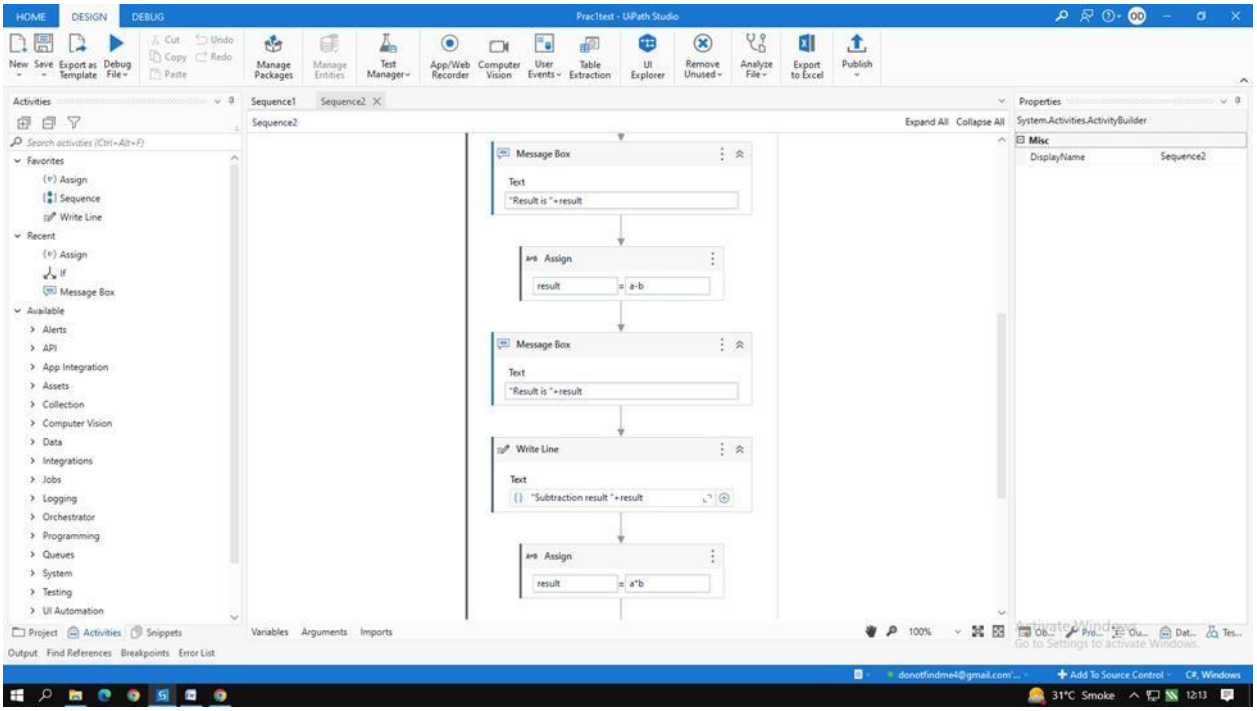
Steps:

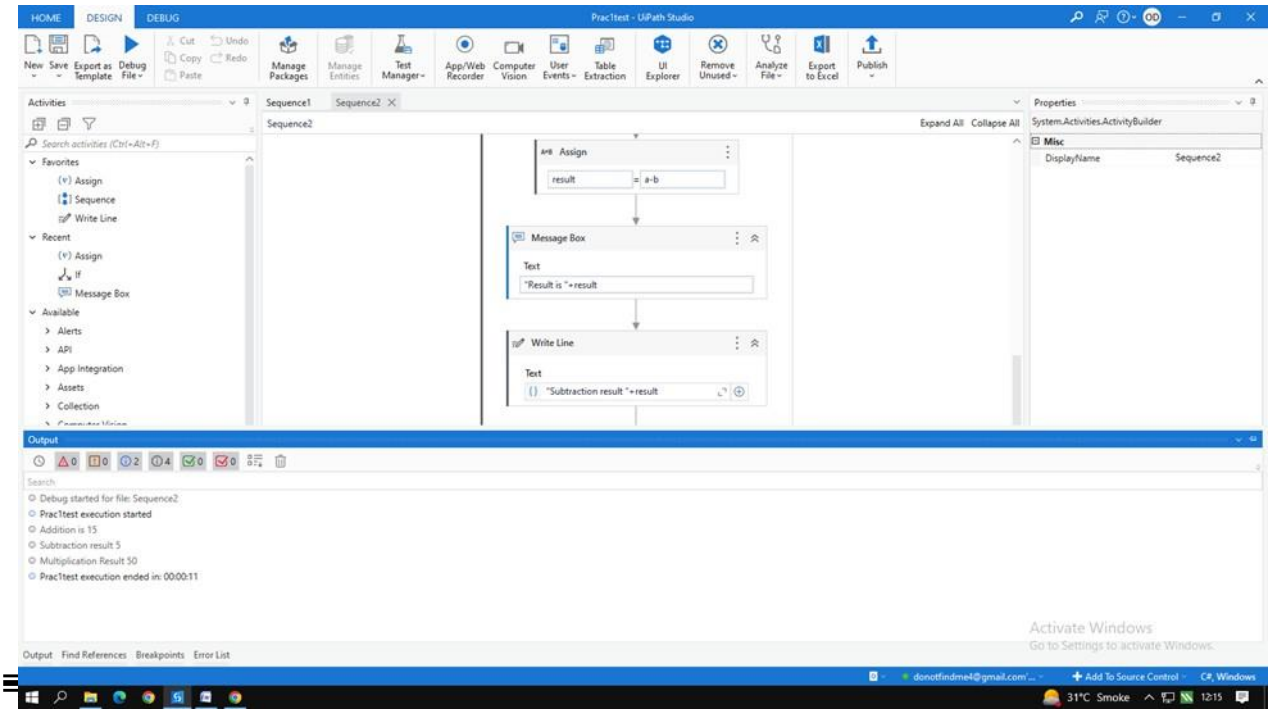
1. Create a new sequence.
2. In the sequence, drag and drop an assign box to assign a variable. Assign a variable “a” with value 10.
3. On the variables tab, create the variable and assign the variable type as Int32.
4. Repeat steps 2 and 3 to create another variable “b” of integer type and give it the value 15.
5. Add a third assign box and create a variable “result” that will store the value of the operation between a and b. (a+b). This variable will also be integer.
6. Once done, drag and drop a write line box and add: “Addition is” + result since the output will be in string format.
7. Repeat steps from 5 and 6 thrice and change the operation as result=a-b, result=a*b, result=a/b.
8. Run the automation.

OUTPUT :

This will give the output of addition, subtraction, multiplication, division between variables a and b




HOME
DESIGN
DEBUG
Prac1test - UiPath Studio
Activities
Sequence1
Sequence2
Properties
Message Box
Text
"Result is "+result
Assign
result = a-b
Message Box
Text
"Result is "+result
Write Line
Text
"Subtraction result "+result
Assign
result = a*b
Variables
Arguments
Imports
Output
Find References
Breakpoints
Error List


HOME
DESIGN
DEBUG
Prac1test - UiPath Studio
Activities
Sequence1
Sequence2
Properties
Assign
result = a-b
Message Box
Text
"Result is "+result
Write Line
Text
"Subtraction result "+result
Output
Debug started for file: Sequence2
Prac1test execution started
Addition is 15
Subtraction result 5
Multiplication Result 50
Prac1test execution ended in: 00:00:11
Output
Find References
Breakpoints
Error List

Practical 2B

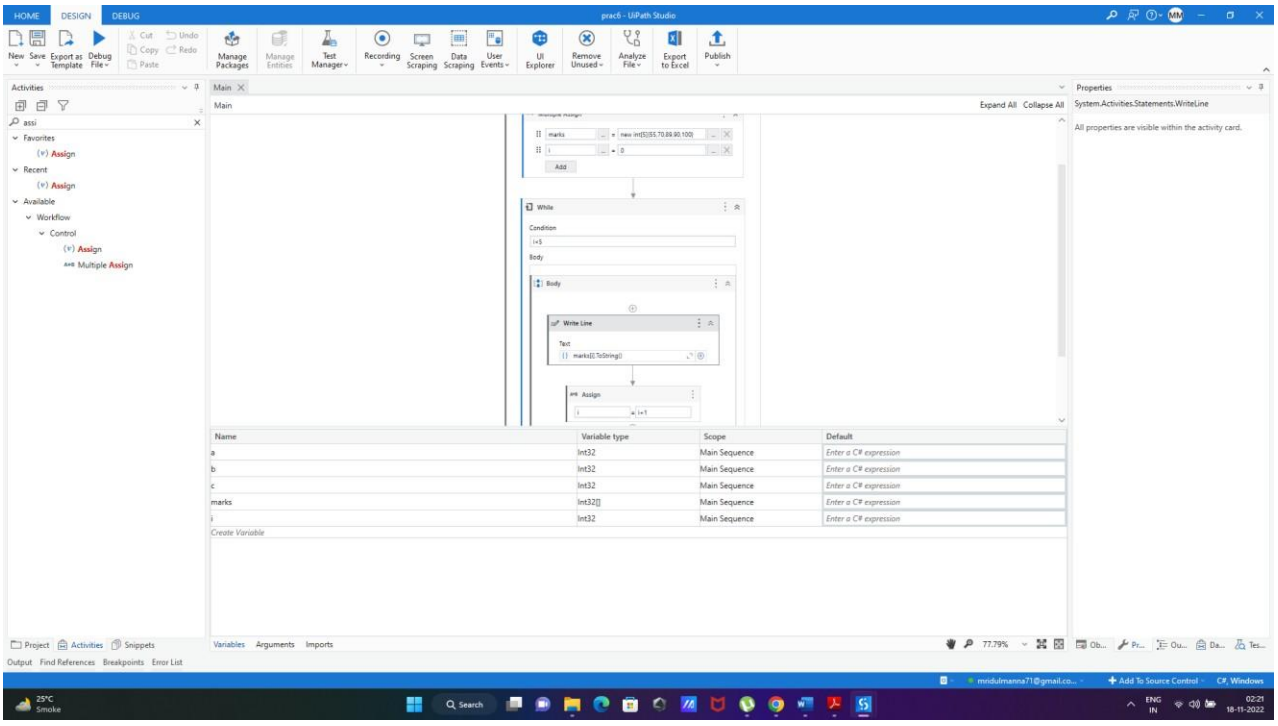
An automation UiPath project using different types of variables (number, datetime,Boolean, generic, array, data table)

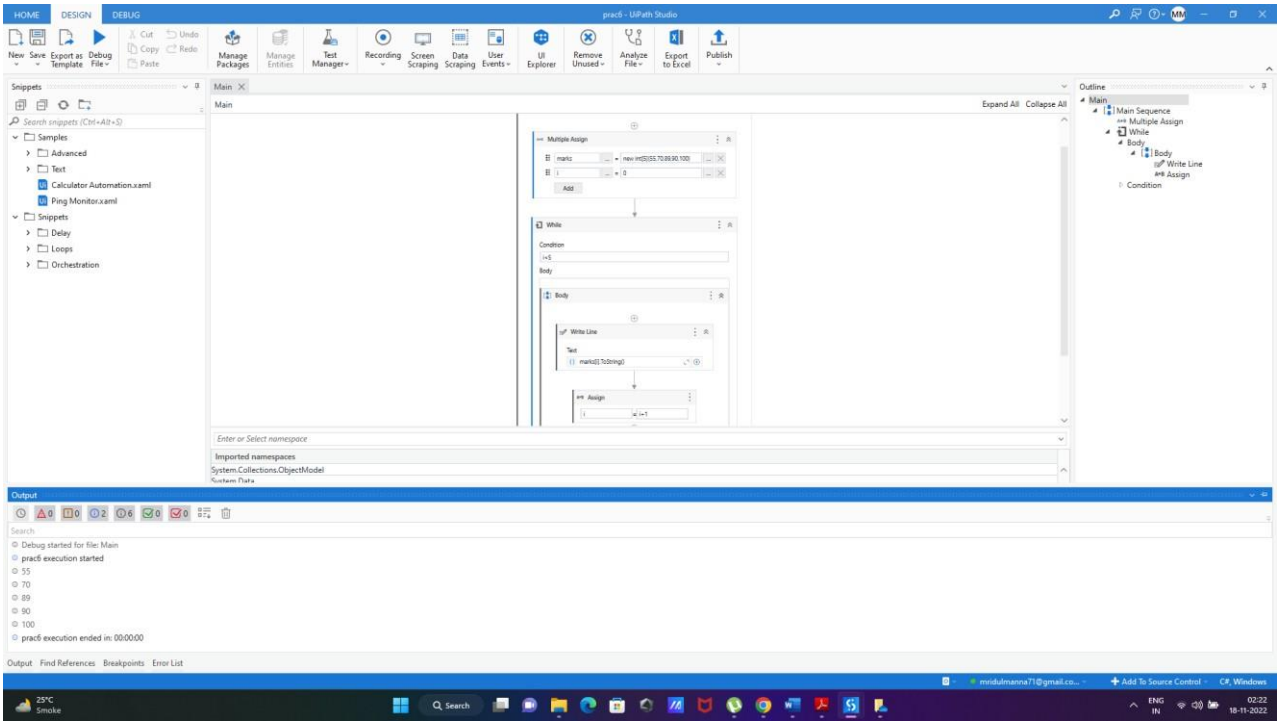
STEPS :

To create array

1. Create a new sequence.
2. Drag and drop multiple assign activity.
3. Create an array variable a where,
    - marks = new int[5]{99,100,50,22,98}
4. Once done, go to variable option and change the data type of the array to array of [T]. Click on the option and select the array type as int 32.
5. Create another variable i=0 where i will be the iteration.
6. Drag and drop while activity. In the condition part add i<5
7. In the body drag and drop writeline and add marks[i].ToString() to display the array
8. Finally drag and drop a new assign and add i=i+1 to increase the iteration by 1. This will display all the members of the array.
9. Run the automation and check the output.

OUTPUT :




HOME
DESIGN
DEBUG
prac6 - UiPath Studio
Snippets
Samples
Advanced
Text
Calculator Automation.xaml
Ping Monitor.xaml
Snippets
Delay
Loops
Orchestration
Main
Expand All
Collapse All
Multiple Assign
While
Body
Write Line
Assign
Outline
Main Sequence
Condition
Enter or Select namespace
Imported namespaces
System.Collections.ObjectModel
Output
Debug started for file: Main
prac6 execution started
55
70
89
90
100
prac6 execution ended in: 00:00:00
Output
Find References
Breakpoints
Error List
25°C
Smoke
Search
ENG
IN
02:22
18-11-2022

Practical 3A

An automation UiPath Project using decision statements.

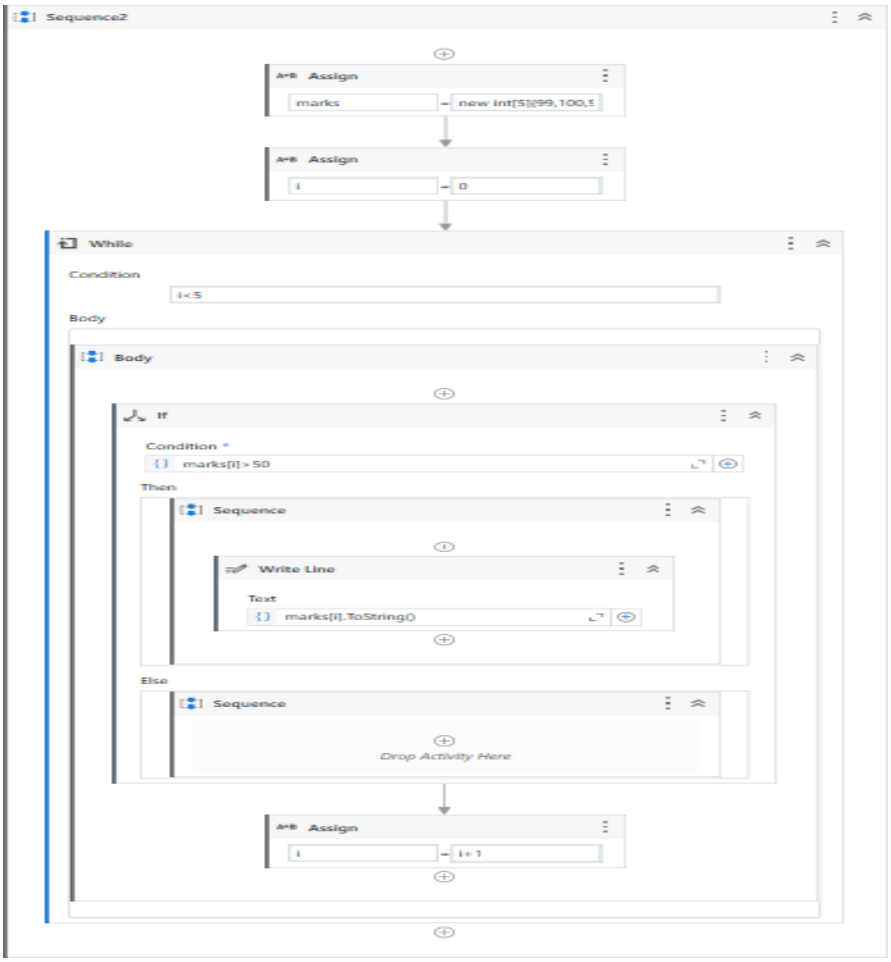
Theory:

The If activity contains a statement and two conditions. The first condition (the activity in the Then section) is executed if the statement is true, while the second one (the activity in the optional Else section) is executed if the statement is false.

If activities can be useful to make decisions based on the value of variables.

STEPS:

1. Create a new sequence.
2. Drag and drop multiple assign activity.
3. Create an array variable a where,
   marks = new int[5]{99,100,50,22,98}
4. Once done, go to variable option and change the data type of the array to array of [T]. Click on the option and select the array type as int 32.
5. Create another variable i=0 where I will be the iteration.
6. Drag and drop while activity. In the condition part add i<5
7. In the body of the while activity, drag and drop the if activity and add the condition: Marks[i]>50
8. In the "then" panel drag and drop writeline activity and add marks[i].ToString() to display the array
9. Finally drag and drop a new assign outside the if activity and add i=i+1 to increase the iteration by 1. This will display all the members of the array.
10. Run the automation and check the output.



Output:

This automation will print the numbers in the array greater than 50.

```
Debug started for file: Sequence2
rpa1-2 execution started
99
100
98
rpa1-2 execution ended in: 00:00:01
```

Practical 3B

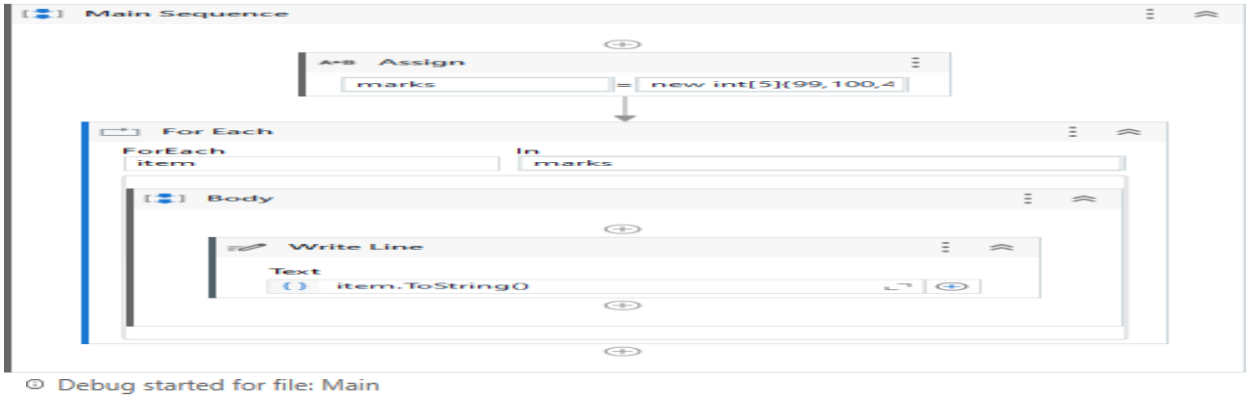
An automation UiPath Project using looping statements.

STEPS :

For-each activity

1. Create a new sequence.
2. Drag and drop assign activity.
3. Create an array variable marks where,
    a. marks = new int[5]{99,100,50,22,98}
4. Once done, go to variable option and change the data type of the array to array of [T]. Click on the option and select the array type as int 32.
5. Drag and drop for each activity.
6. In the ForEach activity add the following statements:
    - In “for each” field add item
    - Add marks in “In” field
7. In the body drag and drop writeline and add item.ToString() to display the array
8. Run the automation and check the output.

OUTPUT : This automation will print an array using for each activity.




```
Debug started for file: Main
foreach execution started
99
100
40
50
65
foreach execution ended in: 00:00:00
```

Practical 4A

Automate any process using basic recording.

Theory :

Recording is an important part of UiPath Studio, that can help you save a lot of time when automating your business processes. This functionality enables you to easily capture a user's actions on the screen and translates them into sequences.

These projects can be modified and parameterized so that you can easily replay and reuse them in as many other processes as you need.

All user interface elements are highlighted while you record, as you can see in the following screenshot, so that you can be sure the correct buttons, fields or menus are selected.

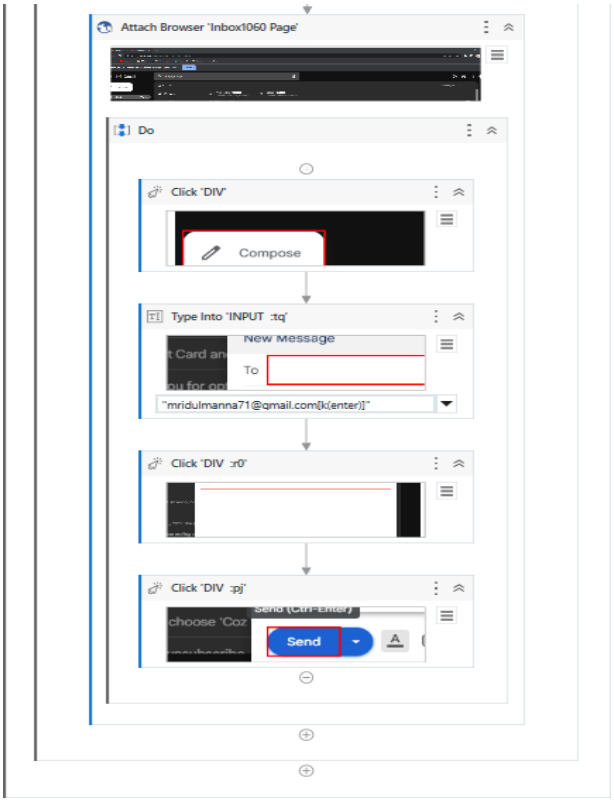
Steps:

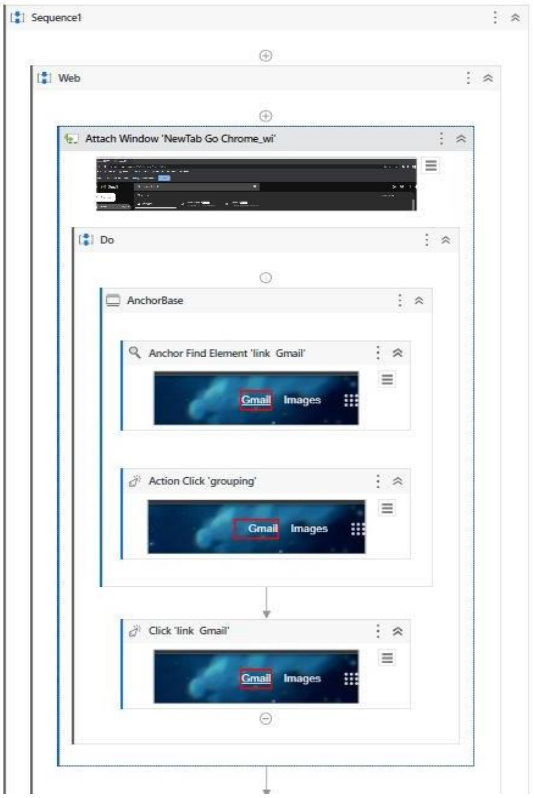
Recording steps to send an email

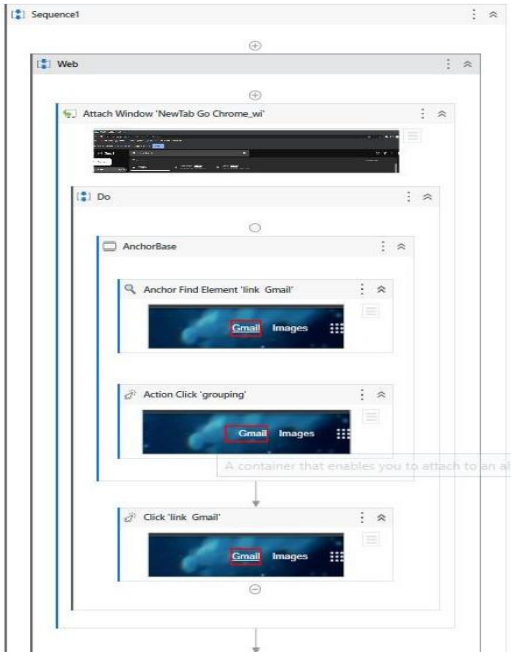
1. Create a new sequence.
2. On the top in the ribbon, click on the recording option. A drop down will appear.
3. Select web recording option from the dropdown.
4. Open the Google chrome window.
5. Start recording
6. On the chrome window, click on gmail link
7. After clicking on gmail link, the gmail webpage opens. Click on compose.
8. After compose, a box appears to enter the mail.
9. Click on the To: option and enter the email address you want the mail to send to
10. After adding the receiver's email address, press enter
11. Click on the body of the email and add any text and click on send
12. Press esc and click on save and exit
13. Run the automation and check the output

Output:

This system will send an email automatically using automation.







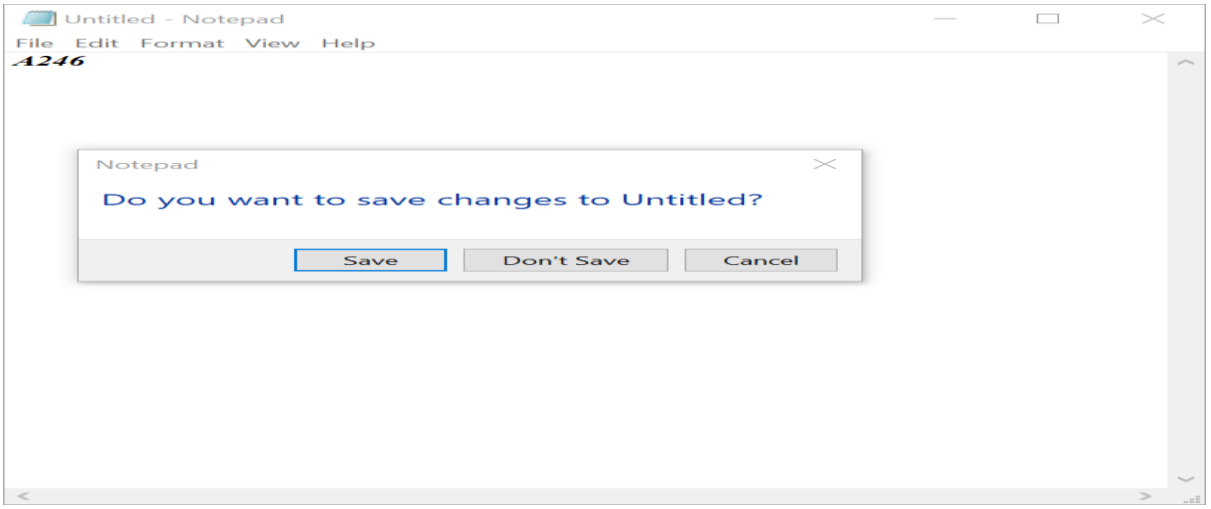
Practical 4B

Automate any process using desktop recording.

STEPS :

1. Open Notepad.
2. In UiPath Studio, create a new sequence.
3. a. In the **Design** ribbon tab, in the **Wizards** group, **select Record > Basic**. The **Basic Recording** toolbar is displayed and the main view is minimized.
   b. In the **Design** ribbon tab, in the **Wizards** group, **select Record > Desktop**. The **Desktop Recording** toolbar is displayed and the main view is minimized.
4. In the **Wizards** group, click **Automatic Recorder**. The automating recording process starts.
5. In Notepad, click on the main panel. A pop-up window is displayed.
6. Type a custom text and press Enter. The string is displayed in Notepad.
7. From the **Format** menu, select **Font**. The **Font** window is displayed.
8. Select a different font style, such as Bold Italic, and click **OK**.
9. Press Esc two times. You exit the recording view and the saved project is displayed in the **Designer** panel.
10. Press F5. The automation is executed as expected.
11. Add an **Open Application** activity between **Excel Application Scope** and the Recording sequence.
12. Use **Indicate window on screen** to select the active **Notepad** window.
13. Place the Recording sequence inside the **Open Application** activity.
14. Add a **Close Application** activity after **Open Application**.
15. Use **Indicate window on screen** again to select the active **Notepad** window to be closed.
16. Make sure the **OffsetX** and **OffsetY** properties (**Cursor Position**) are empty.
    What was added to the project should look as in the following screenshot.

OUTPUT :



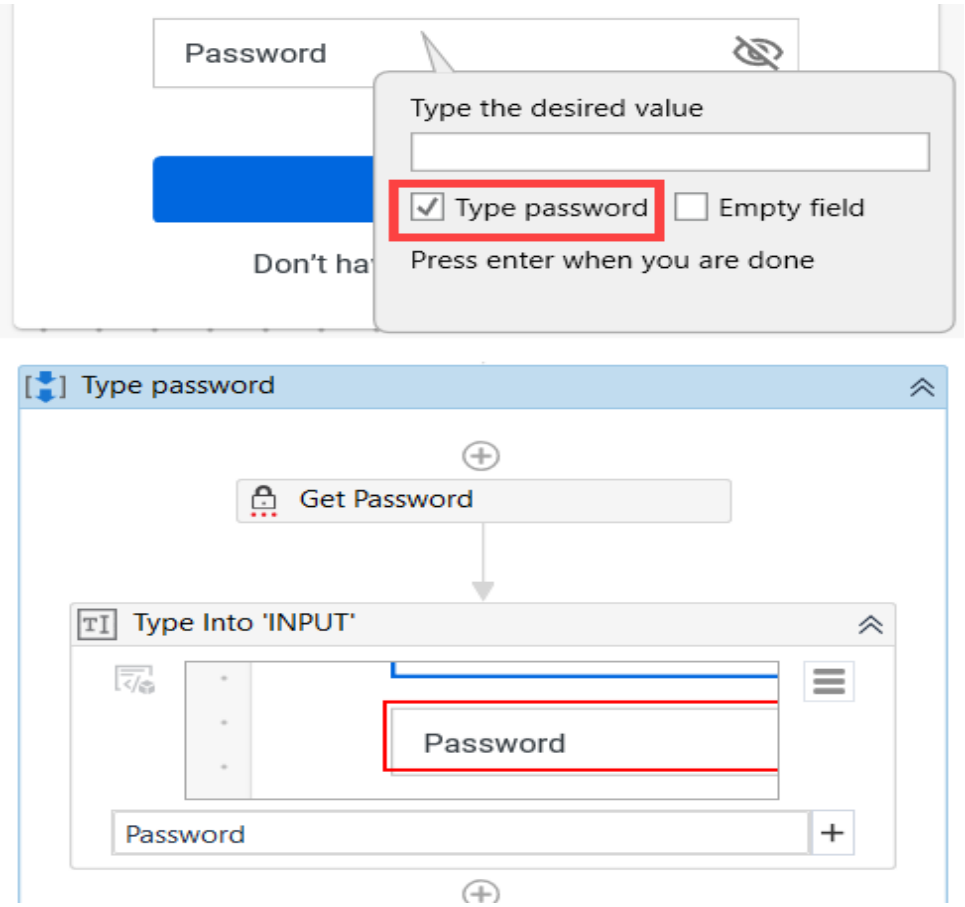
Practical 4C

Automate any process using web recording.

STEPS :

1. Open an Internet Explorer instance and navigate to https://academy.uipath.com.
2. In UiPath Studio, create a new sequence.
3. Add an **Open Browser** activity to the **Designer** panel.
4. Select the activity and, in the **Url** field, write https://academy.uipath.com.
5. In the **Design** tab, in the **Wizards** group, select **Recording > Web**. The **Web Recording** toolbar is displayed and the main view is minimized.
6. Click **Record**. The automating recording process starts.
7. In Internet Explorer, click **Login/ Sign up**, and then select **Continue with Email**.
8. Enter your email address and password.
9. Click **Login** and press Esc two times. The recording is saved and displayed in the **Designer** panel.
10. Close Internet Explorer manually.
11. In Studio, add a **Close Tab** activity as the last activity in the **Attach Browser** container.
12. Press F5. The automation is executed as expected.

OUTPUT :



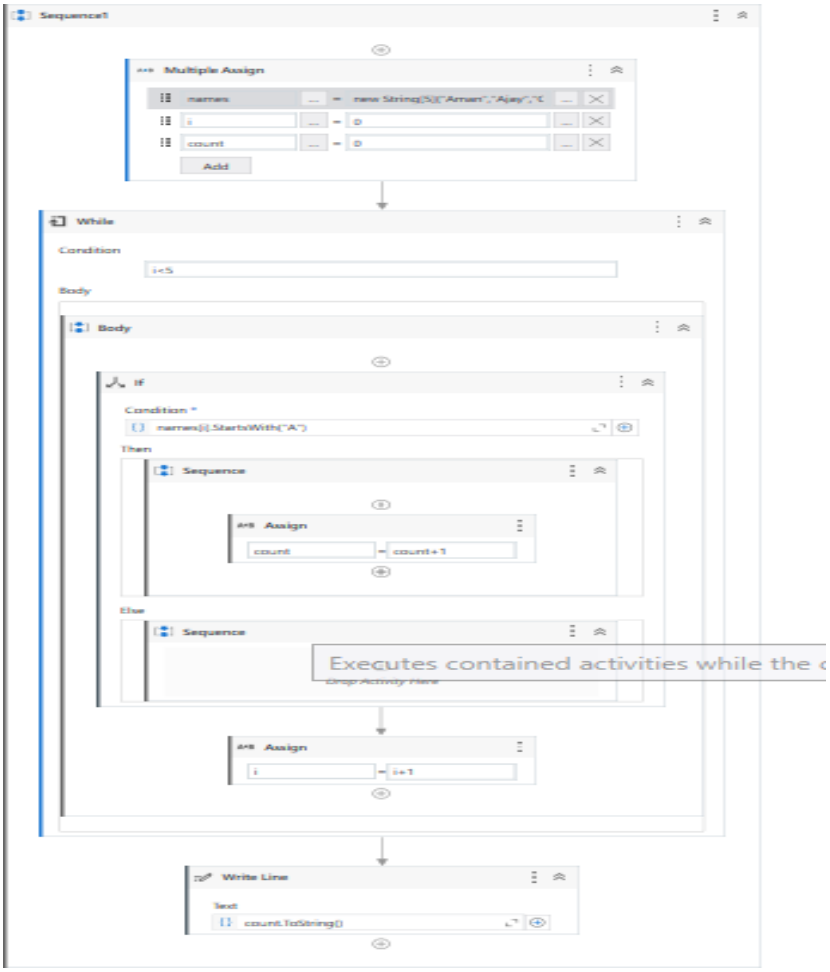
Practical 5A

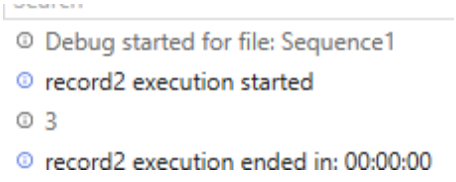
Consider an array of names. We have to find out how many of them start with the letter "a". Create an automation where the number of names starting with "a" is counted andthe result is displayed.

Steps:

1. Create a new sequence.
2. Drag and drop multiple assign activity
3. Create a string array and add the names you wish to add
   Names=New String[5]{“Aman”,”Ajay”,”Omkar”,”Ashish”,”Tejas”}
4. Once the array is created, go to the variable tab and change the data type of the array to Array of [T] and select the datatype as string.
5. Once array is created, create a new variable i=0 for iteration and count=0 and select the datatype as Int32 for both the variables.
6. Once done, Drag and drop a while activity from activities panel.
7. Add condition as i<5
8. In the body panel of the while activity, drag and drop an if activity
9. Add the following condition in the if activity:
   Names[i].StartsWith(“A”)
10. Drag and drop an assign activity in the “then” panel of the if activity and add
    Count=count+1(This will increase the counter by 1 if the condition is true)
11. After the else panel in the if activity, drag and drop an assign activity and add
    i=i+1
12. Outside the while loop in the main sequence, drag and drop writeline activity and add:
    Count.ToString().
13. Run the automation and check the output.

Output:


Sequence1
Multiple Assign
names
i
count
Add
While
Condition
i<5
Body
Body
If
Condition *
names[i].StartsWith("A")
Then
Sequence
Assign
count
count+1
Else
Sequence
Executes contained activities while the
Assign
i
i+1
Write Line
Text
count.ToString()


Debug started for file: Sequence1
record2 execution started
3
record2 execution ended in: 00:00:00

Practical 6A

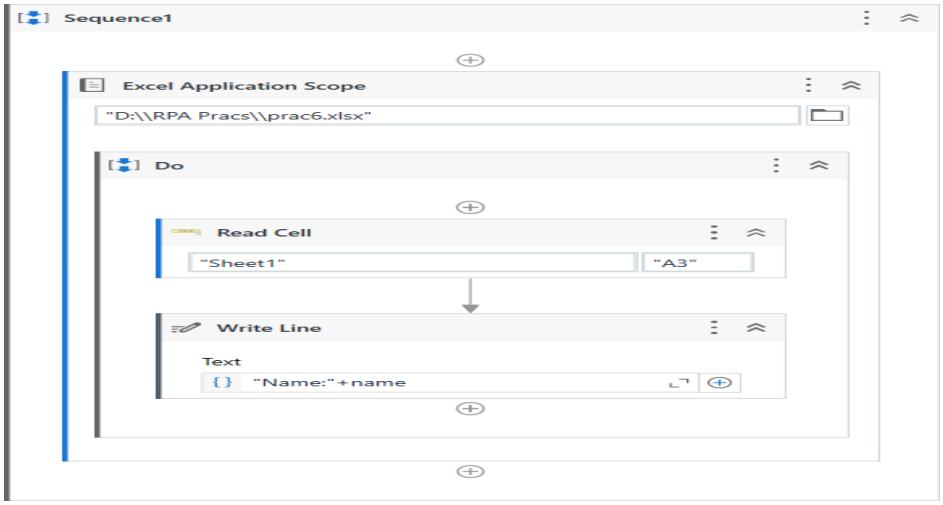
Create an application automating the read, write and append operation on excel file.

Steps:

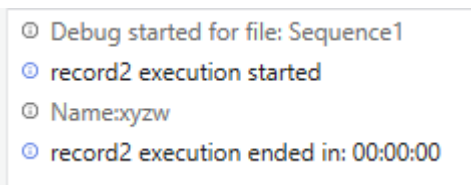
1. Create a new sequence.
2. Create a excel file and add 2 columns of data

	A	B
1	fname	roll
2	abcd	1234
3	xyzw	5678
4	abcd	1234
5		

3. Drag and drop Excel application scope from the activities panel
4. In the application scope, browse the excel file in the system and add the path of the file.
5. In the DO panel of the excel application scope, Drag and drop read cell activity.
6. It will automatically take the sheet as “Sheet1” and cell as “A1” as the initial point .
7. Right click on the “Sheet1” and select create variable and add a variable as name and add any cell number you want to print.
8. Drag and drop a Writeline activity and add : “Name:”+name
9. Run the automation and check the output.



Output: The system will print the data available in cell A3

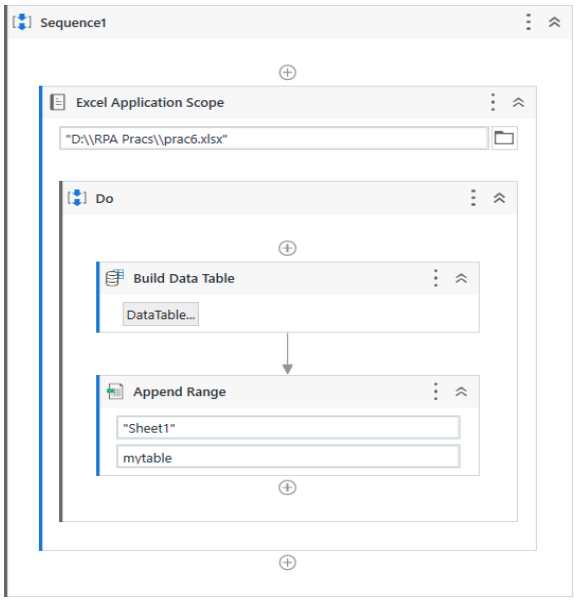


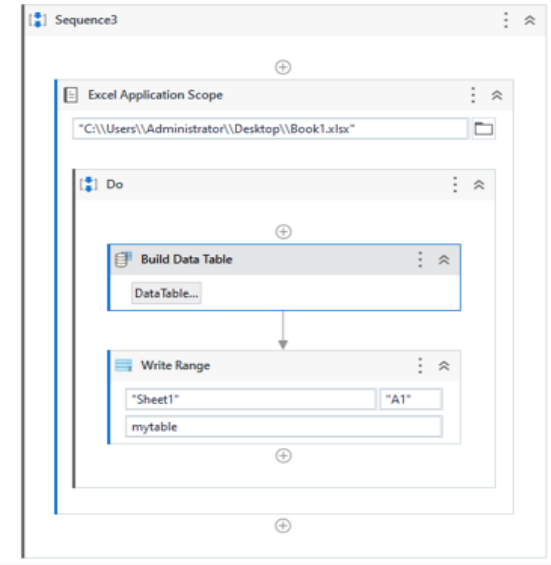
Practical 6B

Automate the process to extract data from an excel file into a data table and vice versa

Steps:

1. Create a new sequence.
2. Create a excel file and keep the file empty
3. Drag and drop Excel application scope from the activities panel
4. In the application scope, browse the excel file in the system and add the path of the file.
5. Drag and drop Build data table activity and add a datatable and click on create option
6. On the right side of the page, right click on the output option and create a variable “mytable”
7. In the DO panel of the excel application scope, Drag and drop append range / write range activity.
8. It will automatically take the sheet as “Sheet1” and cell as “A1” as the initial point if you use write range.
9. It will also give a text box to enter the data you wish to write in the excel file. Add the variable name as mytable.
10. Run the automation and check the output.





Output: This automation will write the data of the data table in the excel file and append the data the number of times you run the automation.

	A	B
1	abcd	1
2	xyzw	2
3	defg	2
4	abcd	1
5	xyzw	2
6		

Practical 7A

Implement the attach window activity.

Attach Window

SUGGEST EDITS

UiPath.Core.Activities.WindowScope
A container that enables you to attach to an already opened window and perform multiple actions within it. This activity is also automatically generated when using the Desktop recorder.

Properties

Input

- **Selector** - Text property used to find a particular UI element when the activity is executed. It is actually a XML fragment specifying attributes of the GUI element you are looking for and of some of its parents. If both this property and the **Window** property are configured, the **Selector** property is used at design time, and the **Window** property is used at runtime.
- **Window** - The window to attach to. This field accepts only Window variables. If both this property and the **Selector** property are configured, the **Selector** property is used at design time, and the **Window** property is used at runtime.

Options

- **SearchScope** - The application window in which to search for the UI element defined by the Selector property.

Common

- **DisplayName** - The display name of the activity.
- **TimeoutMS** - Specifies the amount of time (in milliseconds) to wait for the activity to run before an error is thrown. The default value is 30000 milliseconds (30 seconds).
- **ContinueOnError** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.

Output

- **ApplicationWindow** - The found active window. This field supports only Window variables. When a Window variable is specified, SearchScope and Selector properties are ignored.

Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

Practical 7B

Find different controls using UiPath

STEPS :

1. Open Studio and create a new **Process**.
2. Drag a **Sequence** container in the **Workflow Designer**.
   - Create the following variable:

Variable Name	Variable Type
EditElement	**UiPath.Core.UiElement**
FormatElement	**UiPath.Core.UiElement**

3. Drag an **OpenApplication** activity inside the **Sequence** container.

4. Drag a **Find Element** activity below the **OpenApplication** activity.
   - In the **Properties** panel, select the **COMPLETE** option from the **WaitForReady** drop-down list.
   - Add the variable EditElement in the **FoundElement** field.
5. Place a **Find Relative** activity below the **Find Element** activity.
   - In the **Properties** panel, add the value 20 in the **OffsetX** field.
   - Select the option **TopRight** from the **Position** drop-down list.
   - Add the variable EditElement in the **Element** field.
   - Add the variable FormatElement in the **RelativeElement** field.
6. Place a **Click** activity below the **Find Relative** activity.
   - In the **Properties** panel, add the variable FormatElement in the **Element** field.
   - Select the **None** option from the **KeyModifiers** drop-down list.
7. Place another **Click** activity below the first **Click** activity.
   - Inside the activity, click the **Indicate on screen** option. The GIF below shows all the steps you need to follow:

Drag an **Activate** activity below the **Click** activity.

- Inside the activity, click the **Indicate on screen** option. The GIF below shows all the steps you need to follow:

Place a **Set Focus** activity below the **Activate** activity.

- Inside the activity, click the **Indicate on screen** option, and select the **Size** menu option. The GIF below shows all the steps you need to follow:

4. Drag a **Send Hotkey** activity below the **Set Focus** activity.

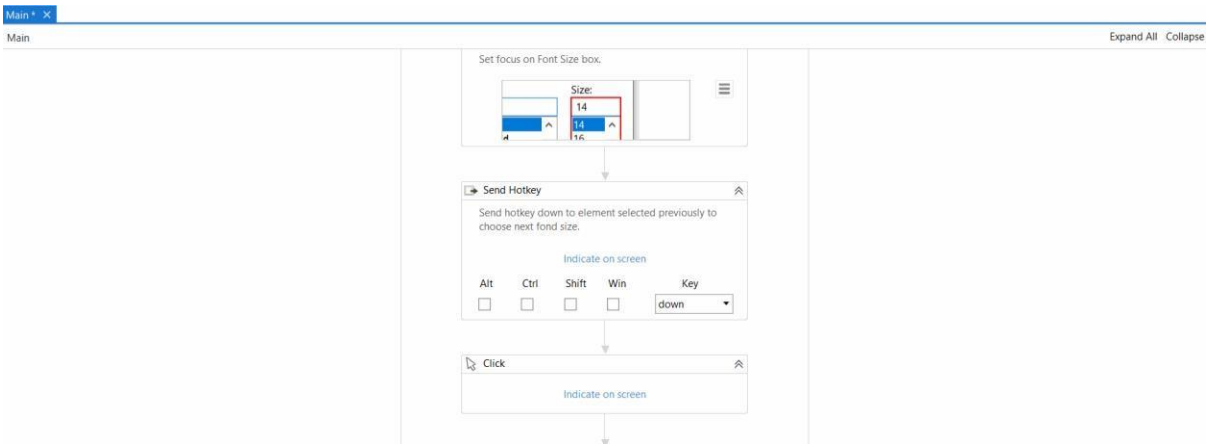
- In the Key field, type the value down.

- In the **Properties** panel, select the check box for the **Activate** option. This option brings the UI element to the foreground and activates it before the text is written.
- Select the **None** option from the **KeyModifiers** drop-down list.
- Select the check box for the **SpecialKey** option. This indicates that you are using a special key in the keyboard shortcut.

11. Place a **Click** activity below the **Send Hotkey** activity.

- Inside the activity, click the **Indicate on screen** option. The GIF below shows all the steps you need to follow:
    - Run the process. The automation opens a new Notepad file, navigates through the menu

OUTPUT :



Practical 7C

Demonstrate the following activities in UiPath:

i.MOUSE

UiPath Studio features activities that simulate any type of keyboard or mouse input that a human would use. Also, there are activities that can set focus to a certain window, minimize or maximize it, or perform any other kind of action on it. These activities are essential in creating an automation that simulates human behaviour. As explained here, there are several technologies that can be used for these activities, each with their own advantages in certain situations.

Double Click, **Click**, **Hover** are activities that simulate the clicking or hovering of UI elements. These activities are very useful in situations when human behavior must be mimicked. As input, these activities receive a Target, which can be either a Region variable, a UIElement variable or a selector, that helps you identify what you want to automate and where the actions must be performed. The target can also be automatically generated by using the **Indicate on Screen** functionality, which tries to identify UI elements in the indicated region, and generates selectors for them. If this does not work for you, then manual intervention might be required.

ii.Type Into

Type Into sends keystrokes to a UI element. Special keys are supported and can be selected from the drop-down list. This is a basic text input activity that is widely used in automations and is also generated by the automatic recording wizards. As input, this activity receives a string or a string variable that contains the text to be written, and a Target, which can be either a Region variable, a UIElement variable or a selector, that helps you identify what you want to automate and where the actions must be performed. The target can also be automatically generated by using the **Indicate on Screen** functionality, which tries to identify UI elements in the indicated region, and generates selectors for them. If this does not work for you, then manual intervention might be required.

iii. Type Secure text

Type Secure Text sends a secure string to a UI element. As input, this activity receives a SecureString variable that contains the text to be written, and a Target, which can be either a Region variable, a UIElement variable or a selector, that helps you identify what you want to automate and where the actions must be performed. The target can also be automatically generated by using the **Indicate on Screen** functionality, which tries to identify UI elements in the indicated region, and generates selectors for them. If this does not work for you, then manual intervention might be required. This activity is useful for secure automations, as it can use passwords that are stored in SecureString variables. Usually, the SecureString variable is supplied by a **Get Secure Credential** activity.

Practical 8A

Demonstrate the following events in UiPath:

i. **Element triggering event**
ii. **Image triggering event**
iii. **System Triggering Event**

STEPS :

1. Open Studio and create a new **Process** named by default **Main**.
2. Drag a **Sequence** container in the **Workflow Designer**.
3. Create the following variable:

Variable Name	Variable Type
ContinueMonitor	**Boolean**

4. Drag a **Log Message** activity inside the **Sequence** container.
   - In the **Properties** panel, select the **Level** option from the **Message** drop-down list.
   - Add the expression "Start monitoring..." in the **Message** field.
5. Add an **Assign** activity under the **Log Message** activity.
   - In the **Properties** panel, add the variable ContinueMonitor in the **To** field.
   - Add the condition True in the **Value** field.
6. Place a **Monitor Events** activity below the **Assign** activity.
   - In the **Properties** panel, add the value ContinueMonitor in the **RepeatForever** field.
7. Drag a **Hotkey Trigger** activity inside the **Monitor Events** activity. This activity opens the **Calculator** app from **Windows**.
   - Select the checkboxes for the Alt and Shift options.
   - In the Key field, type the letter c.
   - In the **Properties** panel, select the option **EVENT_BLOCK** from the **EventMode** drop-down list.
8. Drag another **Hotkey Trigger** activity and place it next to the previous **Hotkey Trigger** activity. This activity opens a new browser tab and searches on Google the text previously selected by the user.
   - Select the checkboxes for the **Alt** and **Shift** options.
   - In the **Key** field, type the letter g.
   - In the **Properties** panel, select the option **EVENT_BLOCK** from the **EventMode** drop-down list.
9. Drag another **Hotkey Trigger** activity and place it next to the previous **Hotkey Trigger** activity. This activity stops monitoring the events.
   - Select the check boxes for the **Alt** and **Shift** options.
   - In the **Key** field, type the letter s.
   - In the **Properties** panel, select the option **EVENT_BLOCK** from the **EventMode** drop-down list.
10. Add a new **Sequence** container and place it below the **Hotkey Trigger** activity.

- In the **Properties** panel, add the name Event Handler in the **DisplayName** field.
- Create the following variable:

Variable Name	Variable Type
TriggerHotkey	**UiPath.Core.EventInfo**
ContinueMonitor	**Boolean**

11. Drag a **Log Message** activity inside the **Event Handler**.

- In the **Properties** panel, select the **Info** option from the **Level** drop-down list.
- Add the expression "Event triggered" in the **Message** field.

12. Drag a **Get Event Info** activity below the **Log Message** activity.

- In the **Properties** panel, add the variable TriggerHotkey in the **Result** field.
- Select the **UiPath.Core.EventInfo** option from the **TypeArgument** drop-down list.

13. Place a **Switch** activity below the **Get Event Info** activity. All **Hotkey Triggers** are described inside this activity and treated as cases.

- In the **Properties** panel, add the value TriggerHotkey.KeyEventInfo.KeyName.ToLower in the **Expression** field.
- Select the **String** option from the **TypeArgument** drop-down list.

14. Click the **Add new case** button from the **Switch** activity.

- Add the value c in the **Case value** field.

15. Place an **Open Application** activity and place it inside the **Case c** container. This represents the first **Hotkey Trigger** case that opens the **Calculator** app.

- In the **Properties** panel, add the expression "calc.exe" in the **Arguments** field.
- Add the expression "<wnd app='applicationframehost.exe' title='Calculator' />" in the **Selector** field.

16. Click the **Add new case** button from the **Switch** activity.

- Add the value g in the **Case value** field.

17. Drag a **Sequence** container and place it inside the **Case g** container. This represents the second **Hotkey Trigger** case that initiates a Google search for the previously selected text.

- In the **Properties** panel, add the name Google selected text in the **DisplayName** field.
- Create the following variable:

Variable Name	Variable Type
TextToSearch	**GenericValue**

18. Drag a **Delay** activity and place it inside the **Google selected text** sequence.

- In the **Properties** panel, add the value 00:00:00.5000000 in the **Duration** field.

19. Add a **Copy Selected Text** activity below the **Delay** activity.

- In the **Properties** panel, add the value True in the **ContinueOnError** field.
- Add the variable TextToSearch in the **Result** field.
- Add the value 2000 in the **Timeout (milliseconds)** field.

20. Drag an **If** activity under the **Copy Selected Text** activity.

- In the **Properties** panel, add the expression TextToSearch IsNot Nothing in the **Condition** field.

21. Place an **Open Browser** activity inside the **Then** box.

- In the **Properties** panel, select the **IE** option from the **BrowserType** drop-down list.
- Add the expression "www.google.com" in the **Url** field.
- Select the checkbox for the **NewSession** option. This starts a new session in the selected browser.

22. Place a **Type Into** activity inside the **Do** sequence.

- In the **Properties** panel, select the **Target** option from the **Target** drop-down list.
- Add the expression "<webctrl tag='INPUT' aaname='Search' />" in the **Selector** field.
- Select the **INTERACTIVE** option from the **WaitForReady** drop-down list.
- Add the variable TextToSearch in the **Text** field.
- Select the checkbox for the **Activate** option. This option brings the UI element to the foreground and activates it before the text is written.
- Select the checkbox for the **SimulateType** option. This option simulates the type using the technology of the target application.

23. Drag a **Send Hotkey** application below the **Type Into** activity.

- In the **Properties** panel, add the expression "enter" in the **Key** field.
- Select the **Target** option from the **Target** drop-down list.
- Add the expression "<webctrl tag='INPUT' aaname='Search' />" in the **Selector** field.
- Select the **INTERACTIVE** option from the **WaitForReady** drop-down list.
- Select the checkbox for the **Activate** option. This option brings the UI element to the foreground and activates it before the text is written.
- Select the **None** option from the **KeyModifiers** drop-down list.
- Select the checkbox for the **SpecialKey** option. This option indicates if the use of a special key in the keyboard shorcut.

24. Drag a **Message Box** activity in the **Else** container.

- In the **Properties** panel, select the **Ok** button from the **Buttons** drop-down list.
- Add the expression "Text could not be copied. Please try again." in the **Text** field.
- Select the checkbox for the **TopMost** option. This option always brings the message box to the foreground.

25. Click the **Add new case** button from the **Switch** activity.

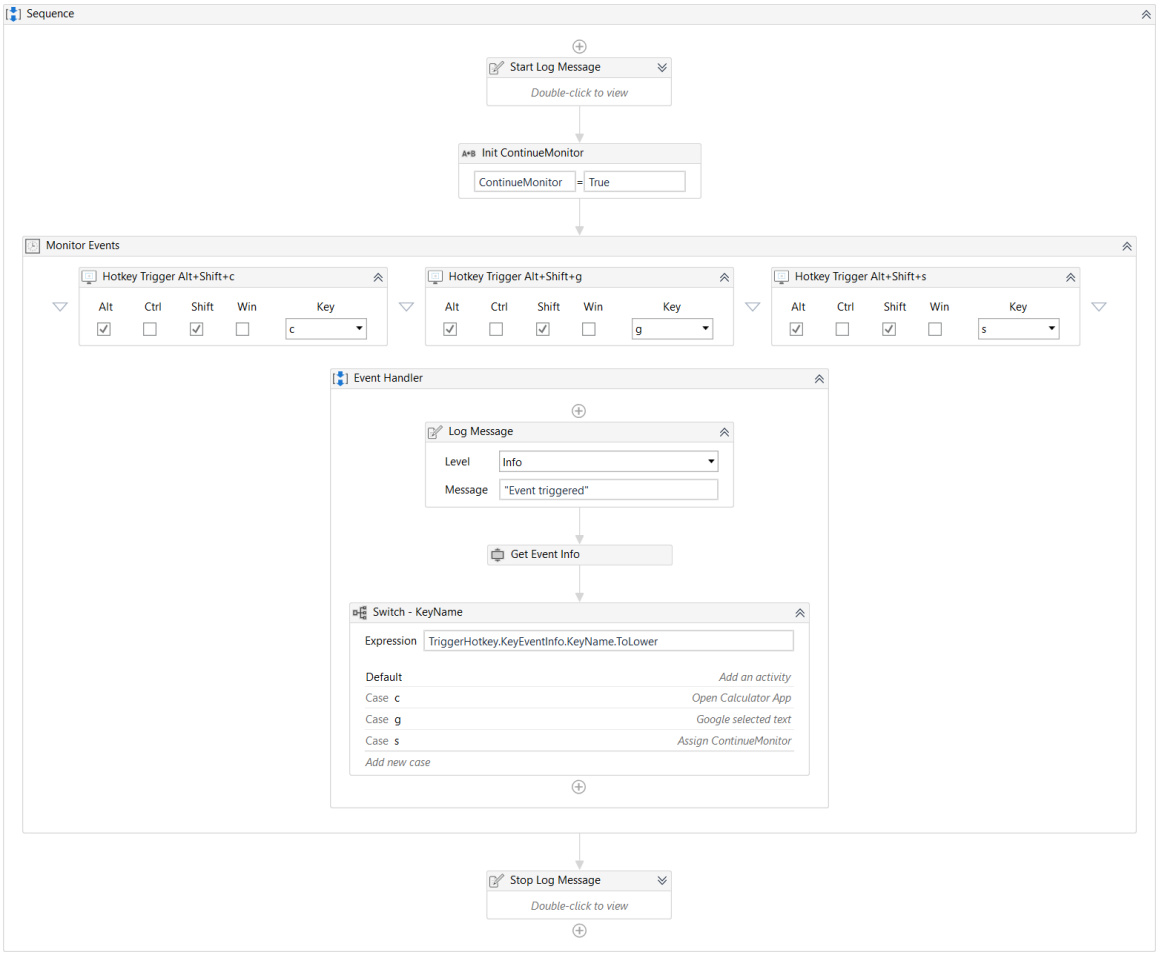
- Add the value s in the **Case value** field.

26. **Assign** activity inside the **Case s** container. This represents the third **HotkeyTrigger** case that stops monitoring the events.

- Add the variable ContinueMonitor in the **To** field.
- Add the condition False in the **Value** field

27. Place a **Log Message** activity below the **Monitor Events** activity.

- In the **Properties** panel, select the **Info** option from the **Level** drop-down field.
- Add the expression "Stop monitoring.." in the **Message**



Practical 8B

Automate the following screen scraping methods using UiPath

i. **Full Test**
ii. **Native**
iii. **OCR**

STEPS :

Output or screen scraping methods refer to those activities that enable you to extract data from a specified UI element or document, such as a .pdf file.

To understand which one is better for automating your business process, let's see the differences between them.

Capability Method	Speed	Accuracy	Background Execution	Extract Text Position
FullText	10/10	100%	Yes	no
Native	8/10	100%	No	yes
OCR	3/10	98%	No	yes

FullText is the default method, it is fast and accurate, yet unlike the **Native** method, it cannot extract the screen coordinates of the text.

Both these methods work only with desktop applications, but the **Native** method only works with apps that are built to render text with the Graphics Device Interface (GDI).

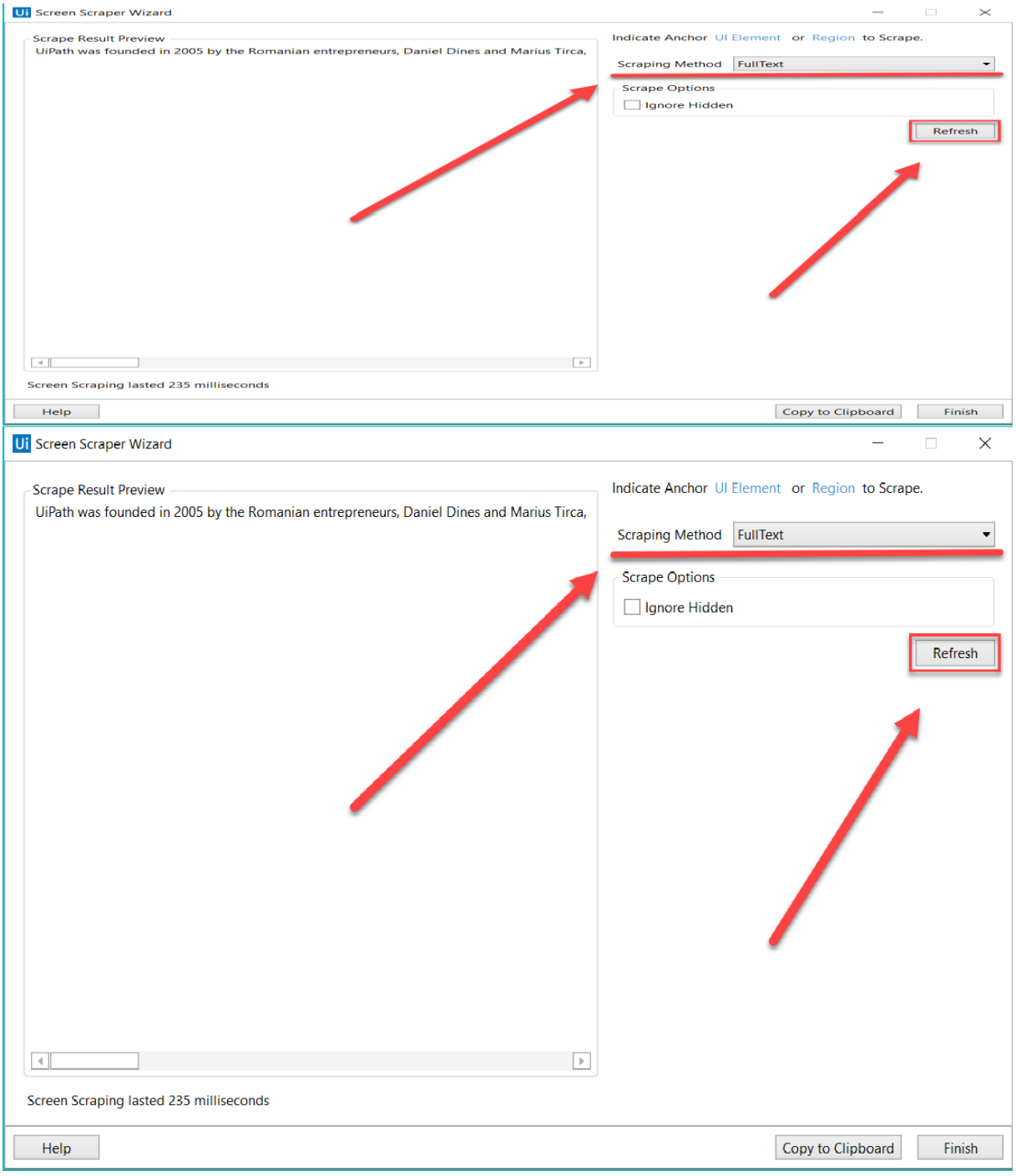
OCR is not 100% accurate but can be useful to extract text that the other two methods could not, as it works with all applications including Citrix. Studio uses two OCR engines, by default: Google Tesseract and Microsoft Modi.

Languages can be changed for OCR engines and you can find out how to **Install OCR Languages** here.

Capability Method	Multiple Languages Support	Preferred Area Size	Support for Color Inversion	Set Expected Text Format
Google Tesseract	Can be added	Small	yes	yes
Microsoft MODI	Supported by default	Large	no	no

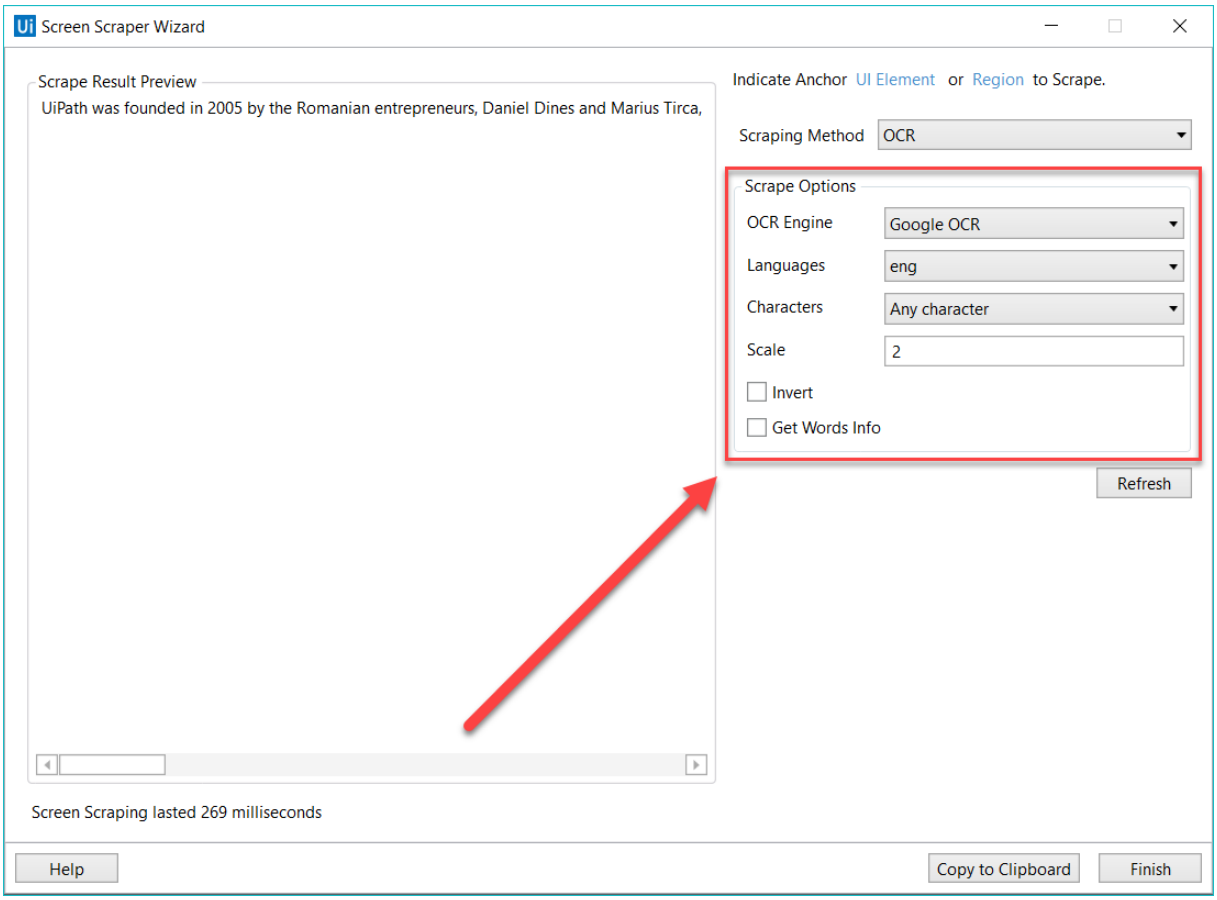
To start extracting text from various sources, click the **Screen Scraping** button, in the **Wizards** group, on the **Design** ribbon tab.

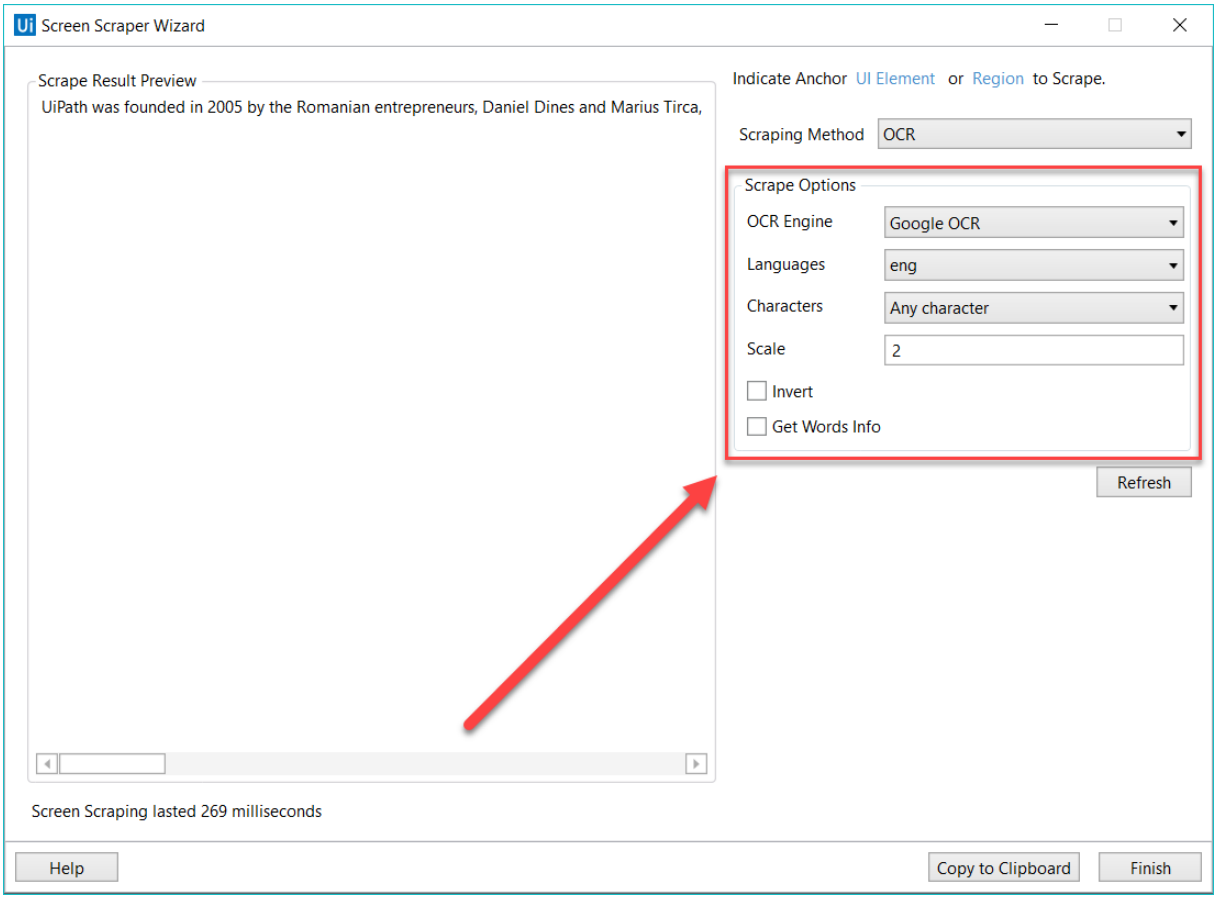
The screen scraping wizard enables you to point at a UI element and extract text from it, using one of the three output methods described above. Studio automatically choses a screen scraping method for you, and displays it at the top of the **Screen Scraper Wizard** window.



To change the method of screen scraping, select another one from the **Options** panel and then click **Refresh**.

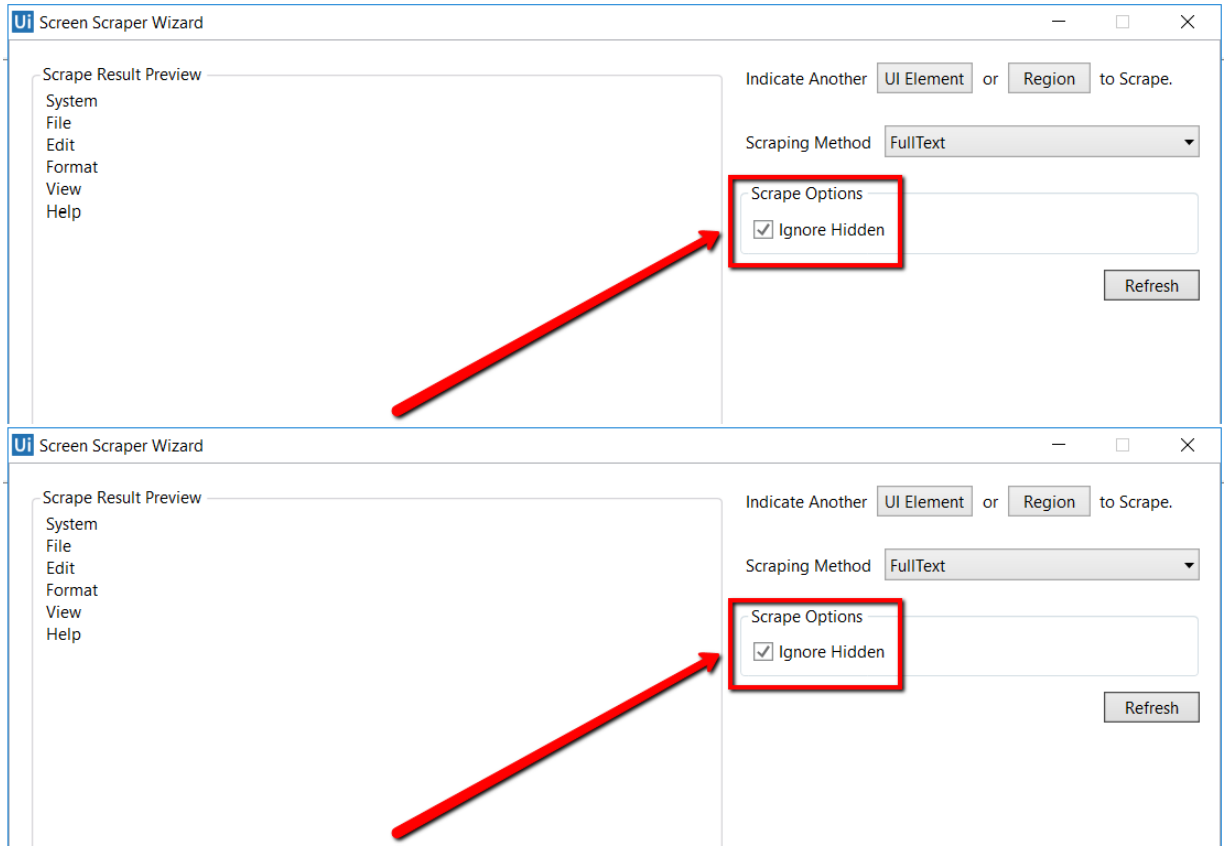
When you are satisfied with the scraping results, click **Copy to Clipboard** and then **Finish**. The latter option copies the extracted text to the Clipboard, and it can be added to a **Generate Data Table** activity in the **Designer** panel. Just like **desktop recording**, screen scraping generates a container (with the selector of the top-level window) which contains activities, and partial selectors for each activity.


Screen Scraper Wizard
Scrape Result Preview
UiPath was founded in 2005 by the Romanian entrepreneurs, Daniel Dines and Marius Tirca,
Indicate Anchor UI Element or Region to Scrape.
Scraping Method
OCR
Scrape Options
OCR Engine
Google OCR
Languages
eng
Characters
Any character
Scale
2
Invert
Get Words Info
Refresh
Screen Scraping lasted 269 milliseconds
Help
Copy to Clipboard
Finish


Screen Scraper Wizard
Scrape Result Preview
UiPath was founded in 2005 by the Romanian entrepreneurs, Daniel Dines and Marius Tirca,
Indicate Anchor UI Element or Region to Scrape.
Scraping Method
OCR
Scrape Options
OCR Engine
Google OCR
Languages
eng
Characters
Any character
Scale
2
Invert
Get Words Info
Refresh
Screen Scraping lasted 269 milliseconds
Help
Copy to Clipboard
Finish

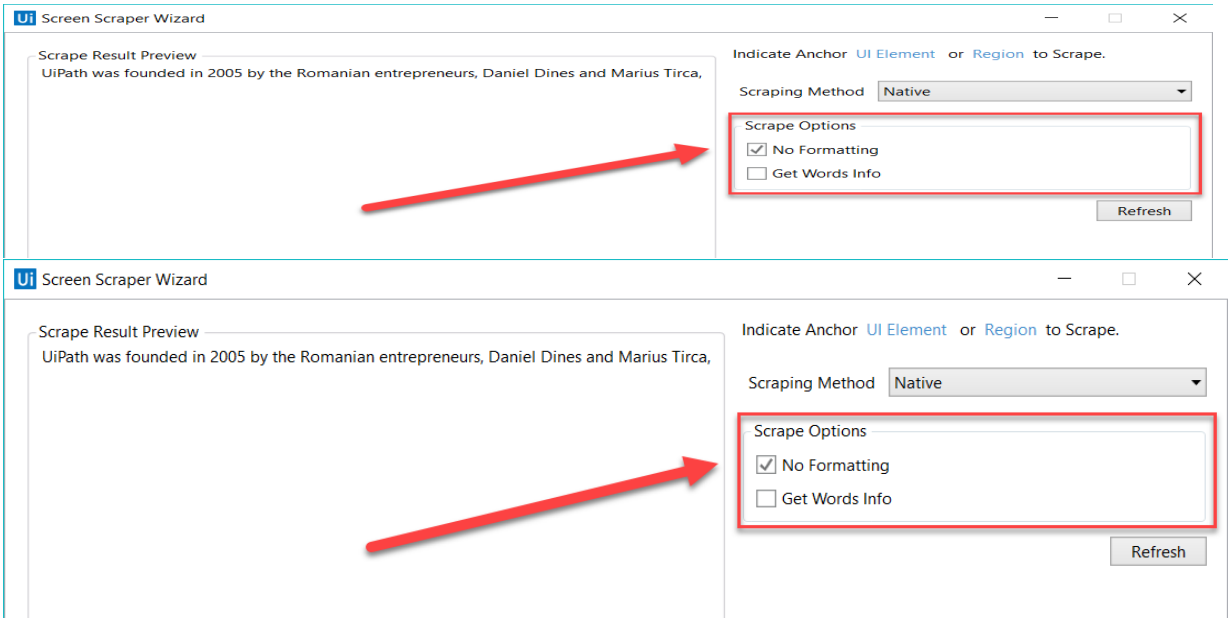
Each type of screen scraping comes with different features in the **Screen Scraper Wizard**, in the **Options** panel:

1. FullText



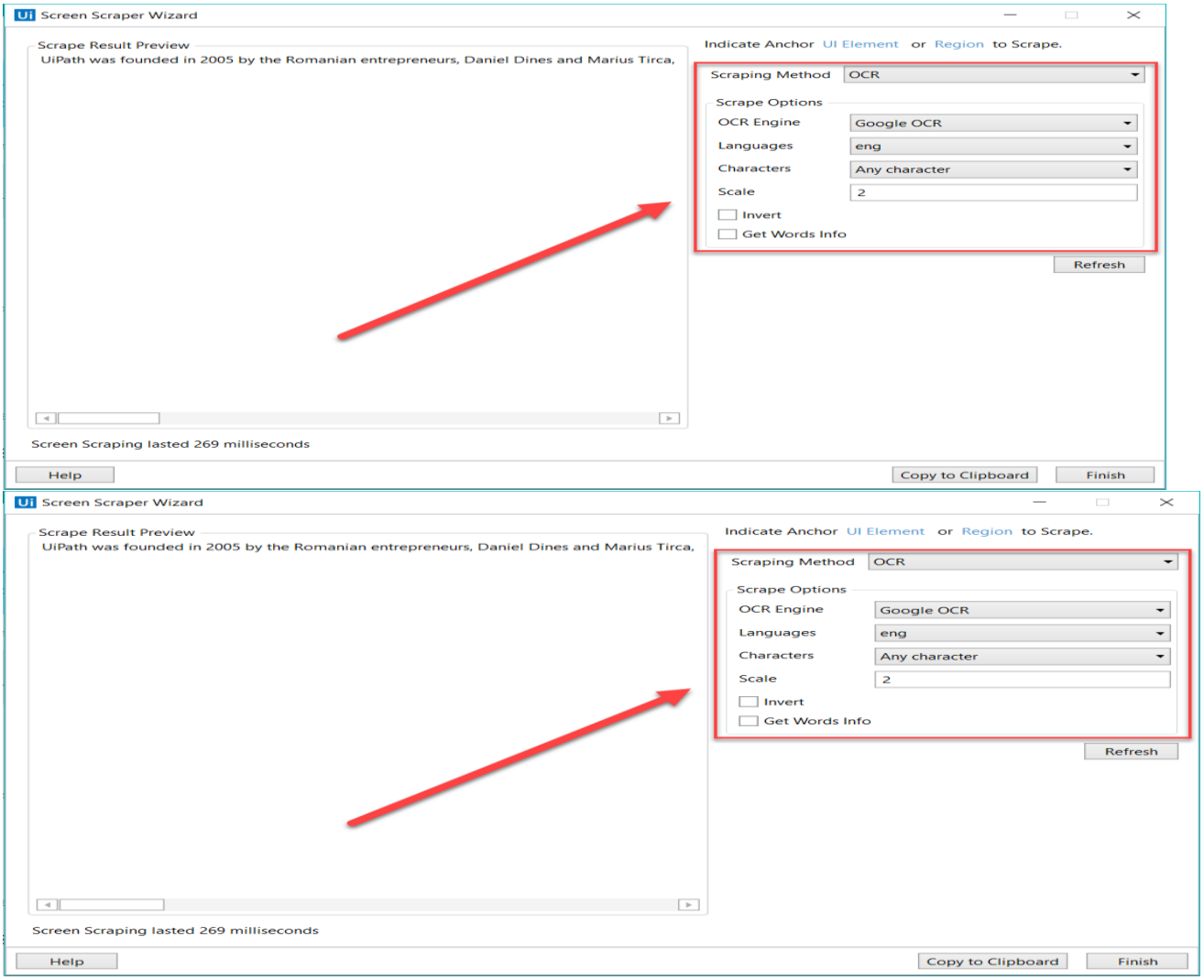
- **Ignore Hidden** – when this checkbox is selected, the hidden text from the selected UI element is not copied.

2. Native



- **No Formatting** – when this checkbox is selected, the copied text does not extract formatting information from the text. Otherwise, the extracted text's relative position is retained.
- **Get Words Info** – when this checkbox is selected, Studio also extracts the screen coordinates of each word. Additionally, the Custom Separators field is displayed, which enables you to specify the characters used as separators. If the field is empty, all known text separators are used.

3. Google OCR

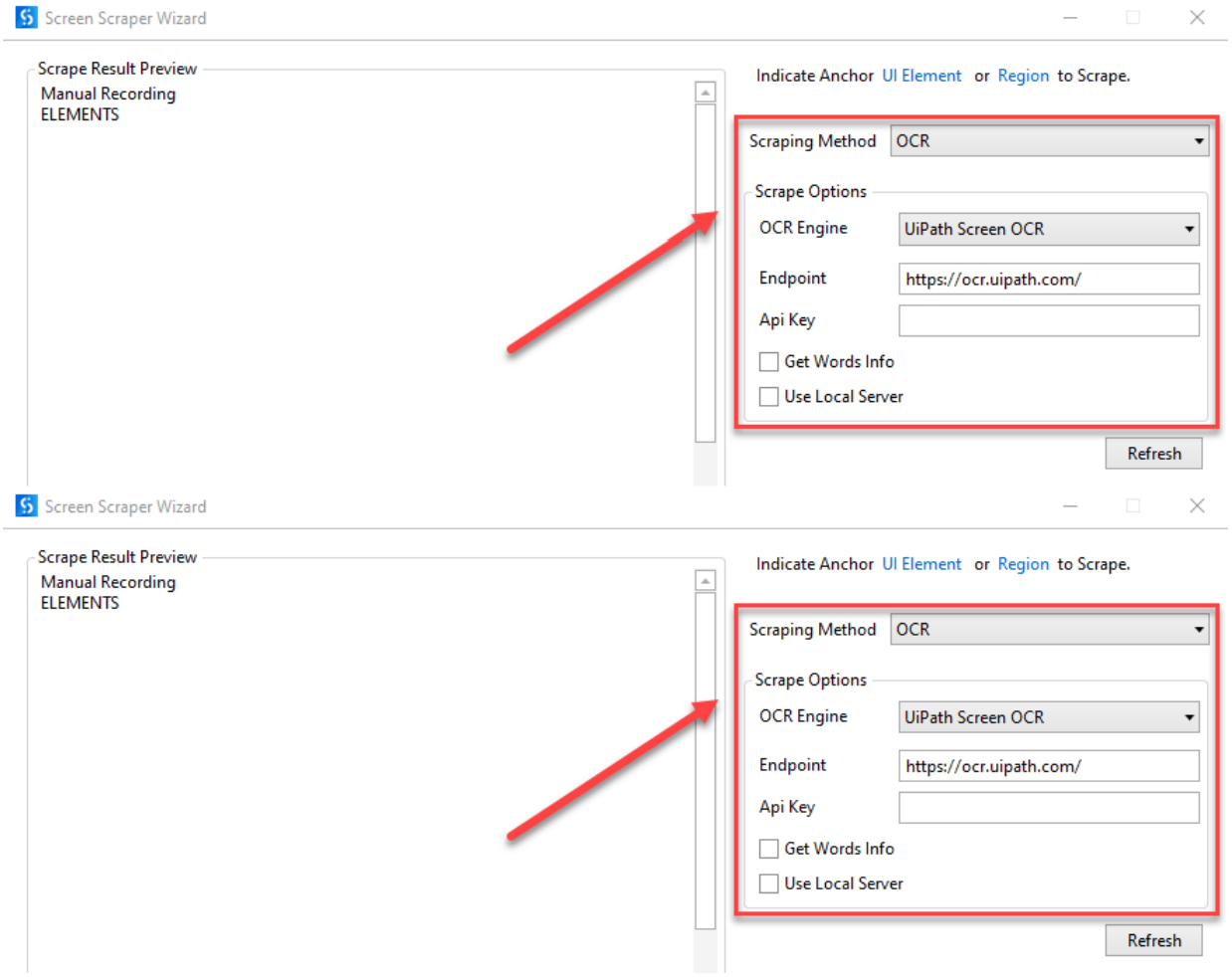


- **Languages** – only English is available by default.
- **Characters** – enables you to select which types of characters to be extracted. The following options are available: **Any character**, **Numbers only**, **Letters**, **Uppercase**, **Lowercase**, **Phone numbers**, **Currency**, **Date** and **Custom**. If you select **Custom**, two additional fields, **Allowed** and **Denied**, are displayed that enable you to create custom rules on which types of characters to scrape and which to avoid.
- **Invert** – when this checkbox is selected, the colors of the UI element are inverted before scraping. This is useful when the background is darker than the text color.
- **Scale** – the scaling factor of the selected UI element or image. The higher the number is, the more you enlarge the image. This can provide a better OCR read and it is recommended with small images.
- **Get Words Info** – gets the on-screen position of each scraped word.

Note:

In some instances of UiPath Studio, the Google Tesseract engine may have training files (about training files: Wikipedia, GitHub) that do not work for certain non-English languages. Running a project with these corrupted training files may lead to an exception being thrown. To fix this issue, download the training file for the language you wish to use from here and copy it into the tessdata folder from the UiPath installation directory. To check if the training files you downloaded work, you can download this test project.

4. UiPath Screen OCR

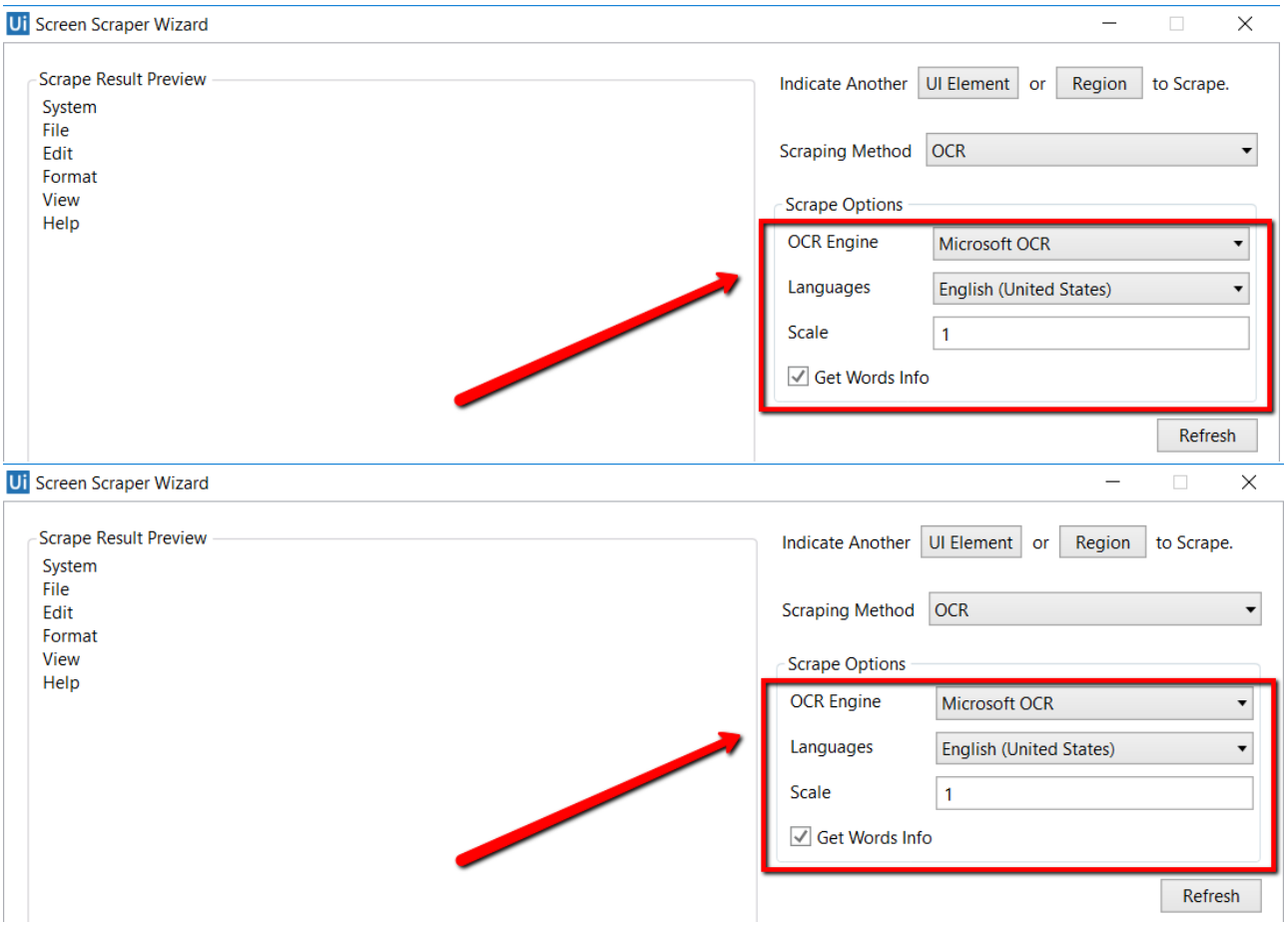


- **Endpoint** – the endpoint where the OCR model is hosted, either publicly or through an ML Skill in AI Center.
- **API Key** – the endpoint API key.
- **Get Words Info** – gets the on-screen position of each scraped word.
- **Use Local Server** – select this option if you want to run the OCR locally (requires Computer Vision Local Server Pack)

5. Microsoft OCR

Important

Microsoft OCR scraping engine does not support .NET 5 workflows.



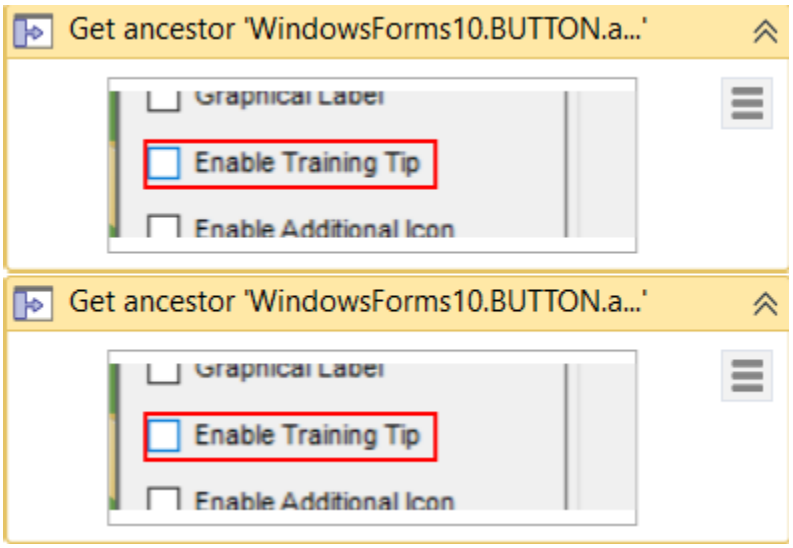
- **Languages** – enables you to change the language of the scraped text. By default, English is selected.
- **Scale** – the scaling factor of the selected UI element or image. The higher the number is, the more you enlarge the image. This can provide a better OCR read and it is recommended with small images.
- **Get Words Info** - gets the on-screen position of each scraped word.

Besides getting text out of an indicated UI element, you can also extract the value of multiple types of attributes, its exact screen position, and its ancestor.

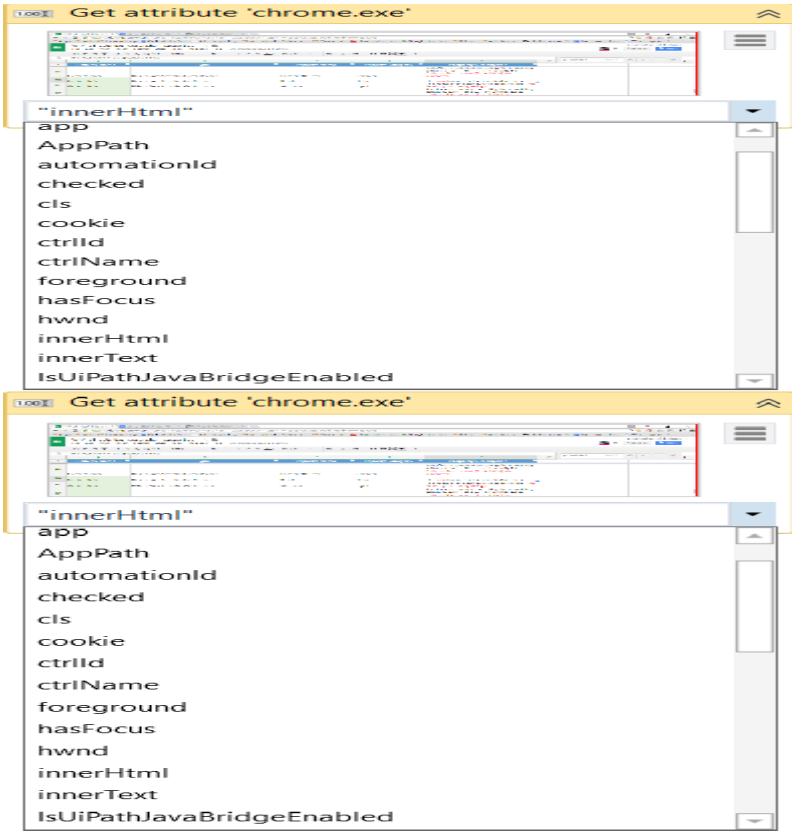
This type of information can be extracted through dedicated activities that are found in the **Activities** panel, under **UI Automation > Element > Find** and **UI Automation > Element > Attribute**.

These activities are:

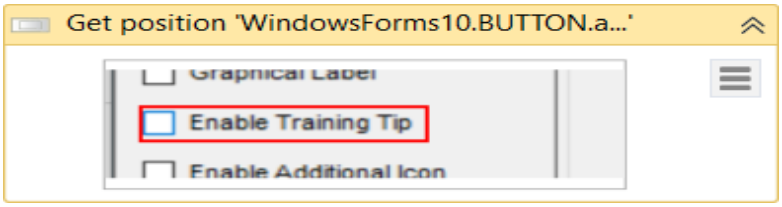
- **Get Ancestor** – enables you to retrieve an ancestor from a specified UI element. You can indicate at which level of the UI hierarchy to find the ancestor, and store the results in a UiElement variable.



- **Get Attribute** – retrieves the value of a specified UI element attribute. Once you indicate the UI element on screen, a drop-down list with all available attributes is displayed.



- **Get Position** – retrieves the bounding rectangle of the specified UiElement, and supports only Rectangle variables.



Practical 8C

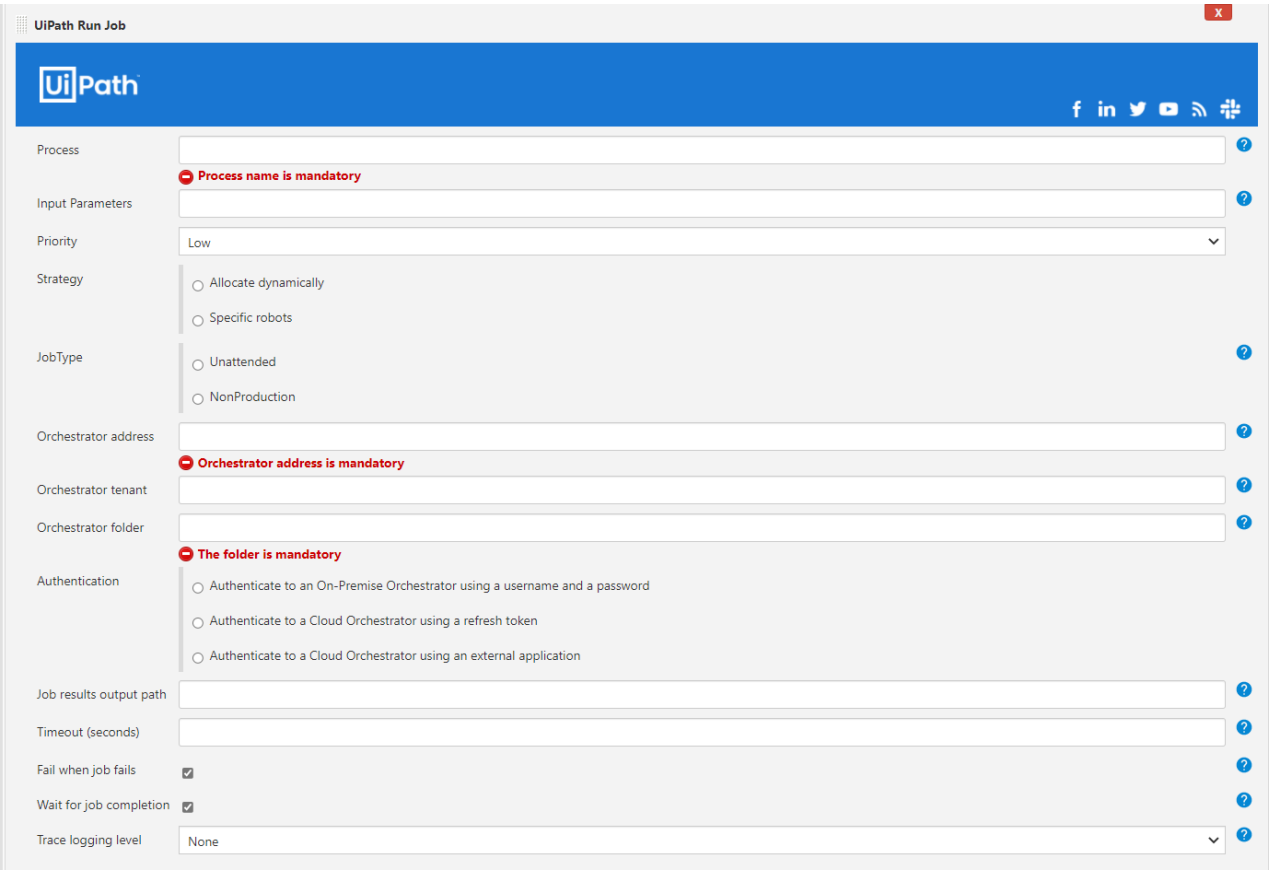
Install and automate any process using UiPath with the following plug-ins:

i. **Java Plugin**
ii. **Mail Plugin**
iii. **PDF Plugin**
iv. **Web Integration**
v. **Excel Plugin**
vi. **Word Plugin**
vii. **Credential Management**

STEPS :

UiPath Run Job

The UiPath Run Job post-build step starts an already deployed process on an Orchestrator instance. The processes this task refers to are found in Automations->Processes on newer versions of Orchestrator and directly on the Processes tab on older versions of Orchestrator.



⚙ **Configuration**

Argument	Description
Process	(Required) Process name. You can take the process name from the Orchestrator UI. If the process is deployed in a Modern Folder then this argument should be the NAME of the process in the Processes tab. If the process is deployed in a Classic Folder, then the argument must be formed by the NAME of the process and the ENVIRONMENT (eg: NAME: ProcessA ENVIRONMENT: Production ProcessName: ProcessA_Production).
Parameters	The full path to a json input file. This is used when the Process requires input.
Priority	The job run priority.
Strategy	Specify the job run strategy, dynamically allocated job(s) or robot specific job(s). Options: Allocate dynamically, Specific robots
Job Type	This feature is available only on modern folders! Choose the license model of the runtime under which the job is to be executed.
Orchestrator address	The address of the Orchestrator instance where we'll run the process.
Orchestrator tenant	Specify the Orchestrator tenant.
Orchestrator folder	Specify the folder where the specified process was deployed.
Authentication	For authentication towards Orchestrator, credentials have to be created in Jenkins upfront. There are 3 options to authenticate: *(1)* Authenticate to an On-Premise Orchestrator using username and password *(2)* Authenticate to a Cloud Orchestrator using a refresh token (API key). The account name and API key are accessible via Services->API Access (see below for a detailed explanation on how to retrieve this) *(3)* Authenticate to a Cloud Orchestrator using external app authentication.
Job results output path	Specify the output full path of the job results, e.g. testResults.json. The results are outputted in json format. If not specified, the results are outputted to the artifact staging directory as UiPathResults.json. The output is in json format.
Timeout	Specify the job run(s) timeout in seconds.
Fail when job fails	The task fails when at least one job fails. (default true)
Wait for job completion	Wait for job run(s) completion. (default true)
Trace logging level	Setting used to enable the trace logging to one of the following level: None, Critical, Error, Warning, Information, Verbose. (default None). Useful for debugging purposes.
	Parameters used for strategy Allocate dynamically

Argument	Description
No. of jobs	The number of job runs. (default 1)
User	The name of the user. This feature is available only on modern folders! This should be a machine user, not an orchestrator user. For local users, the format should be MachineName\UserName
Machine	The name of the machine. This feature is available only on modern folders!
	Parameters used for strategy Specific robots
Robot names	Comma-separated list of specific robot names.

☎ Pipeline example:

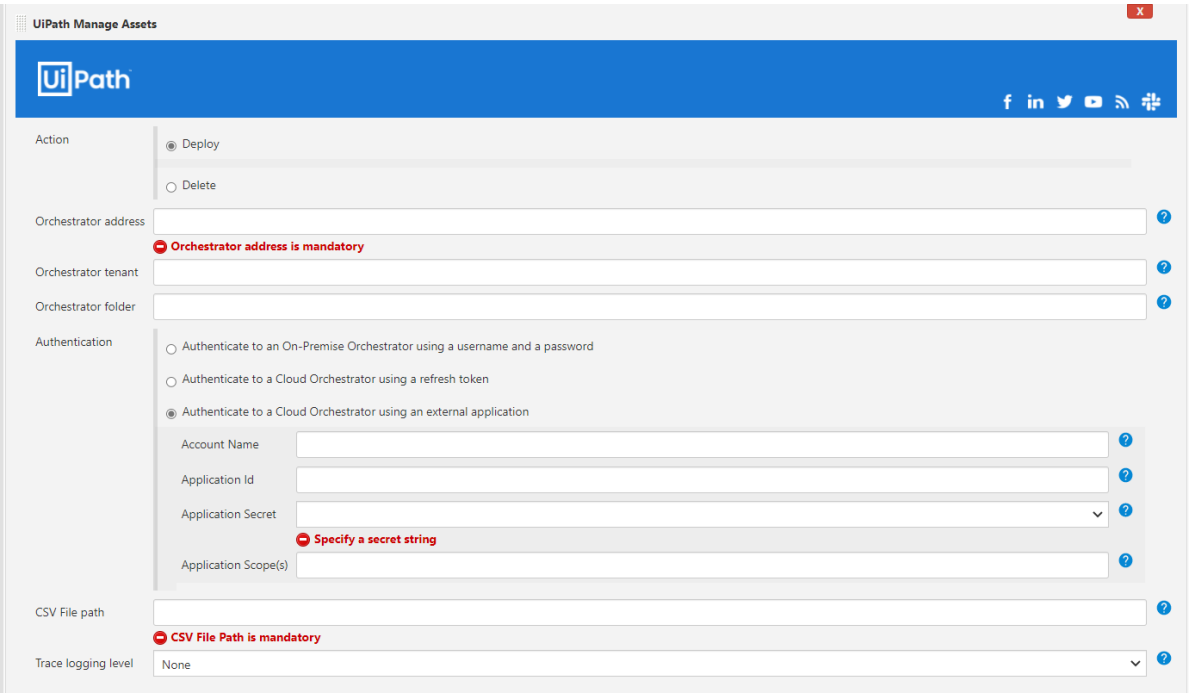
```
pipeline  {
  agent any
  environment {
      MAJOR = '1'
      MINOR = '0'
  }
  stages {
    stage ('Build') {
        UiPathRunJob(
          credentials: UserPass('825c83c9-9a14-44eb-883a-af54f8078af0'),
          failWhenJobFails: true,
          folderName: 'A_Classic',
          orchestratorAddress: 'https://testorchestrator.some-domain.com',
          orchestratorTenant: 'Default',
          parametersFilePath: '',
          priority: 'Low',
          processName: 'ProcessA_EnvB',
          resultFilePath: 'output.json',
          strategy: Dynamically(jobsCount: 1, machine: 'TestMachine', user: 'TestUser'), timeout: 3600,
waitForJobCompletion: true, traceLoggingLevel: 'None'
        )
        UiPathRunJob(
          credentials: UserPass('825c83c9-9a14-44eb-883a-af54f8078af0'),
          failWhenJobFails: true,
          folderName: 'A_Classic',
          orchestratorAddress: 'https://testorchestrator.some-domain.com',
          orchestratorTenant: 'Default',
          parametersFilePath: '',
          priority: 'Low',
          processName: 'ProcessA_EnvB',
          resultFilePath: 'output.json',
          strategy: Robot('robot1,robot2'),
          timeout: 1800,
          waitForJobCompletion: false,
          traceLoggingLevel: 'None'
        )
    }
  }
}
```

💼 UiPath Manage Assets

The UiPathManageAssets step enables you to deploy, update or delete assets on an Orchestrator instance. In order to deploy assets you must describe them in a CSV file like the one in the example below encoded in uft-8.

```
name,type,value,description
asset_1_name,text,asset_value,this is an test description # we can have comments
asset_2_name,integer,123
asset_3_name,boolean,false
asset_4_name,credential,"username::password"
```

There are 4 types of assets text, integer, boolean and credential. For the credential you must encode the username and password by using :: to separte the two fields.



⚙ Configuration

Argument	Description
Action	What to do with the provided assets: deploy or delete. If a deployed asset exists then it will be updated instead.
Orchestrator address	The address of the Orchestrator instance where we'll deploy or update assets.
Orchestrator tenant	Specify the Orchestrator tenant onto which the assets will be deployed or updated.
Orchestrator folder	Specify the folder where assets will be deployed or updated.
Authentication	For authentication towards Orchestrator, credentials have to be created in Jenkins upfront. There are 3 options to authenticate: *(1)* Authenticate to an On-Premise

Argument	Description
	Orchestrator using username and password *(2)* Authenticate to a Cloud Orchestrator using a refresh token (API key). The account name and API key are accessible via Services->API Access (see below for a detailed explanation on how to retrieve this) *(3)* Authenticate to a Cloud Orchestrator using external app authentication.
CSV File Path	The path to the csv file containing the asset details. The same file can be used to deploy or update the assets although the type isn't required for update. The type field can also be empty but the column must be present. For delete, only the name column is used, so the other columns can be empty but they must be present. You can set an optional description for each asset (e.g. type, value, description). Make sure to remove any line breaks as each line is interpreted as a new asset.
Trace logging level	Setting used to enable the trace logging to one of the following level: None, Critical, Error, Warning, Information, Verbose. (default None). Useful for debugging purposes.

☎ Pipeline example:

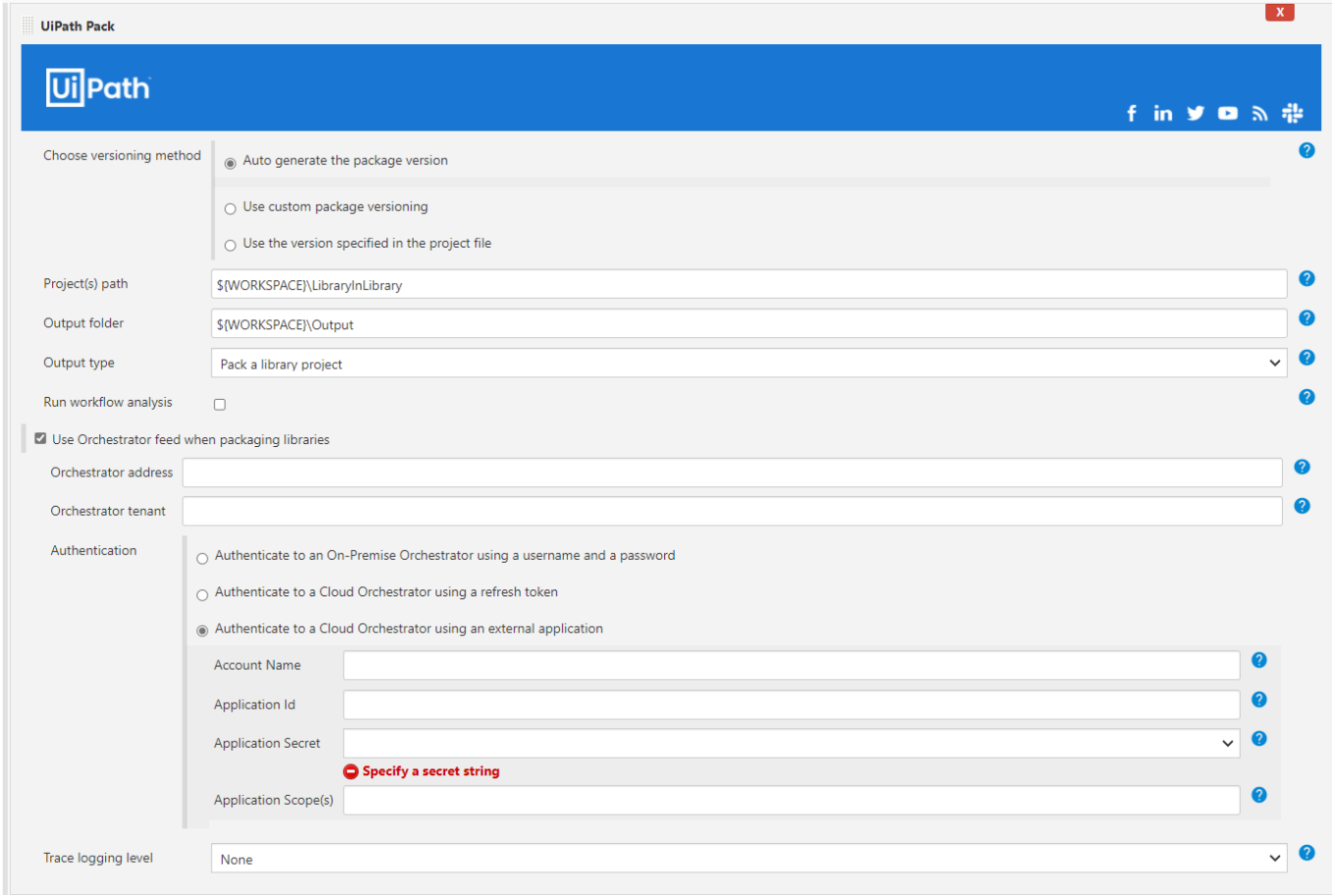
```
pipeline  {
  agent any
  environment {
     MAJOR = '1'
     MINOR = '0'
  }
  stages {
    stage ('Build') {
       UiPathAssets (
          assetsAction: DeployAssets(),
          credentials: Token(accountName: '', credentialsId: ''),
          filePath: '${WORKSPACE}/test.csv',
          folderName: 'Default',
          orchestratorAddress: 'https://test-orchestrator.somedomain.com',
          orchestratorTenant: 'Default',
          traceLoggingLevel: 'None'
       )
       UiPathAssets(
          assetsAction: DeleteAssets(),
          credentials: UserPass('825c83c9-9a14-44eb-883a-af54f8078af0'),
          filePath: '${WORKSPACE}/test.csv',
          folderName: 'Default',
          orchestratorAddress: 'https://test-orchestrator.somedomain.com',
          orchestratorTenant: 'Default',
          traceLoggingLevel: 'None'
       )
    }
  }
}
```

🎁 UiPath Pack

Application: RPA

Type: Build task

UiPath Pack is available in standard jobs and pipelines, and lets you package an existing UiPath project into a NuGet package.



Configuration

Job parameter	Description
Choose versioning method	UiPath packages are versioned. With UiPath pack you can choose between 3 different options: (1) Auto generate (2) Define custom version (3) Use the current version set in the project.
Project(s) path	The location of the project(s) to be packaged. It can be a direct path to a project.json file or a directory with one or multiple projects. In the latter case, each level one project is packaged individually.
Output folder	Path to a folder, where the created package should be placed.
Output type	The output type of the project(s). There are 5 options for the project(s) type: *(1)* Output type of the project *(2)* Pack a process project *(3)* Pack a library project *(4)* Pack a tests project *(5)* Pack an objects project.

Job parameter	Description
Run workflow analysis	Run the workflow analysis before packing, checking the project via predefined rules for violations. Fail the job in case of errors. By default the analysis is not run.
Trace logging level	Setting used to enable the trace logging to one of the following level: None, Critical, Error, Warning, Information, Verbose. (default None). Useful for debugging purposes.
Use orchestrator	Use Orchestrator feed when packaging libraries. The Orchestrator must be 20.4 or higher. The library feed needs to allow API Key authentication in Tenant -> Setting -> Deployment.
Orchestrator address	The address of the Orchestrator instance from which library dependencies should be restored.
Orchestrator tenant	The Orchestrator tenant from which library dependencies should be restored.
Authentication	For authentication towards Orchestrator, credentials have to be created in Jenkins upfront. There are 3 options to authenticate: *(1)* Authenticate to an On-Premise Orchestrator using username and password *(2)* Authenticate to a Cloud Orchestrator using a refresh token (API key). The account name and API key are accessible via Services->API Access (see below for a detailed explanation on how to retrieve this) *(3)* Authenticate to a Cloud Orchestrator using external app authentication.

☎ Pipeline Example:

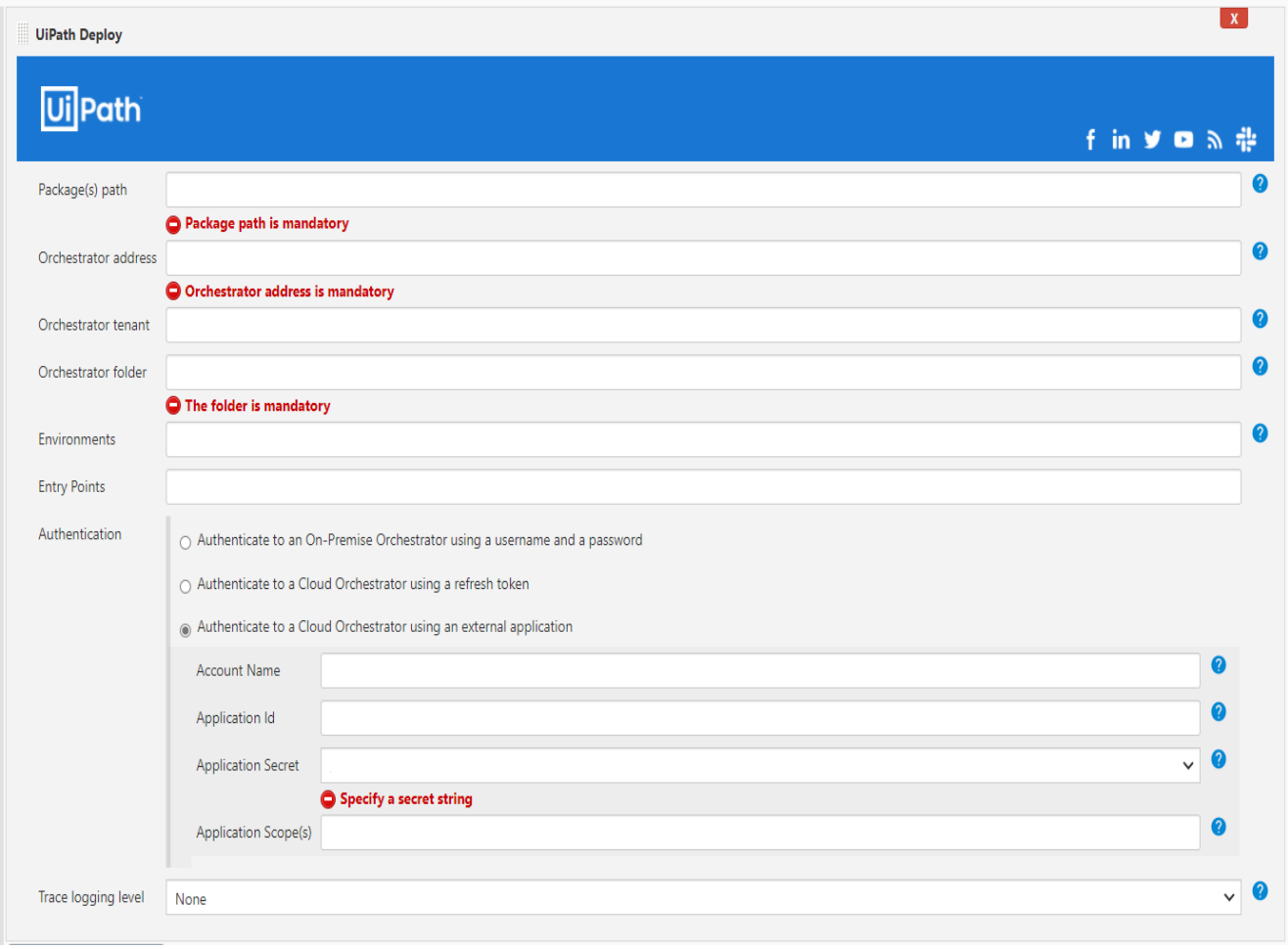
```
pipeline  {
  agent any
  environment {
     MAJOR = '1'
     MINOR = '0'
  }
  stages {
    stage ('Build') {
      steps {
      UiPathPack (
         outputPath: "Output\\${env.BUILD_NUMBER}",
         projectJsonPath: "UiBank\\project.json",
         version: [$class: 'ManualVersionEntry', version:
"${MAJOR}.${MINOR}.${env.BUILD_NUMBER}"]
         useOrchestrator: true,
         traceLoggingLevel: "None",
         orchestratorAddress: "OrchestratorUrl",
         orchestratorTenant: "tenant name",
         credentials: [$class: 'UserPassAuthenticationEntry', credentialsId: “credentialsId”]
       )
      }
    }
  }
}
```

🚀 UiPath Deploy

Application: RPA

Type: Post-Build task

UiPath Deploy is available in standard jobs and pipelines, and lets you deploy a UiPath NuGet package onto UiPath Orchestrator.



⚙ Configuration

Job parameter	Description
Package(s) path	The folder that holds your UiPath nuget package(s).
Orchestrator address	The address of the Orchestrator instance onto which the package(s) will be deployed.
Orchestrator tenant	The Orchestrator tenant onto which the package(s) will be deployed.
Orchestrator folder	The folder to deploy to. If the folder is a classic folder, you will also need to set the environments field. For modern folders, setting the environments is not required.

Job parameter	Description
Environments	The environment onto which the package will be deployed as a process. For the project and environment with existing processes, the processes will be updated to use the latest project version. Specify the environment onto which the package will be deployed as a process. For the project and environment with existing processes, the processes will be updated to use the latest project version. Required when using a classic folder, otherwise not applicable.
Entry Points	Specify entry points to create or update a process. The entry point specifies the filePath starting from the root of the project. Conditions: • Entry points are available for Orchestrator version 21.4 or higher (e.g. 21.4.UiPathDeploy.entryPoints). For Orchestrator versions lower than 21.4, you need to enter any value, as the field must not remain empty. • Default entry point set to Main.xaml. • For classic folders (deprecated) you can specify only one entry point for each environment • Mulitple Entrypoints can be specified as ' Main.xaml, EntryPoint2.xaml ' For more information, see Orchestrator Entry Points
Authentication	For authentication towards Orchestrator, credentials have to be created in Jenkins upfront. There are 3 options to authenticate: *(1)* Authenticate to an On-Premise Orchestrator using username and password *(2)* Authenticate to a Cloud Orchestrator using a refresh token (API key). The account name and API key are accessible via Services->API Access (see below for a detailed explanation on how to retrieve this) *(3)* Authenticate to a Cloud Orchestrator using external app authentication.
Trace logging level	Setting used to enable the trace logging to one of the following level: None, Critical, Error, Warning, Information, Verbose. (default None). Useful for debugging purposes.

Note :

Make sure that your network allows access to the following NuGet package feed:

- https://api.nuget.org/v3/index.json
- https://uipath.pkgs.visualstudio.com/_packaging/nuget-packages/nuget/v3/index.json
- https://uipath.pkgs.visualstudio.com/Public.Feeds/_packaging/UiPath-Internal/nuget/v3/index.json
- https://www.myget.org/F/workflow
- http://www.myget.org/F/uipath
- https://www.myget.org/F/uipath-dev/api/v3/index.json

☎ Pipeline Example:

```
pipeline  {
  agent any
  environment {
      MAJOR = '1'
      MINOR = '0'
  }
  stages {
    stage ('PostBuild') {
      steps {
      UiPathDeploy (
          packagePath: "path\\to\NuGetpackage",
          orchestratorAddress: "OrchestratorUrl",
          orchestratorTenant: "tenant name",
          folderName: "folder name",
          environments: "environment",
          credentials: [$class: 'UserPassAuthenticationEntry', credentialsId: “credentialsId”],
          traceLoggingLevel: 'None'
        )
      }
    }
  }
}
```

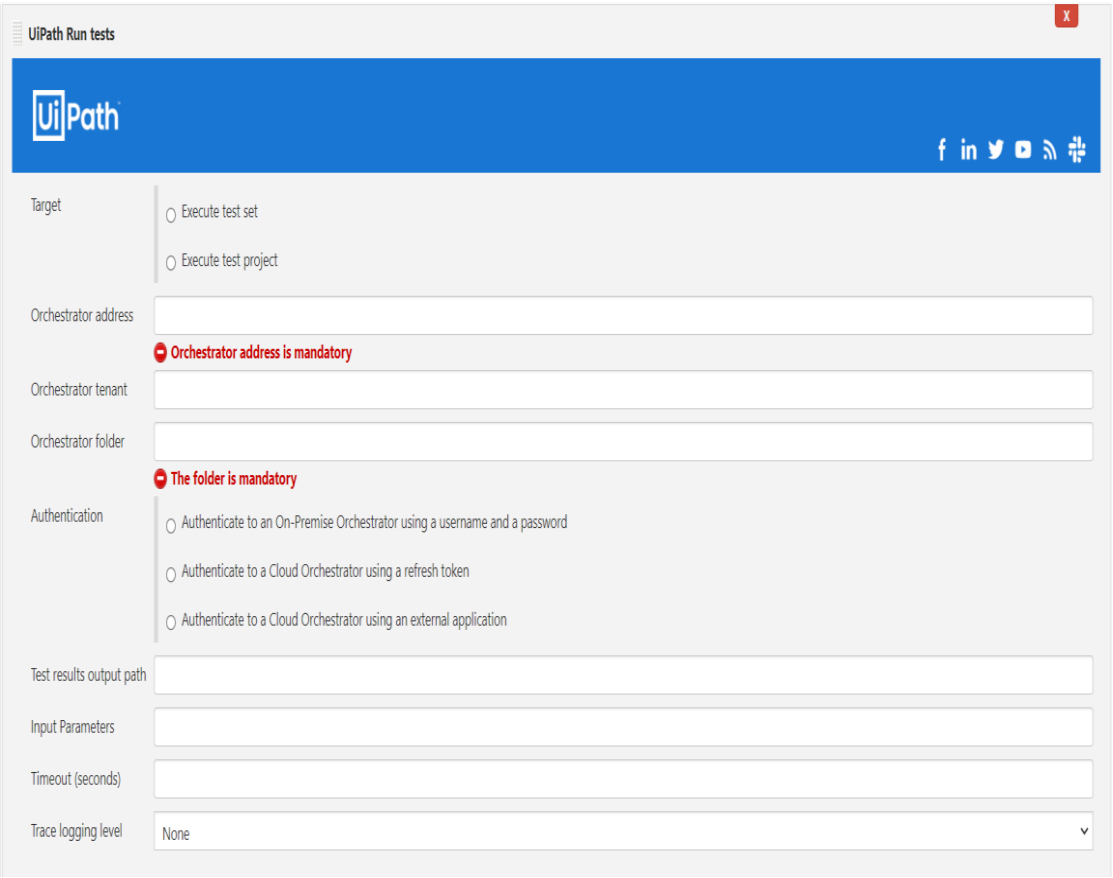
▶ UiPath Run tests
Application: Testing

Type: Post-Build task

UiPath Run tests is available in standard jobs and pipelines, and lets you *(1)* run an existing Test Set on Orchestrator, or *(2)* package, deploy and run test cases, by specifying the path to a UiPath test project.

After the test run has finished, the Test Result tab will be published to the Jenkins build, showing the detailed results. Additionally, a JUnit test results file will be output to the test result output path, when specified, or to the workspace root if not specified.

Depending on the result, the build will be either marked as successful (all test cases are passed), or unstable (at least one test case failed).



Configuration

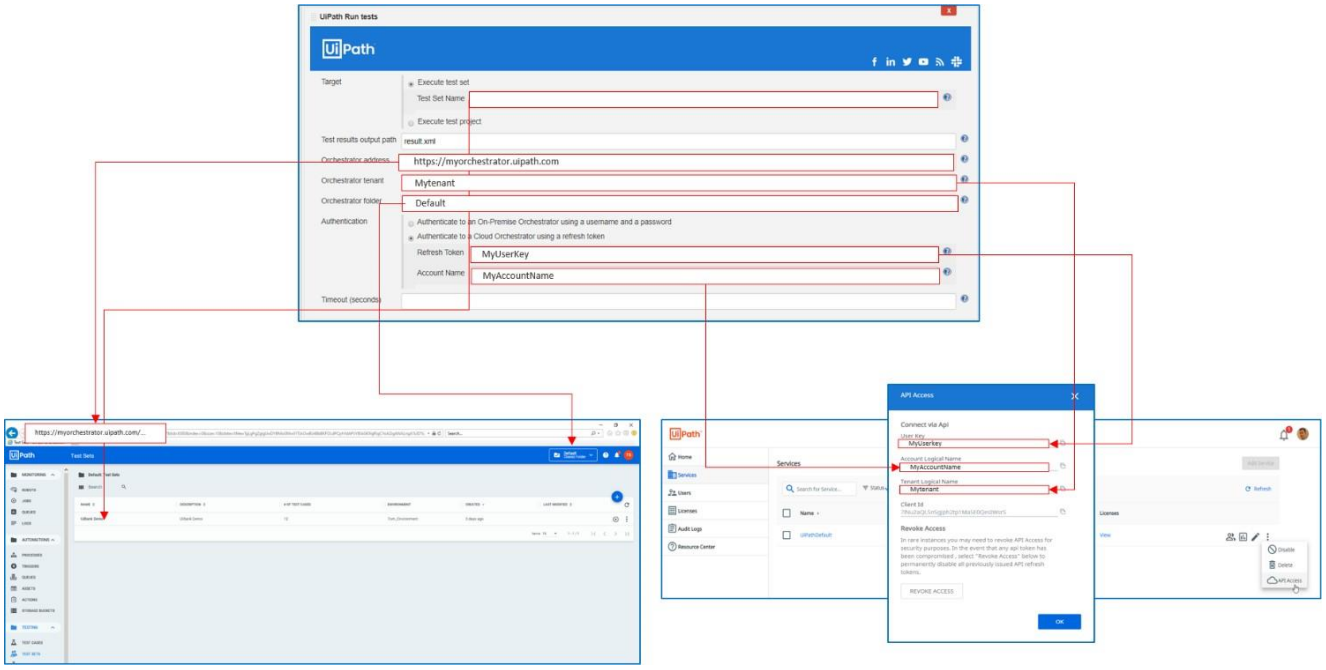
Job parameter	Description
Target	*(1)* Execute test set (specify an existing test set on UiPath Orchestrator) or *(2)* Execute test project (provide the project.json path of a UiPath Testing project)
Test result output path	The executed test set returns the test result as junit.xml. Specify the path where the result should be stored, relative to the Jenkins workspace directory (e.g. result.xml). *Optional*
Input Parameters	Define custom Arguments for your test cases to override default values at the test set level. Through the arguments, you can parameterize the test cases at runtime. To define arguments, you need to have published a package with arguments. *Optional*
Orchestrator address	The address of the Orchestrator instance onto which the package(s) will be deployed.
Orchestrator tenant	The Orchestrator tenant onto which the package(s) will be deployed.
Orchestrator folder	The folder to deploy to. If the folder is a classic folder, you will also need to set the environments field. For modern folders, setting the environments is not required.
Environments	The environment onto which the package will be deployed as a process. For the project and environment with existing processes, the processes will be updated to

Job parameter	Description
	use the latest project version. Specify the environment onto which the package will be deployed as a process. For the project and environment with existing processes, the processes will be updated to use the latest project version. Required when using a classic folder, otherwise not applicable.
Authentication	For authentication towards Orchestrator, credentials have to be created in Jenkins upfront. There are 3 options to authenticate: *(1)* Authenticate to an On-Premise Orchestrator using username and password *(2)* Authenticate to a Cloud Orchestrator using a refresh token (API key). The account name and API key are accessible via Services->API Access (see below for a detailed explanation on how to retrieve this) *(3)* Authenticate to a Cloud Orchestrator using external app authentication.
Timeout (seconds)	The execution timeout for the test run. The default value is 7200 seconds. If the timeout exceeds before the execution on Orchestrator is finished and returned the final result, the built will cancel and be marked as failed.
Trace logging level	Setting used to enable the trace logging to one of the following level: None, Critical, Error, Warning, Information, Verbose. (default None). Useful for debugging purposes.

☎ Pipeline Example:

```
pipeline  {
  agent any
  environment {
  MAJOR = '1'
    MINOR = '0'
  }
  stages {
    stage ('PostBuild') {
      steps {
        UiPathTest (
          testTarget: [$class: 'TestSetEntry', testSet: "My Test Set"],
          orchestratorAddress: "OrchestratorUrl",
          orchestratorTenant: "tenant name",
          folderName: "folder name",
          timeout: "10000",
          traceLoggingLevel: 'None',
          testResultsOutputPath: "result.xml",
          credentials: [$class: 'UserPassAuthenticationEntry', credentialsId: "credentialsId"]
        )
      }
    }
  }
}
```

Obtaining the Cloud Orchestrator API Key



Configure service connection for external apps

Step 1: Configure your external application and scopes in Automation Cloud. After adding the application, keep the App ID, Secret and Application Scopes at hand, to be used for the next step.

Note: If you generate and use a new Secret, the old one is going to be invalidated.

Step 2: Configure application credentials as secret text in Jenkins. For this step you need the Secret generated in Automation Cloud.

Step 3: Configure the Authentication for each task under Post-Build Actions, by adding the Account Name, followed by the App ID, Secret and Application Scopes generated through Automation Cloud.

Note: Consider using the external app in individual pipelines to avoid invalidation errors.

Additional information

All paths specified should be local to the current workspace. You can use environment variables in paths, though you should ensure that they result in paths that are local to the workspace. All paths

In order to deploy packages or run tests, ensure that the authenticated user has the Folders View (or OrganizationUnits View) and (20.4+ only) Background Tasks View permissions.

In order to package libraries when connected to an Orchestrator instance, ensure that the authenticated user has the Libraries View permission.

Authentication on Cloud Orchestrator using External Apps.

For further details on managing external apps on Orchestrator, pls refer to the official documentation.

Questions
Do you have any questions regarding the plugin? Ask them here.

Troubleshooting
Unauthorized error

Forbidden error

Folder/environment not found

Package already exists (Conflict)

Failed to run the command (Generic error))

Jenkins fails to process paths containing non-Latin characters

Unauthorized error
If using basic authentication:

- ensure the correctness of the username-password combination on the web login
- if federated authentication is enabled, make sure your write the username in the task as "DOMAIN\user"

If using token authentication:

- Revoke the token from the API access panel and generate a new one
- Ensure that the user that generated the key has can access the Orchestrator and has a user on the Orchestrator instance

If authenticating against on an on-premise Orchestrator you might receive this error as a result of the certificate used for the Orchestrator not being valid. This might mean that it has the wrong CN or other validation issues. Ensure that the Orchestrator certificate is valid and that the machine running the job trusts the Orchestrator certificate in case you are using a self-signed certificate.

Forbidden error
Likely, the user does not have the permission to perform the action.

Ensure that the user has permissions to read folders, upload packages, create and update processes, read test sets and test cases, create and run test sets, read background tasks.

Folder/environment not found
Ensure that the authenticated user used by CI/CD plugins has the Folders.View and (20.4 only) BackgroundTask.View permissions.

Package already exists (Conflict)
Ensure that the package that you are trying to deploy does not exist with the same version already. If it does, consider using automatic package versioning, so that the new version is bumped up every time we deploy.

Failed to run the command (Generic error)
If the Jenkins workspace is inside a location on disk (like C:\Windows or C:\Program Files) to which the user does not have permissions, ensure that the workspace is placed on a path that can be accessed smoothly by the user

Jenkins fails to process paths containing non-Latin characters
Jenkins is not able to pass correctly non-standard encoded characters when invoking the UiPath Plugin. The unknown characters will be replaced by ???.

The solution depends on how Jenkins is deployed on both the server and the agent host machines, but involves setting "file.encoding" to UTF-8 in Java options:

- Windows
  - Running Jenkins in Windows as a Service In the service configuration file add the arguments into the tag. It should look like this:

```
<arguments>-Xrs -Xmx512m -Dhudson.lifecycle=hudson.lifecycle.WindowsServiceLifecycle -Dfile.encoding=UTF-8 -jar "%BASE%\jenkins.war" --httpPort=8080 --webroot="%BASE%\war"</arguments>
```

  - Running Jenkins inside Docker The JAVA_OPTS should be passed to the container via --env JAVA_OPTS="..." like the following:

```
docker run --name myjenkins -p 8080:8080 -p 50000:50000 --env JAVA_OPTS=-Dhudson.lifecycle=hudson.lifecycle.WindowsServiceLifecycle -Dfile.encoding=UTF-8 jenkins/jenkins:lts
```

  - Running Jenkins inside Tomcat Use environment variable CATALINA_OPTS:

```
export CATALINA_OPTS="-DJENKINS_HOME=/path/to/jenkins_home/ -Dhudson.lifecycle=hudson.lifecycle.WindowsServiceLifecycle -Dfile.encoding=UTF-8 -Xmx512m"
```

- Linux
  - Debian / Ubuntu based Linux distributions In the configuration file search for the argument JAVA_ARGS and add the file enconding. It might look like this:

```
JAVA_ARGS="-Dfile.encoding=UTF-8 -Xmx512m"
```

  - RedHat Linux based distributions In the configuration file search for the argument JENKINS_JAVA_OPTIONS and add the file enconding. It might look like this:

```
JENKINS_JAVA_OPTIONS="-Dfile.encoding=UTF-8 -Xmx512m"
```

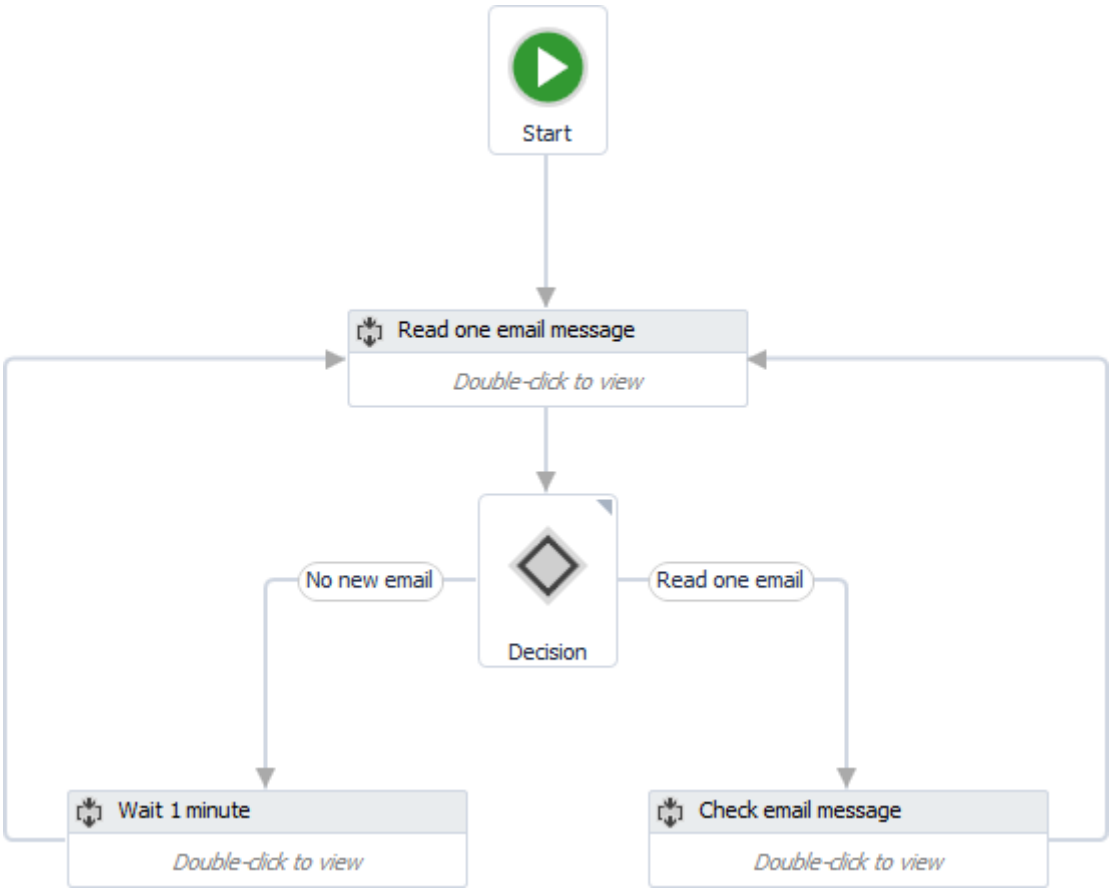
License

Practical 9A

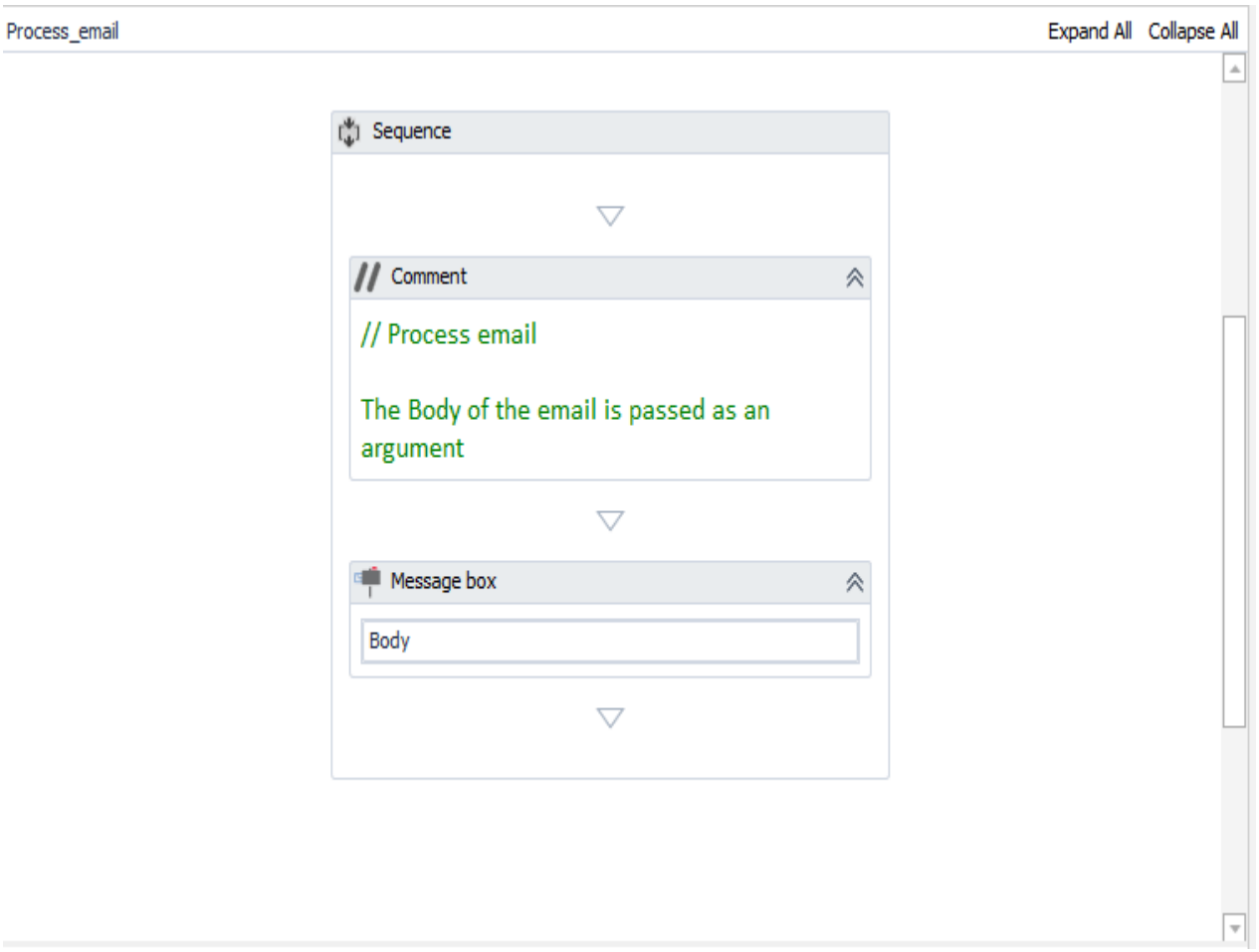
Automate the process of send mail event (on any email).

STEPS :

Email triggers are the foundation of any email automation process. In the attached ***Mail_Trigger_Sample*** **Workflow**, the Inbox is checked every 60 seconds for fresh emails.



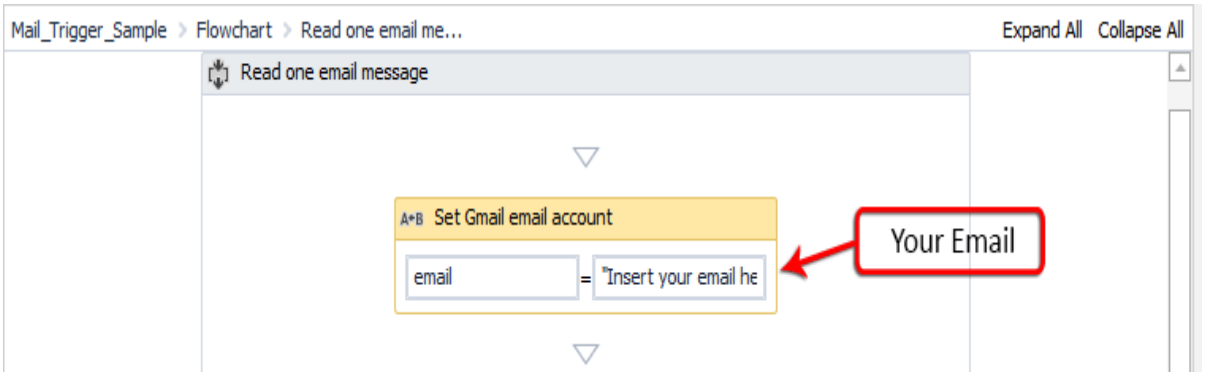
If a new email is detected and a certain condition is met (a specific Keyword is found in the Subject), then the message Body is passed as an argument to the ***Process_Email*** *W*orkflow.



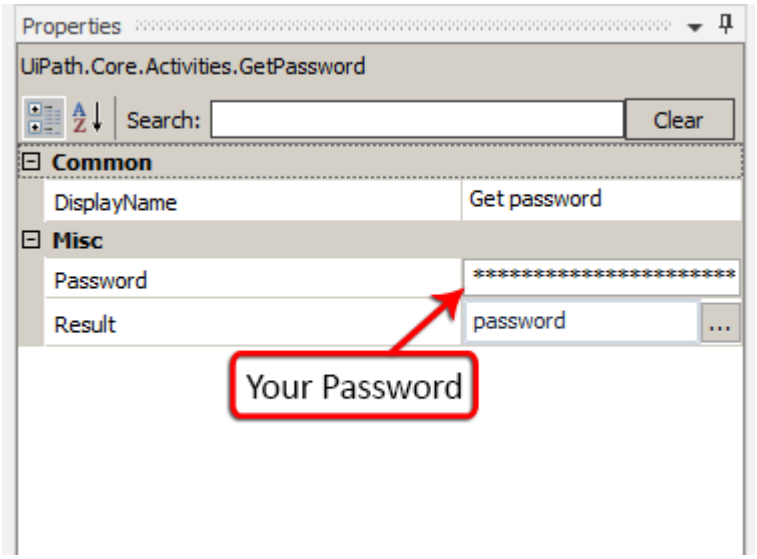
Workflow Steps:

1. Before running the ***Mail_Trigger_Sample*** workflow, you will need to set:

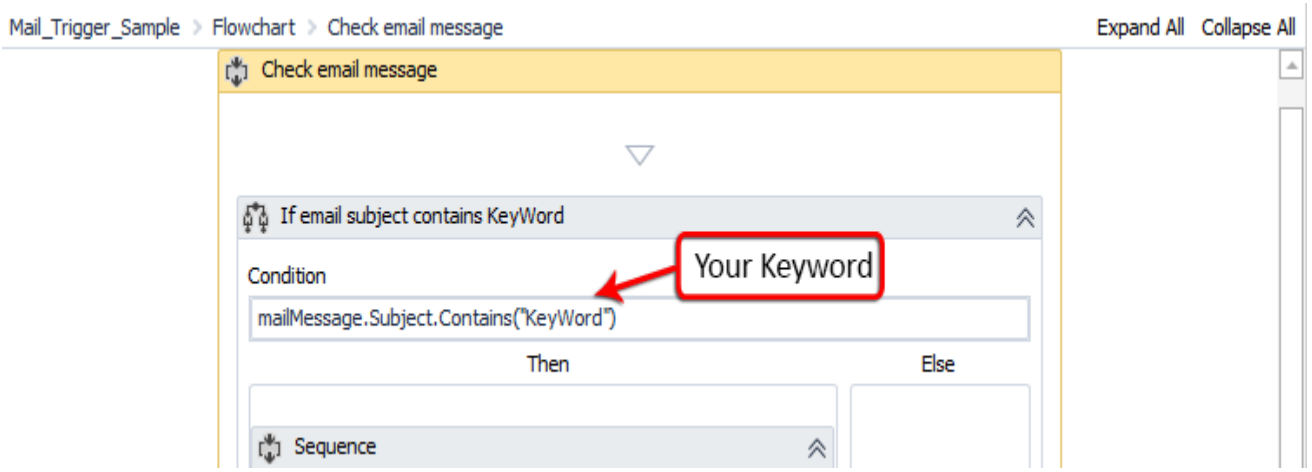
Email address:



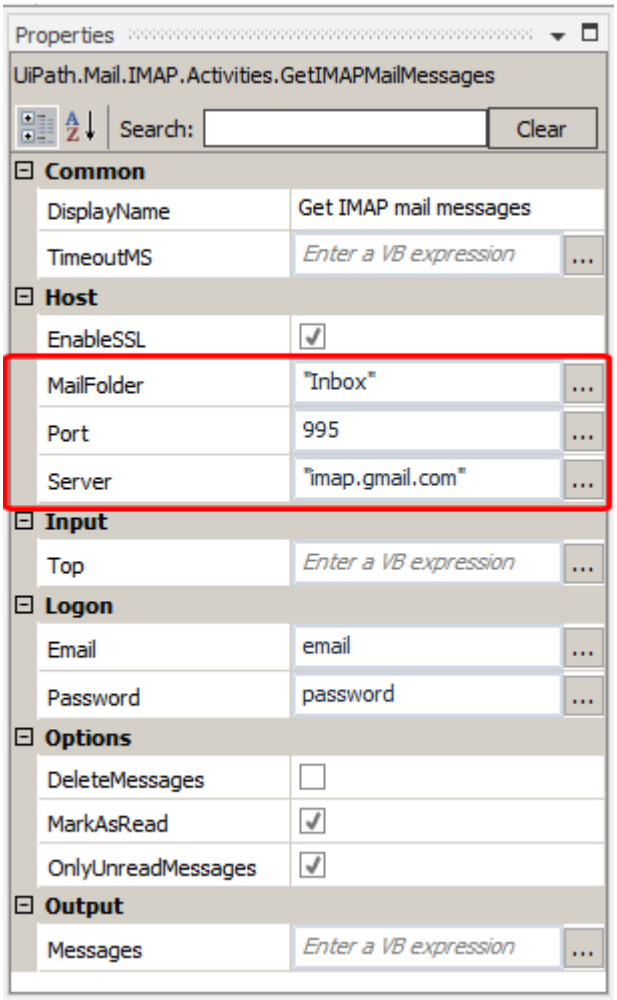
Password:



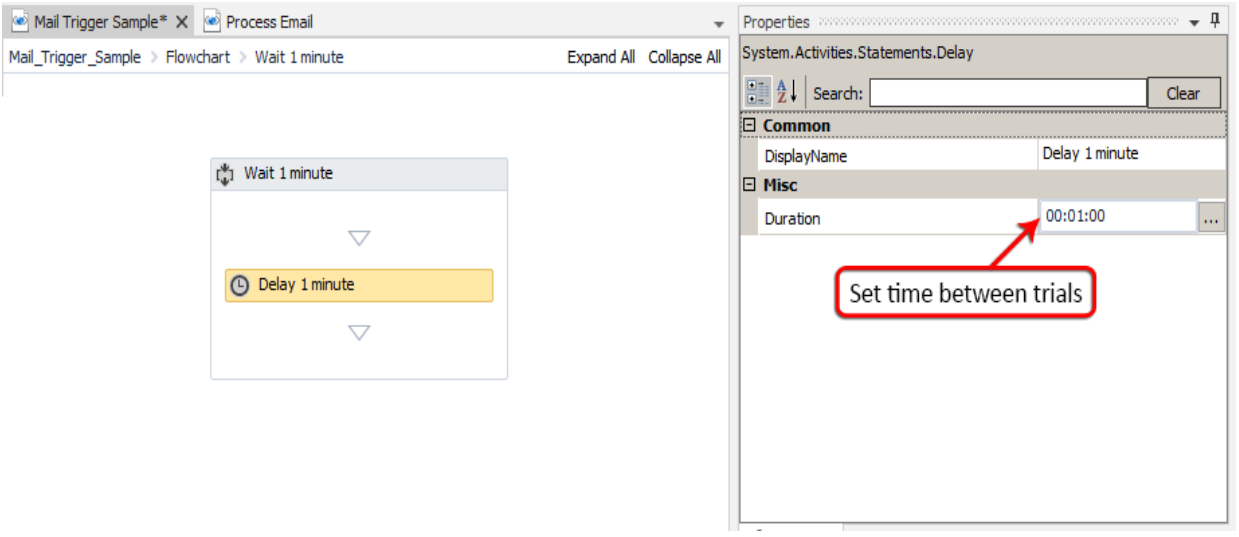
Keyword:



Also, please make sure you set your *MailFolder*, *Port*, and *Server* in the **GetIMAPMailMessages** activity properties. By default, the **Workflow** is configured to work with the Gmail settings.



2. When the **Workflow** is running, it checks your email for new messages (using the **GetIMAPMailMessages** activity). If no new email is found, it waits for 60 seconds and then it tries again. You can change the waiting time to other value, within the **Delay** activity.



3. If a new email is found, it is marked as read (you can change that by un-checking the *MarkAsRead* property in the **GetIMAPMailMessages** activity)

4. The **Workflow** then checks if the email's subject contains the keyword. If there's a match, the ***Process_Email*** **Workflow** is invoked and the email body is passed as an argument (`Body`)

5. The ***Process_Email*** **Workflow** will pop up a window with the message body.

Note: The workflows are created & tested with UiPath Studio 8.

Attachments:

Mail_Trigger_Sample.xaml

Process_Email.xaml

Practical 9B

Automate the process of launching an assistant bot on a keyboard event.

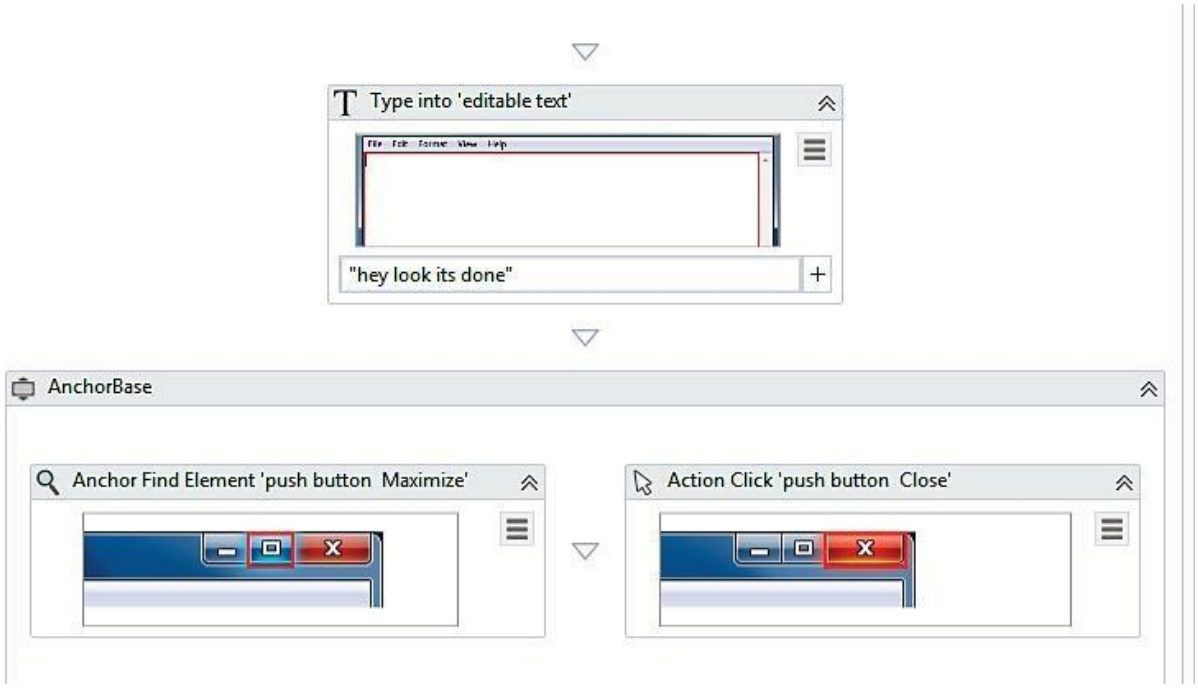
STEPS :

1.**Drag and drop the Monitor events activity**: In this step, we will just drag and drop the **Monitor events** activity into the workflow

2. Drag the Hotkey trigger activity: In the next step, we will use the **Hotkey trigger** activity for the user to start the automation process. Assign *Alt + W* to the hotkey so that, when the user presses this hotkey, the event will be executed:

3. Open Notepad and type into it: Our final step is to record the sequence of the steps to be performed. In this case, this is to open Notepad and then type into it. For that just use the help of the **Desktop** recorder. First, we double-click on the Notepad application in the window as shown in the screenshot. Select the **ClickType** as **CLICK_DOUBLE** from the **Properties** panel: After that, we record the typing action and close the Notepad window. Then click on **Do not Save** because you do not want to save your file. The sequence is shown in the following screenshot:

OUTPUT :



Practical 9C

Demonstrate the Exception handing in UiPath

STEPS :

1. Drag and drop the **Try catch** activity: Create a blank project. Drag and drop the **Flowchart** activity into the Designer panel. Search for the **Try catch** activity in the **Activities** panel and drag it into the **Flowchart**. Set it as the **Start** node:

2. **Try**: When we double-click on the **Try catch** activity, dragged and dropped inside the workspace, space for the **Try** activity appears

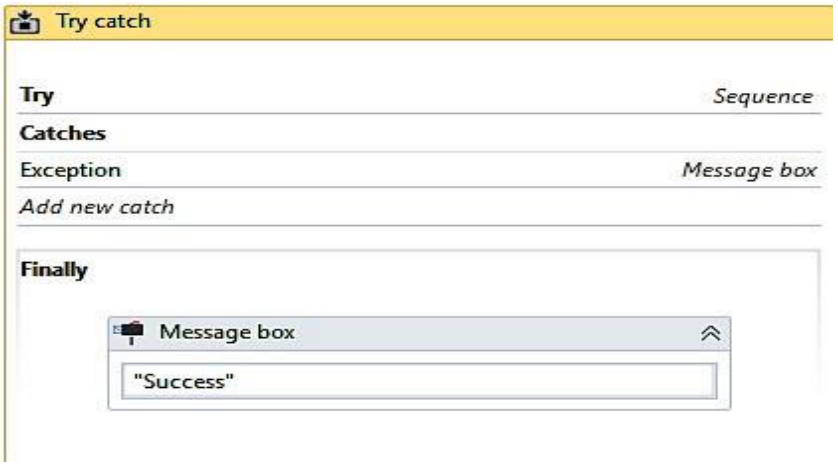
3. **Catches**: Inside the **Catches** activity, first we have to click on **Add new Catch** and then click on **Add Exception** option, from which we have to select the type of exception. In most cases, **System.Exception** is preferred. The following screenshot shows the types of exception. There are many more exceptions which can be viewed by clicking on the **Browse for Types** option:

4.Say the execution fails: for example, the **Click** activity is unable to be executed because of the unavailability of a UI element. In such a case, we can use the **Catches** block in order to either view the error that has occurred or for an alternative method to be used if that particular error occurs. As shown in the following screenshot, we will drop the activity in the **Catches** block. To print a message, we use a **Message box**:

5. When we click on **Add new catch**, we are asked to select the type of exception. We have selected **System.Exception**. Now inside the exception block, we have dropped a **Message box** activity. Entering will display theerror that occurred during execution

6. **Finally**: When we have defined the exception for our sequence, the **Finally** block will always work, regardless of whether the execution was successful or not. Suppose we want to display a message to the user notifying that the process is complete. To make sure that the whole **Try catch** activity is executed, we will just drop a **Message box** activity in the area provided in the **Finally** block

OUTPUT :



Practical 9D

Demonstrate the use of config files in UiPath

THEORY :

When it comes to configuration, UiPath does not have any pre-built configuration file such as Visual Studio, but we can create one. It is considered to be one of the best practices to keep environment settings in a config file so that they can be easily changed by the user when required. Thus, when we create a project, the file that holds all theactivities is created automatically. can be found in the folder where theproject is saved. To access the folder, we can just open the Project, then copy the path (as shown in the following screenshot), and paste it into File Explorer:

STEPS :

You can also store your settings with the help of a spreadsheet or credentials. There are various parameters contained in the file. They are:

Name: This is the title of the project that is provided when creating a project in the create New Project window:

Description: When creating a project, a description is also required. You can add the description in the Create New Project window, as shown in the preceding screenshot.
Managing and Maintaining the Code Chapter 9

Main: This is the entry point for the project. It is saved as by default,but you can change its name from the **Project** panel. Also, you have multiple workflows for a project, it is necessary to attach all these files to the main file with the **Invoke Workflow File** activity. Otherwise, those files will not be executed:
Dependencies: These are the activities packages that are used in a project and their versions.

Excluded data: Contains keyword that can be added to the name of an activity to prevent the variable and argument values from being logged at the verbose level.

Tool version: The version of Studio used to create a project.
Managing and Maintaining the Code Chapter 9

Adding Credential: We can add particular settings that can be used further. For example, we can save the username and password to be used further, so this can be done with the help of the **Add credential** activity that can be found in the **Activities** panel, as shown in the following screenshot:

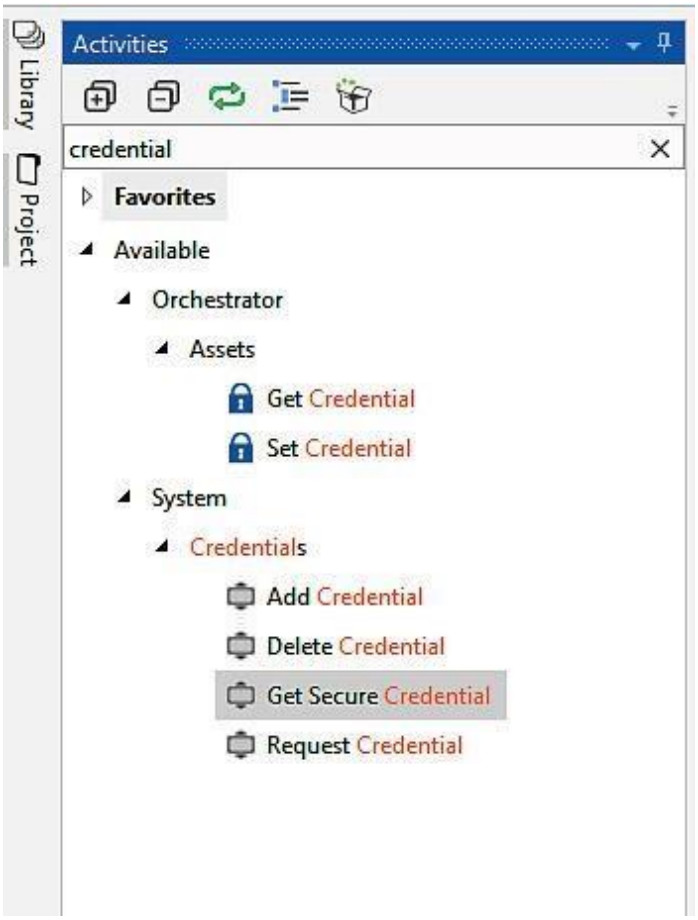
After adding credentials, type the required values in the **Properties** panel, as shown in the following screenshot:
Managing and Maintaining the Code Chapter 9

So, when the credentials are set, we can delete, secure, or request credentials, as shown in the following steps:
1. **Delete Credentials**: If we want to delete a credential then we can just drag and drop the **Delete Credentials** activity and then define the target for the credential:
Managing and Maintaining the Code Chapter 9

2. **Get Secure Credential**: This is used to get the values, that is, the username and password, that were set during the addition of a credential. We have to set the target the same as before; the output will be the username and password:

3. **Request Credential**: This is a property in which the robot displays a message dialog asking the user for username and password and stores this information as a string. This can then be used in further processes. The user can select OK to provide credentials or even cancel it if they do not want to provide credentials.

OUTPUT :



Practical 10A

Automate the process of logging and taking screenshots in UiPath

THEORY :

UiPath has a multi-process architecture that offers to execute each workflow separately in the executor. Executors are managed by UI robots. So, if any executor stops working, then the entire process will not be affected.

STEPS :

Common

- **Continue on error** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (True, False). The default value is False. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to True, the execution of the project continues regardless of any error.
- **Delay before** - Delay (in seconds) between the time the previous activity is completed and the time this activity begins performing any operations. The default value is 0.2 seconds. Adding a delay between activities ensures that one activity has enough time to complete before the next activity begins.
- **Delay before screenshot** - Delay (in seconds) between the time the element is brought into the foreground and the time the screenshot is taken. The default value is 0.2 seconds.
- **DisplayName** - The name displayed for the activity in the Designer panel. A display name is automatically generated when you indicate a target.
- **Timeout** - Specify a number of seconds for which to wait for the activity to be executed before throwing an error. The default value is 30 seconds.

Input

- **Auto increment** - Select what to append to the filename when saving the screenshot in case of filename conflicts. For example, if a file with the same name as the one you save already exists, choosing **Index** keeps the file name, creating a new file for each screenshot and adding an index number to each, consecutively. The options are **None** (if a file with the same name already exists, replace it), **Index** (add a number to the filename, for example screenshot (1).png), or **DateTime** (add the date and time when the screenshot is taken to the filename in the format YYYY.MM.DD at HH.MM.SS).
- **File name** - The name of the file where the screenshot of the specified UI elemend will be saved.
- **Target** - Before indicating on screen the application you want to automate, this field is set to (null). Once the target is indicated, all properties regarding the element that was indicated are displayed.
  - **Target.Click Offset** - Specifies an offset for the click activity, which can be further configured.
  - Target.Click Offset.Anchoring Point - Describes the starting point of the cursor to which offsets from OffsetX and OffsetY properties are added. The following options are available: TopLeft, TopRight, BottomLeft, BottomRight, and Center. By default, Center is selected.
  - Target.Click Offset.OffsetX - Horizontal displacement of the cursor position according to the option selected in the Position field. This field supports only Int32 variables.
  - Target.Click Offset.OffsetY - VerticaldDisplacement of the cursor position according to the option selected in the Position field. This field supports only Int32 variables.
  - **Target.Fuzzy selector** - The parameters for the fuzzy selector.
  - **Target.Native text** - The text that is used to identify the target element.

- **Target.Selector** - The selector that is generated for the indicated element.
- **Target.Targeting methods** - The selector types that you want to use for identifying the element. This property can be set to any combination of Selector, Fuzzy selector, or Image.
- **Target.Visibility check** - Checks whether the UI element is visible or not. There are three options available:
  - None - Does not check for visibility.
  - Interactive (for Fuzzy Selector) - Checks if the element is potentially visible, ignoring page scroll and obstructions by other apps, or the fact that the application is minimized. This check is useful when trying to ensure that you are not targeting invisible elements that exist in the DOM but are hidden.
  - Fully visible - Checks if the UI element is visible or not.
- **Target.Wait for page load** - Before performing the action, wait for the application to become ready to accept input. The following options are available:
  - None - Does not wait for the target to be ready.
  - Interactive - Waits until only a part of the app is loaded.
  - Complete - waits for the entire app to be loaded.
- **Target.Window selector (Application instance)** - The selector that is used for the application window. Only applicable when the window attach mode is set to Application instance.

Input/Output Element

- **Input Element** - The Input UI Element that defines the screen element that the activity will be executed on.
- **Output Element** - Output a UI Element to use in other activities as an Input UI Element.

Misc

- **Private** - If selected, the values of variables and arguments are no longer logged at Verbose level.

Output:

- **Saved file path** - The full path to the file where to save the screenshot, including the appended suffix, if applicable. This also dictates where the **Auto increment** property saves the indexed screenshot files.
- **Saved image** - The screenshot saved as Image; used when Output is set to image.

Practical 10B

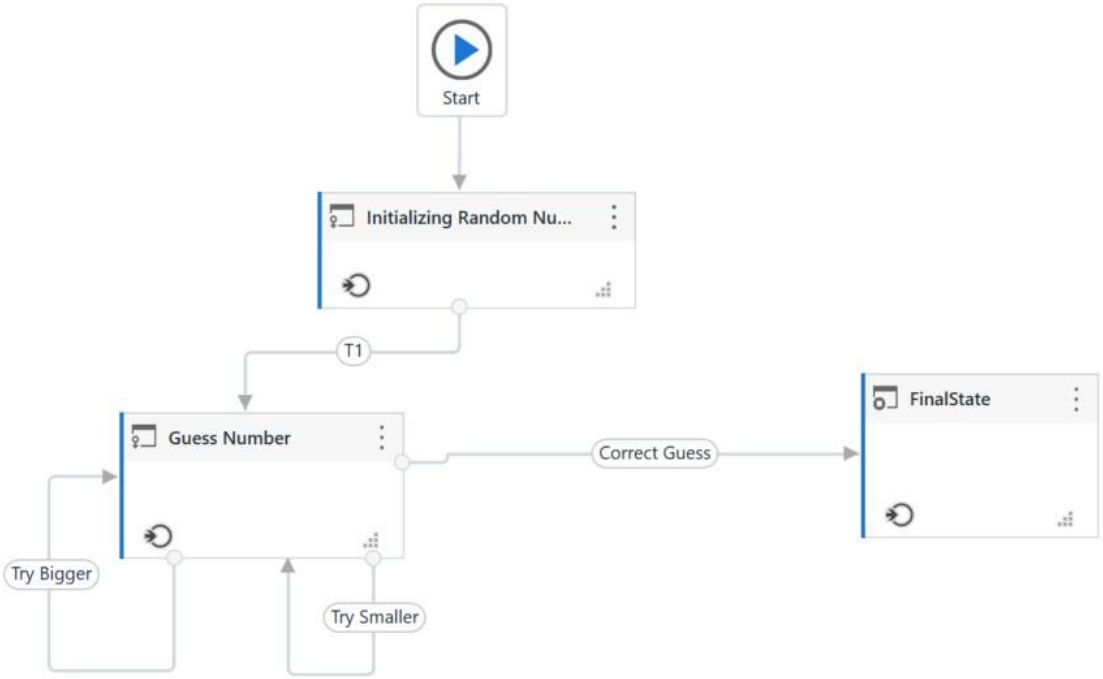
Automate any process using State Machine in UiPath

STEPS :

1. Create a blank process and, on the **Design** tab, in the **File** group, select **New > State Machine**. The **New State Machine** window is displayed.
2. In the **Name** field type a name for the automation, such as "First State Machine", and leave the default project location or add a subfolder. Click **Create**. The **Designer** panel is updated accordingly.
3. Create two integer variables, InitialGuess and RandomNumber. The first variable stores your guess, while the second stores the random number.
4. Add a **State** activity to the **Designer** panel and connect it to the **Start** node. This is the initial state, and it is used to generate a random number.
5. Double-click the activity. This **State** activity is displayed expanded in the **Designer** panel.
6. In the **Properties** panel, in the **DisplayName** field, type Initializing Random Number. This enables you to easily tell states apart.
7. In the **Entry** section, add an **Assign** activity.
8. In the **To** field, add the RandomNumber variable.
9. In the **Value** field, type new Random().Next(1,100). This expression generates a random number.
10. Return to the main project view and add a new **State** activity.
11. Connect it to the previously added activity.
12. Double-click the last added **State** activity. This activity is displayed expanded in the **Designer** panel.
13. In the **Properties** panel, in the **DisplayName** field, type Guess Number. This state is used to prompt the user to guess a number.
14. In the **Entry** section, add an **Input Dialog** activity.
15. Select the **Input Dialog**, and in the **Properties** panel, add an appropriate **Label** and **Title** to prompt the user to guess a number between 1 and 100.
16. In the **Result** field, add the InitialGuess variable. This variable stores the user's guess.
17. Return to the main project view and create a transition that points from the Guess Number state to itself.
18. Double-click the transition. The transition is displayed expanded in the **Designer** panel.
19. In the **Properties** panel, in the **DisplayName** field, type Try Smaller. This message is displayed on the arrow, enabling you to run through your automation easier.
20. In the **Condition** section, type InitialGuess > RandomNumber. This verifies if the user's guess is bigger than the random number.
21. In the **Action** section, add a **Message Box** activity.
22. In the **Text** field, type something similar to "Your guess is too big. Try a smaller number." This message is displayed when the user's guess is bigger than the random number.
23. Return to the main project view and create a new transition that points from the **Guess Number** state to itself.
24. Double-click the transition. The transition is displayed expanded in the **Designer** panel.
25. In the **Properties** panel, in the **DisplayName** field, type "Try Bigger". This message is displayed on the arrow, enabling you to run through your automation easier.
26. In the **Condition** section, type InitialGuess < RandomNumber. This verifies if the guess is smaller than the random number.
27. In the **Action** section, add a **Message Box** activity.

28. In the **Text** field, type something similar to "Your guess is too small. Try a bigger number." This message is displayed when the users guess is smaller than the random number.
29. Return to main project view and add a **Final State** activity to the **Designer** panel.
30. Connect a transition from the **Guess Number** activity to the **Final State**.
31. In the **Properties** panel, in the **DisplayName** field, type "Correct Guess".
32. In the **Condition** field, type InitialGuess = RandomNumber. This is the condition on which this automation steps to the final state and end.
33. Double-click the **Final State** activity. It is displayed expanded in the **Designer** panel.
34. In the **Entry** section, add a **Message Box** activity.
35. In the **Text** field, type something similar to "Congratulations. You guessed correctly! The number was " + RandomNumber.ToString + "." This is the final message that is to be displayed, when the user correctly guesses the number.

OUTPUT :



Practical 10C

Demonstrate the use of publish utility.

STEPS :

1. In Studio, create a new project.
2. In the **Design** ribbon tab, click **Publish**. The **Publish** window opens. Notice that the window's title bar changes depending on the context:

- **Publish Process** when publishing a process;
- **Publish Library** when publishing a library project;
- **Publish UI Library** when publishing a UI library project;
- **Publish Test Cases** when publishing test cases.
- **Publish Templates** when publishing templates.

3. In the **Package Properties** tab:

- Enter a name for the package. The drop-down list contains up to 5 of the most recent names of packages that you previously published.
- In the **Version** section, review the **Current Version** of your project, and type a **New Version** if needed. Check the **Is Prerelease** box to mark the version as alpha. Please note that this automatically changes the project's version schema to semantic. When publishing a new version of the file locally, make sure that the custom location does not already include a file with the same proposed version number. For more details about project versioning, check the **About Automation Projects** page.
- Optionally, use the **Project Icon** option to define a custom icon for the project. You can browse to and select a file, or enter a path or public URL to a jpeg, jpg, or png file up to 1MB in size.
  After the project is published, the icon is displayed as follows:

- For processes, in the Assistant next to the process name, making it easier to identify it in the list of processes.
- For templates, next to the template in **Home** (Studio Backstage View) > **Templates**.
- For libraries, next to the package in the **Manage Packages** window in Studio.

  The icon is not visible in Manage Packages if a local file is used for a library published to Orchestrator or to a feed that does not support embedded icons, In this case, specify the icon using a URL.

- In the **Project tags** box, you can add one or more tags to the project, either by creating new ones or by reusing tags already defined in Orchestrator. There are two types of tags: **labels** and **properties** (key-value pairs). Tags are included in the published package and they help describe and categorize projects. For example, they can refer to the automated application (an Excel label) or the department (a department:accounting key-value property).
  When you start typing, possible matches are suggested from already defined tags, and you can reuse one by selecting it from the list of matches. For a property match, the key followed by the : (colon) character is displayed first, and the associated values are displayed after you select the key,
  To add a new tag, after you enter the name, click the entry with the plus sign next to the name. Separating strings with the : (colon) character enables you to add properties, while entries that don't contain a : add labels.
  Labels and key-value properties are limited to 256 characters. Tag names can't contain these characters: <, >, %, &, \, ?, /, :.
  Project tags can be automatically applied to processes in Orchestrator. For more information about using tags, see Organizing resources with tags in the Orchestrator guide.

- In the **Release Notes** text box, enter details about the version and other relevant information. Release notes for published projects are visible in the **Packages** section in Orchestrator. Please note that the Release Notes field accepts a maximum of 10,000 characters.

4. Click **Next**.
   If you are publishing a template, the **Template info** tab opens next (step 5). Otherwise, proceed to step 6.
5. *(For templates only)* In the **Template info** tab, provide the following information, and then click **Next**:
   - **Name** - The name of the template.
   - **Description** - The template description in the **Templates** tab.
   - **Default Project Name** - The default project name when creating a new project using this template.

     Note: Avoid using punctuation marks, separator characters, and characters that are not allowed in file names. These characters may be removed from the default name when the template is used.
   - **Default Project Description** - The default description when creating a new project using this template.

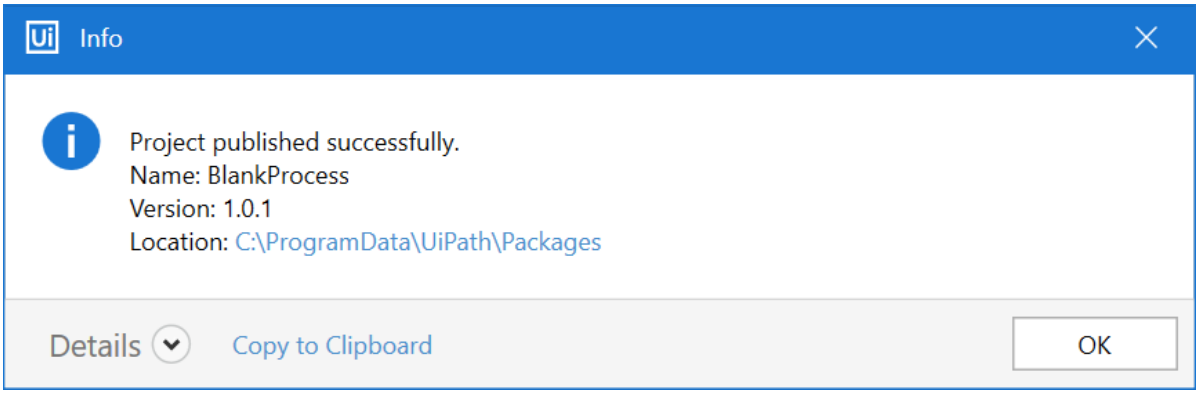
In the **Publish options** tab, select where to publish the project. The available options depend on the type of project you are publishing:

- For **processes** (including StudioX projects):
  - **Orchestrator Tenant Processes Feed**, **Orchestrator Personal Workspace Feed** and, if a first-level folder with a separate package feed or one if its subfolders is selected from the folders menu in the Studio status bar, the feed for that folder hierarchy. These options are available if Studio is connected to Orchestrator. The Orchestrator Personal Workspace Feed is available only if your user has the Personal Workspace feature enabled in Orchestrator.
    If the Personal Workspace or a folder from a hierarchy with a separate package feed is selected in Studio, the feed for that folder is the default option. Otherwise, the tenant feed is the default option. If you already published a project in the current session, the last publish location you used is the default selection until you close Studio or change the Orchestrator folder from the Studio status bar.
  - **Assistant (Robot Defaults)** - the default package location for the Robot and Assistant, C:\ProgramData\UiPath\Packages. Projects published here automatically appear in the Assistant. The option is not available if Studio is connected to Orchestrator.
  - **Custom** - either a custom NuGet feed URL or local folder. Adding an **API Key** is optional.
- For **test cases**:
  - The same options that are available for processes, with the exception of Orchestrator Personal Workspace Feed.
- For **libraries** and **UI libraries**:
  - **Orchestrator Tenant Libraries Feed** or **Orchestrator Shared Libraries Feed** - available if Studio is connected to Orchestrator. The available option depends on whether the tenant libraries feed is enabled in Orchestrator.
  - **Custom** - either a custom NuGet feed URL or local folder. Adding an **API Key** is optional.
- For **templates**:
  - **Orchestrator Tenant Libraries Feed** or **Orchestrator Shared Libraries Feed** - available if Studio is connected to Orchestrator. The available option depends on whether the tenant libraries feed is enabled in Orchestrator.
  - **Local** - the location for publishing templates locally, by default: C:\Users\User\Documents\UiPath\.templates.
  - **Custom** - either a custom NuGet feed URL or local folder. Adding an **API Key** is optional.

If you are publishing a library or any project with the Windows or cross-platform compatibility except for templates, additional settings are available in the **Publish options** tab under **Compilation Settings**:

- *(For libraries only)* **Activities Root Category** - enter a name for the category under which the reusable component will be listed in the **Activities** panel.
- **Include Sources** - select this option to package all .xaml sources within the published package, including workflows that were previously made private. For Windows - Legacy libraries, the files are saved in the generated assembly file and in the lib\net45 folder in the .nupkg file. For Windows and cross-platform libraries and processes, the files are saved in the content folder in the .nupkg file.
- **Remove Unused Dependencies** - select this option to remove all installed packages that are not referenced in the project. This option is not available for Windows-legacy processes.
- *(For Windows-legacy libraries only)* **Compile activities expressions** - select this option to compile and package all activities expressions. This results in an improved execution time.
- *(For Windows - legacy libraries only)* **Ready to Run** - select this option to optimize the generated assemblies for faster JIT compilation at runtime.
    - Click **Next** to advance to the **Certificate signing** tab, or **Publish** to publish your project.
    - *(Optional)* In the **Certificate Signing** tab, add a local **Certificate Path** next to the **Certificate** box. Furthermore, add the **Certificate Password** and an optional certificate **Timestamper** if needed. For more details, check out the **Signing**
    - Click **Publish**. A NUPKG file is created and uploaded to Orchestrator, the custom NuGet feed, or saved in the local directory. Depending on the project, the package contains:
        - For template projects and processes with the Windows - Legacy compatibility, the project source files.
        - For libraries and projects with the Windows or Cross-platform compatibility, compiled DLL files.
    - If the project is published successfully, the **Info** dialog box is displayed and the project is copied to the NuGet location set in the NuGetServerUrl parameter, in the UiPath.settings file.

OUTPUT :



Practical 10 D

Create and provision Robot using Orchestrator

THEORY :

When deploying multiple Standard Robots from the same machine on Orchestrator, you need to have the same **Machine Name** and **Machine Key** for each. To retain the values in the fields, you can click **Create Another** in the **Add Robot** window. Alternatively, you can copy the **Machine Name** and **Machine Key** from an already deployed Robot by clicking **More Actions > Duplicate**.

STEPS :

1. In Orchestrator, on the **Robots** page, click **Add**. The options to add a **Standard Robot** or a **Floating Robot** are displayed.
2. Click the **Standard Robot** button. The **Create a New Standard Robot** window is displayed.
3. Select the **Standard Machine** you want to register your Robot on. There are two possibilities:
   a. You had already created the machine on the **Machines** page beforehand. In this case, you can select it from the **Machine** drop-down list.
   b. You had not created the machine. In this case, simply type the name of a new one on the **Machines** field and click the **Provision Machine** button. Note that this step also adds the machine in the **Machines** page.
4. In the **Name** field, type any name for the Robot.
5. In the **Domain\Username** field, type the username that is used to login to the specified machine. If the user is in a domain, you are required to specify it in a DOMAIN\username format. You must use short domain names, such as desktop\administrator and NOT desktop.local/administrator.
6. (**Optional**) Add the Windows password for the specified username.
7. Select the desired robot type from the **Type** drop-down list. For more information, see the **About Robots** page.
8. (**Optional**) Add a description for the Robot. We recommend populating this field, especially when dealing with an environment with many robots.
9. Click **Create**. The Robot is now displayed on the **Robots** page and provisioned to Orchestrator, but it is offline.