



SUMMER TRAINING REPORT  
ON  
**“MACHINE LEARNING USING PYTHON”**

SUBMITTED BY - SHIVANSH SHARMA  
ROLL NO.- 17042100309  
CLASS – B.TECH (ECE) SEM 5  
BATCH (2021-2025)

# *CONTENTS:*

1.CERTIFICATE.....	
2.ACKNOWLEDGEMENT.....	
3.INTRODUCTION.....	
3.1 INTRODUCTION.....	
3.2PYTHON PROGRAMMING.....	
3.4LITERALS AND CONSTANTS.....	
3.5TYPE CONVERSION.....	
3.6DATA TYPES.....	
3.7PYTHON LIBRARIES.....	
3.8MACHINE LEARNING.....	
4.PROJECT.....	
4.1 LOAN PREDICTION USING MACHINE LEARNING....	

# CERTIFICATE



## ACKNOWLEDGMENT:

I would like to express my sincere gratitude to the following individuals and organizations who played a significant role in making my industrial training a valuable and enriching experience:

1. GNDU, Amritsar: I am thankful to Guru Nanak Dev University, Amritsar for providing me with the opportunity to undergo this training, which has been instrumental in enhancing my knowledge and skills.
2. Mr.Sarvesh Agarwal :- I extend my thanks to Sarvesh sir for their guidance and mentorship throughout my training their expertise and insights were valuable in my training from Internshala.
3. My Family and Friends: Last but not least, I would like to thank my family and friends for their unwavering support and encouragement throughout this journey. Their belief in me was my driving force.

# CHAPTER 1

## INTRODUCTION

- It was created by Guido van Rossum in Netherlands, and released in 1991.
- The name "Python" was adopted from the Rossum's favourite comedy series "Monty Python's Flying Circus".
- It is a high level general purpose programming language.
- It is compiled to byte code and executed in Python Virtual Machine.
- It is suitable for use as a scripting language, Web application implementation language, etc.
- It has a strong structuring constructs (nested code blocks, functions, classes, modules, and packages) and the use of objects and object oriented programming, enables us to write clear, logical applications for small and large tasks.
- Python is interpreted language.
- Python is available as Free and Open Source.
- Python run on different platform like Windows , Linux , Unix etc.

## **Python programming:**

- Comments – Everything after # on a line is ignored.
- Blocks and Indentation – It follows the block structure and nested block structure with indentation. ABC Block-1 ABC Block-2 ABC Block-3
- Lines – Statement separator is a semi colon, but needed only when there are more than one statement on a line.
- Variables - Variables are containers for storing data values.
- Python has no command for declaring a variable.
- A variable is created at the time, first value assign to it.
- Rules for Variable Naming - A variable can have a short name (like x and y) or a more descriptive name (age, EmpName, total\_volume).

## **Rules for Python variables:**

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number

- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
- Variable names are case-sensitive
- Example - age, Age and AGE are three different variable.
- Reserve Words – It can not be used as constant or any other identifier name. and exec not assert finally or break for pass class from print continue global raise def if return del import try elif in while else is with except yield lambda
- Basic Data Types - The data stored in memory can be of many types. • For example, a person's age is stored as numeric value and his or her address is stored as alphanumeric characters.. Python has five standard data types Numbers String List Tuple Dictionary. Python Identifiers. It is a name used to identify a variable, function, class, module or other objects in Python program. An identifier starts with a letter (A to Z) or (a to z) or an underscore) followed by zero or more letters, underscores and digits (0 to 9). Python does not allow the characters @, \$, and % Valid identifier- myvar, my var,

## **Literals and constants**

### **Literals:**

- Literal is a raw data assigned to a variable or constant.
- Literals are immutable.
- **Types of Literal**
  - Text: str
  - Numeric int, float, complex
  - Boolean Type: bool
  - Special: None
  - Numeric Literal
  - Constants
- A constant is a type of variable whose value cannot be changed, during the execution of program. In python, it does not prevent reassignment. Constants are usually declared and assigned in a module. Module is a file containing variables, functions, etc which is imported to the main file,
- **Numeric Literal**- Numeric int, float, complex



- **Integer Literal**- No fractional part allowed.
- **Types of Integer Literal** - Decimal Literal (base 10)-  
10 Binary Literal (base 2)- 0b1010 Octal Literal  
(base 8)-0010 Hexadecimal Literal (base 16)-  
0x12cFloat Literals Fractional part allowed.
- Types of Float Literal Fixed Literal - 10.45
- Scientific Literal-10e2, 10E210c2 is equivalent to  
10 X 10<sup>2</sup> 10e-2 is equivalent to 10 X 10<sup>-2</sup>
  - Complex Literal It has real and imaginary part  
10+12j)
- String Literal It is a sequence of characters  
surrounded by quotes. Quotes can be single,  
double, or triple quotes for a string. A string that is  
prefixed with an r or R before the opening quotes  
is a "raw" string

### **Type conversion-**

- Type Conversion Casting allows to specify a type  
on to a variable.
- int(x) Converts x to an integer, base specifies the  
base if x is a string
- long(x) Converts x to a long integer. base specifies  
the base if x is a string.

- `float(x)` Converts x to a floating-point number
- `complex(real [imag])` Creates a complex number.
- `str(x)` Converts object x to a string representation.
- `Tuple(s)` Converts to a tuple

## Python libraries

- Numpy:

Numpy is python package and it stands for Numerical Python. It is core python library used working with arrays and for scientific computing. It also has inbuilt functions for working with linear algebra, Fourier transform, matrices and data science. It has an N-dimensional array object(ndarray) which is in the form of rows and columns.

## NumPy Data Types

Python provides the following data types:-

- **strings**- text data enclosed in quote marks, eg. "ABCD"
- **integer**- integer numbers, eg. 1,2,3,-1,-2,-3
- **float**- real numbers. eg. 12.2, 12.42
- **Boolean**- True or False
- **complex** -a number in complex form. eg. 2.0+2.0j, 2.5+1.5j

## **pandas:**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures.

The name Pandas is derived from the word Panel Dataan Econometrics from Multidimensional data.

## **PYTHON DATAFRAMES**

A pandas DataFrame can be created using the following constructor -pandas. DataFrame( data, index, columns, dtype, copy)

### **Missing Values**

Data in real world are rarely clean and homogeneous. They tend to be incomplete, noisy, and inconsistent and it is an important task of a Data scientist to pre-process the data by filling missing values.

## **MACHINE LEARNING:**

### **Supervised Learning**

- Supervised learning is a type of machine learning method in which we provide sample labeled data to the machine learning system in order to train it.
- Supervised learning can be grouped further in two categories of algorithms:

➤ Classification

➤ Regression

### **Regression:**

In regression we do prediction in the format of number or continuous value. It predicts continuous/real values such as temperature, age, salary, price, etc.

### **Classification:**

In classification we do prediction in the form of categorical value. Means we predict specific category or class. Example in gmail we classify mails in spam or not spam.

### **Unsupervised Learning**

- Unsupervised learning is a learning method in which a machine learns without any supervision.
- It can be further classified into two categories of algorithms:

- Clustering
- Association

## **Reinforcement Learning**

In Reinforcement learning model perform some task and learn from the output. Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action.

# Chapter – 2

## Project – Loan prediction

1.



Made a python file in  
jupyter notebook as  
[loan\\_prediction.ipynb](#)



## 2.

```
In [5]: #importing all the modules

In [1]: import numpy as np

In [2]: import pandas as pd

In [3]: import matplotlib.pyplot as plt

In [4]: import seaborn as sns

In [5]: from sklearn.model_selection import train_test_split

In [6]: from sklearn.metrics import accuracy_score
```

### LIBRARIES USED

- Tkinter for GUI
- Numpy for mathematical operations
- Pandas for working with data sets
- Matplotlib for statistics
- Seaborn for data science and machine learning tasks
- Sklearn for classification, regression, clustering and dimensionality reduction, etc.

15

## 3. Scale the data:

```
In [9]: df.describe()

Out[9]:
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.000000	564.000000
mean	5403.459283	1621.245798	146.412162	342.000000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

```
In [10]: df.isnull().sum()
```

```
Out[10]:
```

Loan_ID	0
Gender	13
Married	3
Dependents	15
Education	0
Self_Employed	32
ApplicantIncome	0
CoapplicantIncome	0
LoanAmount	22
Loan_Amount_Term	14
Credit_History	50
Property_Area	0
Loan_Status	0

dtype: int64

```
In [11]: # h_map=df.corr()
# f,ax=plt.subplots(figsize=(9,9))
# sns.heatmap(h_map,vmax=.8,square=True);
```

```
In [12]: #dropping the missing values
df=df.dropna()
```

```
In [13]: df.isnull().sum()
```

```
Out[13]:
```

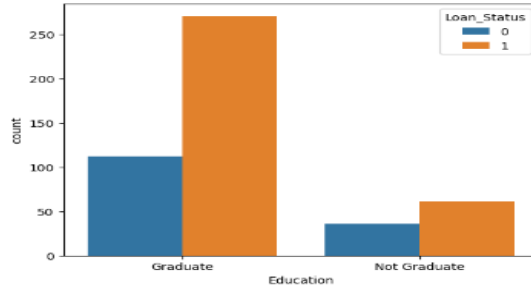
Loan_ID	0
Gender	0
Married	0
Dependents	0
Education	0
Self_Employed	0
ApplicantIncome	0
CoapplicantIncome	0
LoanAmount	0
Loan_Amount_Term	0
Credit_History	0
Property_Area	0
Loan_Status	0

dtype: int64

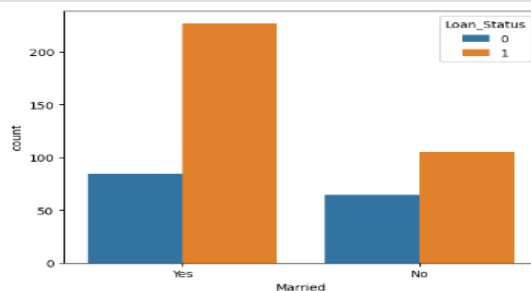
# 4. Data Visualisation:

## Data Visualization

```
In [19]: #Education and Loan status
sns.countplot(x='Education',hue='Loan_Status',data=df);
```



```
In [20]: #Marital status and Loan status
sns.countplot(x='Married',hue='Loan_Status',data=df);
```



# 5. Test-train split

## Test Train Split

```
In [29]: X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.1)
```

```
In [30]: print(X.shape,X_train.shape,X_test.shape)
(480, 11) (432, 11) (48, 11)
```

```
In [31]: #feature scaling
from sklearn.preprocessing import StandardScaler
st_X=StandardScaler()
X_train=st_X.fit_transform(X_train)
X_test=st_X.transform(X_test)
```

```
In [32]: X_test
[[ 0.45834925,  0.74428277, -0.70590944, -2.02048334, -0.40935018,
 -0.62905882,  0.48207297, -0.64714737,  0.27711956, -2.35487888,
  1.2588625 ],
 [ 0.45834925,  0.74428277, -0.70590944,  0.49493108,  2.44289622,
 -0.49247865,  0.78145088,  0.08872112,  0.27711956,  0.42465029,
 -1.3307975 ],
 [-2.18174242, -1.34357538, -0.70590944,  0.49493108, -0.40935018,
 -0.52745864, -0.59436673, -1.69482453,  0.27711956,  0.42465029,
  1.2588625 ],
 [ 0.45834925,  0.74428277,  2.53827013, -2.02048334, -0.40935018,
 -0.58142275, -0.31773567, -0.63467502,  2.12743875,  0.42465029,
 -0.0359675 ],
 [ 0.45834925,  0.74428277,  0.91618034,  0.49493108, -0.40935018,
 -0.29209075, -0.59436673, -0.31039399,  0.27711956,  0.42465029,
 -0.0359675 ],
 [ 0.45834925, -1.34357538, -0.70590944,  0.49493108, -0.40935018,
 -0.37347636, -0.59436673, -1.18345829,  0.27711956,  0.42465029,
  1.2588625 ],
 [ 0.45834925,  0.74428277,  0.91618034, -2.02048334, -0.40935018,
 -0.35537114,  0.45455662, -0.01105766,  0.27711956,  0.42465029,
  1.2588625 ]]
```

```
In [33]: X_train
```

```
Out[33]: array([[ -2.18174242,  0.74428277, -0.70590944, ...,  0.27711956,
  0.42465029, -0.0359675 ],
 [ 0.45834925,  0.74428277,  0.10513545, ...,  0.27711956,
  0.42465029, -0.0359675 ],
 [ 0.45834925, -1.34357538, -0.70590944, ..., -2.49835921,
 -2.35487888,  1.2588625 ],
 ...,
 [ 0.45834925, -1.34357538,  2.53827013, ...,  0.27711956,
  0.42465029, -1.3307975 ],
 [-2.18174242, -1.34357538, -0.70590944, ...,  0.27711956,
  0.42465029, -0.0359675 ],
 [ 0.45834925, -1.34357538, -0.70590944, ...,  0.27711956,
  0.42465029, -0.0359675 ]])
```



## 6. Training the model

### Training the model :

## LogisticRegression

```
In [34]: from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

```
In [35]: mymodel=LogisticRegression()  
mymodel.fit(X_train,Y_train)  
pred=mymodel.predict(X_test)
```

```
In [36]: print("Accuracy : ",accuracy_score(Y_test,pred))
print(confusion_matrix(Y_test,pred))
print(classification_report(Y_test,pred))
```

Accuracy : 0.7708333333333334

```
[[ 4 11]
 [ 0 33]]
```

	precision	recall	f1-score	support
0	1.00	0.27	0.42	15
1	0.75	1.00	0.86	33
accuracy			0.77	48
macro avg	0.88	0.63	0.64	48
weighted avg	0.83	0.77	0.72	48

```
In [37]: print(pred)
```

[illegible]

```
In [38]: #taking data from the dataset
```

```
my_prediction=mymodel.predict([[1,1,0,1,1,4583,1508,128,360,1,2]])
```

```
In [39]: my_prediction
```

```
Out[39]: array([1], dtype=int64)
```

**7.**



Made another python file in jupyter notebook as Myapp.ipynb which will end up running as a GUI interface.

# 8. Design of GUI

```
class myappclass:
    def __init__(self):
        self.window=Tk()
        #=====Background Image=====

        w = self.window.winfo_screenwidth()
        h = self.window.winfo_screenheight()

        self.bimg1 = Image.open("image/img2.jpeg")
        self.bimg1 = self.bimg1.resize((w,h), Image.LANCZOS)

        self.bimg2 = ImageTk.PhotoImage(self.bimg1)

        self.bklbl1=Label(self.window,image=self.bimg2)
        self.bklbl1.place(x=0,y=0)

        self.window.title("Loan Prediction App")
        self.window.geometry("%dx%d+%d+%d"%(500,550,500,100))
        self.headlbl = Label(self.window, text="Loan Prediction", font=("Lucida Calligraphy", 20))
        self.l1 = Label(self.window, text=" Gender")
        self.l2 = Label(self.window, text=" Are you Married ")
        self.l3 = Label(self.window, text="How much persons depends on you")
        self.l4 = Label(self.window, text="What is your Education Qualification")
        self.l5 = Label(self.window, text="Are you Self Employed ")
        self.l6 = Label(self.window, text="How much is your Income")
        self.l7 = Label(self.window, text="How much Coapplicant's Income")
        self.l8 = Label(self.window, text="Loan Amount (in thousand)")
        self.l9 = Label(self.window, text=" Loan Amount Term (in days)")
        self.l10 = Label(self.window, text="Credit History ")
        self.l11 = Label(self.window, text="Property Area")
        self.l12 = Label(self.window, text=" -----")

#=====
        self.v1 = StringVar()
        self.r1 = Radiobutton(self.window, text='Male', value='Male', variable=self.v1)
        self.r2 = Radiobutton(self.window, text='Female', value='Female', variable=self.v1)
        self.v1.set(" ")

        self.list1=["Yes","No"]
        self.v2 = StringVar()
        self.c1= Combobox(self.window, values=self.list1, textvariable=self.v2, state='readonly')
        self.v2.set("Yes")

        self.t1 = Entry(self.window)

        self.list2=["Graduate","Not Graduate"]
        self.v3 = StringVar()
        self.c2= Combobox(self.window, values=self.list2, textvariable=self.v3, state='readonly')
        self.v3.set("Graduate")

        self.v4 = StringVar()
        self.c3= Combobox(self.window, values=['Yes','No'], textvariable=self.v4, state='readonly')
        self.v4.set("Yes")

        self.t2 = Entry(self.window)

        self.t3 = Entry(self.window)

        self.t4 = Entry(self.window)

        self.t5 = Entry(self.window)

        self.v5 = StringVar()
        self.c4= Combobox(self.window, values=['Yes','No'], textvariable=self.v5, state='readonly')
        self.v5.set("Yes")

        self.list3=["Urban","Semi-Urban","Rural"]
        self.v6 = StringVar()
        self.c5= Combobox(self.window, values=self.list3, textvariable=self.v6, state='readonly')
```

## 9. How interface appears

Loan Prediction App

### Loan Prediction

Gender ☐ Male ☐ Female

Are you Married

How much persons depends on you

What is your Education Qualification

Are you Self Employed

How much is your income

How much Coapplicant's Income


Loan Amount (in thousand)

Loan Amount Term (in days)

Credit History

Property Area

.....



# THANK YOU!