

Industrial Internship Report on

” URL SHORTENER Project Using Python”

Prepared by

K.ABDUL KALAM

Executive Summary

This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT). This internship was focused on a project/problem statement provided by UCT.

Description: The URL shortener is a Python project that converts long URLs into shorter, more manageable links. It takes a long URL as input, generates a unique shortened URL, and redirects users to the original URL when the shortened link is accessed.

Scope: The scope of this project involves designing a user interface to input long URLs and display the shortened links, implementing a database to store the mapping between original and shortened URLs, and developing functions to generate unique shortened URLs and handle redirection.

TABLE OF CONTENTS

1	Preface.....	2
1.1	week 1	
1.2	week 2	
1.3	week 3	
1.4	week 4	
2	Introduction	
2.1	About UCT	
2.2	About upskill Campus	
2.3	Objective	
3	My Project	
4	Project Implementation	
5	My learnings	
5.1	code submission	
5.2	Report submiision	

1 Preface

1.1 Week 1:

In Week 1 ,I learned about Basics Of Python. It is a highlevel programming language and it is easy to understand. It is open source and platform independent. It is developed by Guido Van Rossum.

Python Libraries used for providing key features for data Science. It also offers oop programming.

Python has a great library ecosystem.

In this I specifically works on some python libraries such as Numpy , Pandas and Matplotlib

I used Spyder IDE(Integrated Development Environment) for my coding exercises.

The screenshot shows the Spyder IDE interface. On the left, there are two code editors: one for 'k.py' and another for 'untitled1.py'. The 'untitled1.py' editor contains the following code:

```

1  # -*- coding: utf-8 -*-
2  """
3      Created on Tue Jan 23 21:40:49 2024
4
5      @author: Lenovo
6
7
8      a=10
9      b=20
10
11     c=a+20
12
13     d=2*b
14
15     print(a,b,c,d)

```

In the center, the Variable Explorer shows the following data:

Name	Type	Size	Value
a	int	1	10
b	int	1	20
c	int	1	30
d	int	1	40
r1	tuple	3	(5, 1, 6)

On the right, the IPython Console displays the output of the script:

```

In [4]: runfile('C:/Users/Lenovo/Desktop/py for ds/untitled1.py', wdir='C:/Users/Lenovo/Desktop/py for ds')
10 20 30 40

In [5]: 

```

I have tried various code editors such as VScode for using python Libraries.

The screenshot shows the VSCode interface. The Explorer sidebar shows a project structure with files like 'salesprediction.py', 'Sales.csv', and 'salesprediction.py'. The main editor shows the following code:

```

import pandas as pd
dataset=pd.read_csv('Sales.csv')
print(dataset)

```

The terminal window shows the output of running the script:

```

PS C:\Users\Lenovo\Desktop\pro> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python311/python.exe c:/Users/Lenovo/Desktop/pro/salesprediction.py
   TV Radio Newspaper Sales
0  230.1  37.8   69.2  22.1
1   44.5  39.3   45.1  10.4
2   17.2  45.9   69.3  12.0
3  151.5  41.3   58.5  16.5
4  188.8  10.8   58.4  17.9
195 38.2   3.7   13.9   7.6
196 94.2   4.9   8.1    14.0
197 177.0   9.3   6.4    14.8
198 283.6  42.0   66.2   25.5
199 232.1   8.6   8.7    18.4

[200 rows x 4 columns]
PS C:\Users\Lenovo\Desktop\pro> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python311/python.exe c:/Users/Lenovo/Desktop/pro/salesprediction.py
   TV Radio Newspaper Sales
0  230.1  37.8   69.2  22.1
1   44.5  39.3   45.1  10.4
2   17.2  45.9   69.3  12.0
3  151.5  41.3   58.5  16.5
4  188.8  10.8   58.4  17.9
...   ...
195 38.2   3.7   13.8   7.6
196 94.2   4.9   8.1    14.0
197 177.0   9.3   6.4    14.8
198 283.6  42.0   66.2   25.5
199 232.1   8.6   8.7    18.4

```

Pandas offers a variety of functions for Data Wrangling and manipulations.

1.2 Week 2 :

In week 2 I learned Data types, Conditional Statements, Looping Statements in Python. I also worked on these concepts.

Data types in python:

Commonly used data types in python are Boolean, Integer, Float, Complex and String.

I have Used Spyder IDE for practicing my code.

The screenshot shows the Spyder IDE interface. On the left, the code editor displays `ex1.py` with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun Jan 28 14:20:08 2024
4
5 @author: Lenovo
6 """
7
8 a=10
9 print(type(a))
10
11 b=10.2
12 print(type((b)))
13
14 c=True
15 print(type(c))
16
17 d="python"
18 print(type(d))
```

On the right, the IPython Console shows the output of running the code:

```
In [1]: runfile('C:/Users/Lenovo/untitled0.py', wdir='C:/Users/Lenovo')
<class 'int'>
<class 'float'>
<class 'bool'>
<class 'str'>
```

The screenshot shows the Spyder IDE interface. On the left, the code editor displays `ex2.py` with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun Jan 28 14:26:46 2024
4
5 @author: Lenovo
6 """
7
8 #convert the datatype of an object into another datatype
9
10 a=10
11 print(float(a))
12
13 b='12'
14 print(int(b))
15
16 c=10.34
17 print(str(c))
18
19
```

On the right, the IPython Console shows the output of running the code:

```
In [4]: runfile('C:/Users/Lenovo/ex2.py', wdir='C:/Users/Lenovo')
10.0
12
10.34
In [5]:
```

Conditional Statements in python:

Conditional statements are fundamental programming constructs that allow you to control the flow of your program based on conditions that you specify.

Types of Conditional Statements in python:

1. if statement
2. if...else statement
3. if...elif...else statement
4. Nested if statement

if statement :

The screenshot shows the Spyder Python IDE interface. The code editor on the left contains a file named ex3.py with the following content:

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 28 15:15:54 2024
@author: Lenovo
"""

number=int(input("enter a number:"))
if number>0:
    print("positive")
```

The code is run in the IPython Console at the bottom, which outputs:

```
In [6]: runfile('C:/Users/Lenovo/ex3.py', wdir='C:/Users/Lenovo')
enter a number:10
positive
```

The status bar at the bottom indicates the Python version is 3.8.10 and the current temperature is 32°C.

if...else..statement :

The screenshot shows the Spyder IDE interface. The left pane displays a code editor with three tabs: ex1.py, ex2.py, and ex3.py. The ex3.py tab is active, containing the following Python code:

```
1 # -*- coding: utf-8 -*-
"""
Created on Sun Jan 28 15:15:54 2024
@author: Lenovo
"""

number=int(input("enter a number:"))
if number>0:
    print("positive")
else:
    print("negative")
```

The right pane features a variable explorer, a plots section, and an IPython console. The IPython console shows the output of running the script:

```
In [8]: runfile('C:/Users/Lenovo/ex3.py', wdir='C:/Users/Lenovo')
enter a number:-8
negative

In [9]:
```

if...elif...else...statement :

The screenshot shows the Spyder IDE interface. The left pane displays a code editor with three tabs: ex1.py, ex2.py, and ex3.py. The ex3.py tab is active, containing the following Python code:

```
1 # -*- coding: utf-8 -*-
"""
Created on Sun Jan 28 15:15:54 2024
@author: Lenovo
"""

number=int(input("enter a number:"))
if number>0:
    print("positive")
elif number==0:
    print("zero")
else:
    print("negative")
```

The right pane features a variable explorer, a plots section, and an IPython console. The IPython console shows the output of running the script:

```
In [15]: runfile('C:/Users/Lenovo/ex3.py', wdir='C:/Users/Lenovo')
enter a number:0
zero

In [16]:
```

Nested if statement :

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 28 15:25:19 2024
@author: Lenovo
"""

number=int(input("enter a number:"))
if number>=0:
    if number==0:
        print("zero")
    else:
        print("positive")
else:
    print("negative")

In [17]: runfile('C:/Users/Lenovo/untitled4.py', wdir='C:/Users/Lenovo')
enter a number:0
zero

In [18]: runfile('C:/Users/Lenovo/untitled4.py', wdir='C:/Users/Lenovo')
enter a number:9
positive

In [19]: runfile('C:/Users/Lenovo/untitled4.py', wdir='C:/Users/Lenovo')
enter a number:-99
negative
```

Looping Statements:

- **for loop**
- **while loop**

for loop is designed to repeatedly execute a code block.

while loop is used to execute the statements repeatedly until a given condition is satisfied.

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 28 15:38:17 2024
@author: Lenovo
"""

#for loop
for i in range(1,11):
    print(i)

mylist=["apple","orange","Banana"]
for i in mylist:
    print(i)

In [21]: runfile('C:/Users/Lenovo/ex5.py', wdir='C:/Users/Lenovo')
1
2
3
4
5
6
7
8
9
10
apple
orange
Banana
```

The screenshot shows the Spyder IDE interface. On the left, the code editor displays a file named ex6.py with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun Jan 28 15:41:53 2024
4
5 @author: Lenovo
6 """
7
8 #while loop
9
10 number=int(input("enter a number:"))
11 while(number<15):
12     print(number)
13     number+=1
```

The right side of the interface shows the Variable Explorer, which is currently empty. Below the code editor is the IPython console, which has run the script and displayed the output of the while loop:

```
In [23]: runfile('C:/Users/Lenovo/ex6.py', wdir='C:/Users/Lenovo')
enter a number:5
5
6
7
8
9
10
11
12
13
14
15
```

The status bar at the bottom indicates the Python version is 3.8.10, and the current time is 15:44.

1.3 Week 3:

In this week, I learned python libraries such as numpy (numericalpython), pandas (panel data) and basics of python.

NUMPY:

Numpy is a fundamental package for numerical computations in python.

It supports N-dimensional array objects that can be used for processing multidimensional data.

It supports different datatypes.

Examples:

The image displays two vertically stacked instances of the Visual Studio Code (VS Code) interface, both titled "numpy course".

Top Window (VS Code):

- Explorer:** Shows "OPEN EDITORS [1 unsaved]" with "Welcome" and "numpy01.py" listed.
- Editor:** Displays the content of "numpy01.py":

```
1 #create an 1D,2D,3D array
2
3 #importing numpy library
4
5 import numpy as np
6 arr=np.array([1,2,3,4])
7 print(arr)
8 print(arr.size)
9 print(arr.shape)
10 print(arr.ndim)
11 print(type(arr))
```

- Terminal:** Shows the output of running the script:
PS C:\Users\Lenovo\Desktop\numpy course> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/Lenovo/Desktop/numpy course/numpy01.py"
[1 2 3 4]
- Status Bar:** Shows "Ln 11, Col 15" and "Python 3.11.4 64-bit".

Bottom Window (VS Code):

- Explorer:** Shows "OPEN EDITORS [1 unsaved]" with "Welcome", "numpy01.py", and "numpy02.py" listed.
- Editor:** Displays the content of "numpy02.py":

```
1 #2D array
2
3 import numpy as np
4
5 arr=np.array([[1,2,3,4],[4,5,6,7]])
6 print(arr)
7 print(arr.size)
8 print(arr.shape)
9 print(arr.ndim)
10 print(arr[0])
11 print(type(arr))
12 print(arr[0][1])
```

- Terminal:** Shows the output of running the script:
PS C:\Users\Lenovo\Desktop\numpy course> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/Lenovo/Desktop/numpy course/numpy02.py"
[[1 2 3 4]
 [4 5 6 7]]
8
(2, 4)
2
[1 2 3 4]
<class 'numpy.ndarray'>
2
- Status Bar:** Shows "Ln 12, Col 17" and "Python 3.11.4 64-bit".

The image shows two instances of the Visual Studio Code (VS Code) interface side-by-side, both displaying Python code in the Explorer tab.

Left Editor (Top):

- File Path:** C:\Users\Lenovo\Desktop\numpy course\numpy03.py
- Code:**

```
1 #3D array
2
3 import numpy as np
4 arr=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
5 print(arr)
6 print(arr.shape)
7 print(arr.ndim)
8 print(arr[0][1])
9 print(type(arr))
```

Right Editor (Top):

- File Path:** C:\Users\Lenovo\Desktop\numpy course\zeros_array.py
- Code:**

```
1 #create an 1D zeros array
2
3 import numpy as np
4
5 arr=np.zeros(3)
6 print(arr)
7 print(arr.ndim)
8 print(arr.shape)
9 print(arr.size)
10
11 #create an 2D zeros array
12
13 import numpy as np
14
15 arr1=np.zeros((3,4),dtype=int)
16 print(arr1)
17 print(arr1.shape)
18 print(arr1.ndim)
```

Common UI Elements:

- Top Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** numpy course
- Terminal:** powershell, Python
- Status Bar:** Ln 4, Col 22 (4 selected), Spaces: 4, UTF-8, CRLF, Python 3.11.4 64-bit, Go Live, Prettier

The screenshot shows the Visual Studio Code interface with the title bar "numpy course". The Explorer sidebar on the left lists files: Welcome, numpy01.py, numpy02.py, numpy03.py, x.py, zeros_array.py (which is open), and NUMPY COURSE (containing numpy01.py, numpy02.py, numpy03.py, x.py). The code editor displays the following Python script:

```
11 #create an 2D zeros array
12
13 import numpy as np
14
15 arr1=np.zeros((3,4),dtype=int)
16 print(arr1)
17 print(arr1.shape)
18 print(arr1.ndim)
19
20 #create an 3D array
21
22 import numpy as np
23
24 arr2=np.zeros((2,3,4),dtype=int)
25 print(arr2)
26 print(arr2.ndim)
27 print(arr2.shape)
28 print(arr2.size)
29
```

The screenshot shows the Visual Studio Code interface with the title bar "numpy course". The Explorer sidebar on the left lists files: Welcome, numpy01.py, numpy02.py, numpy03.py, zeros_array.py, ones_array.py (which is open), and x.py. The code editor displays the following Python script:

```
1 #creating an 10,20,30 ones array
2
3 import numpy as np
4 arr1=np.ones(3,dtype=str)
5 print(arr1)
6
7 arr2=np.ones((3,4),dtype=int)
8 print(arr2)
9
10 arr3=np.ones((2,3,4))
11 print(arr3)
12 print(arr3.shape)
```

The terminal output shows the execution of the script:

```
PS C:\Users\Lenovo\Desktop\numpy course> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/Lenovo/Desktop/numpy course/ones_array.py"
[[1.  1.  1.]
 [1.  1.  1.]
 [1.  1.  1.]]
(3, 4)
PS C:\Users\Lenovo\Desktop\numpy course>
```

Screenshot of VS Code showing the code for `full_array.py`. The code creates a 3D full array filled with the value 'abdu1'.

```
1 #create an 1D,2D,3D full array
2
3 import numpy as np
4
5 arr=np.full(3,'abdu1')
6 print(arr)
7
8 arr1=np.full((3,4),"abdu1")
9 print(arr1)
10
11 arr2=np.full((3,4,5),"abdu1")
12 print(arr2)
13
```

The terminal output shows the resulting 3D array:

```
[['abdu1' 'abdu1' 'abdu1' 'abdu1']
[['abdu1' 'abdu1' 'abdu1' 'abdu1']
 ['abdu1' 'abdu1' 'abdu1' 'abdu1']
 ['abdu1' 'abdu1' 'abdu1' 'abdu1']]
[[['abdu1' 'abdu1' 'abdu1' 'abdu1']
 ['abdu1' 'abdu1' 'abdu1' 'abdu1']
 ['abdu1' 'abdu1' 'abdu1' 'abdu1']
 ['abdu1' 'abdu1' 'abdu1' 'abdu1']]
```

[[[["abdu1" "abdu1" "abdu1" "abdu1"]
 ["abdu1" "abdu1" "abdu1" "abdu1"]
 ["abdu1" "abdu1" "abdu1" "abdu1"]
 ["abdu1" "abdu1" "abdu1" "abdu1"]]

[[["abdu1" "abdu1" "abdu1" "abdu1"]
 ["abdu1" "abdu1" "abdu1" "abdu1"]
 ["abdu1" "abdu1" "abdu1" "abdu1"]
 ["abdu1" "abdu1" "abdu1" "abdu1"]]]

Screenshot of VS Code showing the code for `choice_Function.py`. The code demonstrates various ways to use the `np.random.choice` function.

```
1 import numpy as np
2 from numpy import random
3
4 arr=np.random.choice(12,123)
5 print(arr)
6
7 arr1=np.random.choice([1,2,3,4,5,6,7])
8 print(arr1)
9
10 arr2=np.random.choice(19)
11 print(arr2)
12
13 arr3=np.random.choice([1,3,5,7,9],size=5,replace=True)
14 print(arr3)
15
16 arr4=np.random.choice([1,2,3,4,5,6],size=4,replace=False)
17 print(arr4)
```

The terminal output shows the resulting arrays:

```
[ 5  9  2  5  7 11 10  7  6 11  9 10  0  3  1  4  2  6  5  5  7  1  9  8
 0  7  5  8  4  1 11  6 11  7  5  5  6  9 11 10 11  5  9  5  3 10  1 11
 3  2 10  6  6  2  7  0 11  7  2  8  9  7  7  4  7  5  8  8 10  3  3  1
 8  0  0  5  1  9  1  4 11  1  6  5  7  9  4 11  4  9  3  1  5  0  4  8
 8 11  4  5  6  3  5  3  7  5  6  1  4 10 11  9  2  6  5  9  7 10 10  6
 6 10 10]
3
14
[1 3 7 7 1]
[3 5 4 2]
PS C:\Users\Lenovo\Desktop\numpy course>
```

The image displays two side-by-side screenshots of a code editor interface, likely Visual Studio Code, showing Python code related to NumPy functions.

Top Window (Screenshot 1):

- File Explorer:** Shows files in the current workspace, including `numpy_functions.py`, `numpy01.py`, `numpy02.py`, `numpy03.py`, `zeros_array.py`, `ones_array.py`, `full_array.py`, `choice_Function.py`, and `numpy_course`.
- Code Editor:** Displays the contents of `numpy_functions.py`. The code uses `np.random.randint` to create a 2x30 array, reshapes it to 3x5, and prints specific elements.
- Terminal:** Shows the output of running the script, displaying a 3x5 matrix of integers.
- Status Bar:** Shows the file path as "C:\Users\Lenovo\Desktop\numpy_course", the command as "python.exe", and the Python version as "3.11.4 64-bit".

Bottom Window (Screenshot 2):

- File Explorer:** Shows files in the current workspace, including `numpy03.py`, `zeros_array.py`, `ones_array.py`, `full_array.py`, `choice_Function.py`, `numpy_functions.py`, and `numpy_functions02.py`.
- Code Editor:** Displays the contents of `numpy_functions02.py`. The code uses `np.random.choice` to create a 3x20 array, reshapes it to 4x5, and calculates the sum along axis 1 and axis 0.
- Terminal:** Shows the output of running the script, displaying a 4x5 matrix of integers and the resulting sums for each axis.
- Status Bar:** Shows the file path as "C:\Users\Lenovo\Desktop\numpy_course", the command as "python.exe", and the Python version as "3.11.4 64-bit".

Top Window (numpy course):

```

1 #replacing elements in numpy array
2
3 import numpy as np
4 arr=np.full((3,3),10)
5 print(arr)
6
7 #replacing all elements
8
9 arr1=arr[:,::]-50
10 print(arr1)
11
12 #replacing specific elements
13
14 arr2=arr[0,1]-500
15 print(arr2)

```

Bottom Window (matrix_operations.py):

```

1 import numpy as np
2 M=np.full((3,3),7)
3 N=np.full((3,3),8)
4 print(M*N)
5 print(M*N)
6 print([np.dot(M,N)])

```

The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** numpy course
- Explorer:** Shows a tree view of files under "OPEN EDITORS" and "NUMPY COURSE".
- Editor:** The main area displays Python code for performing matrix operations using NumPy.
- Terminal:** The terminal output shows the execution results of the code.
- Bottom Status Bar:** Includes file path (PS C:\Users\Lenovo\Desktop\numpy_course>), line and column (Ln 11, Col 24), spaces (Spaces: 4), encoding (UTF-8), and other system information (Python 3.11.4 64-bit, 32°C, 20:04).

```
numpy_functions0.py > ...
1 import numpy as np
2 A=np.array([[1,2,3],[4,5,6],[7,8,9]])
3 print(A)
4 print(A.T)
5 #rank of a matrix
6 from numpy import linalg as lng
7 B=np.array([[1,2],[3,4]])
8 print(np.linalg.matrix_rank(B))
9 print(np.linalg.det(B))
10 print(np.trace(B))
11 print(np.linalg.inv(B))
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
 [[1 4 7]
 [2 5 8]
 [3 6 9]]
 2
 -2.000000000000004
 5
 [[-2.  1. ]
 [ 1.5 -0.5]]
```

PANDAS :

Pandas is a python library used for working with Datasets. It has functions for analyzing, cleaning, exploring, and manipulating data.

The screenshot displays two separate instances of the Visual Studio Code (VS Code) interface, both showing the same Python script named `pandas02.py`. The script contains the following code:

```
import pandas as pd
import numpy as np
data={'a':np.random.randint(10,20,5),'b':np.random.randint(10,30,5)}
df=pd.DataFrame(data)
print(df)
#print(df['a'],df[['a','b']])
print(df.mean())
print(df.mean(axis=1))
```

The top instance of VS Code shows the output of the script as float64 data:

	a	b
0	13	17
1	13	12
2	18	25
3	16	22
4	18	10

The bottom instance of VS Code shows the output of the script as int64 data:

	a	b
0	11	25
1	11	15
2	12	29
3	10	11
4	15	28

Both instances of VS Code have identical terminal outputs:

```
PS C:\Users\Lenovo\Desktop\numpy_course> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/Lenovo/Desktop/numpy_course/pandas02.py"
          a   b
0    11  25
1    11  15
2    12  29
3    10  11
4    15  28
a    59
b   108
dtype: int64
0    36
1    26
2    41
3    21
4    43
dtype: int64
```

The screenshot shows the Visual Studio Code (VS Code) interface. The title bar reads "pandas course". The left sidebar has sections for "OPEN EDITORS" and "PANDAS COURSE", both listing files like "pandas01.py" through "pandas08.py" and CSV files "abc.csv", "birth.csv", "research.csv", and "salary.csv". The main editor area displays the content of "pandas01.py":

```
1 import pandas as pd
2
3 df=pd.read_csv("abc.csv")
4
5 print(df)
6
7 print(df.head())
8
9 print(df.tail())
10
11 print(df.shape)
12
13 print(type(df))
14
15 print(df.info())
16
17 print(df.describe())
```

The "TERMINAL" tab is active, showing the output of running the script:

```
memory usage: 3.2+ MB
None
Year
count 41715.000000
mean 2017.000000
std 2.58282
min 2013.000000
25% 2015.000000
50% 2017.000000
75% 2019.000000
max 2021.000000
PS C:\Users\Lenovo\Desktop\pandas course>
```

The status bar at the bottom shows "Ln 17, Col 20" and "Python 3.11.4 64-bit".

1.4 Week 4 :

-Relation between numpy and pandas:

NumPy and pandas are two popular libraries in Python that are commonly used for data manipulation, analysis, and numerical computations. While they serve different purposes ,they are often used together in data science and analysis workflows.

2 NUMPY:

NumPy (Numerical Python) provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays. It forms the foundation for many other scientific computing libraries in Python.

File Edit Selection View Go Run ... ← → ⌘ ex2

EXPLORER OPEN EDITORS

- Welcome
- ex1.py
- ex3
- X NUM.PY
- ex2.py

OPEN EDITORS

- EX2
- ex1.py
- ex2.py
- ex3
- flavors_of_cocoa.csv
- NUM.PY
- Toyota.csv

NUM.PY > ...

```
1 import numpy as np
2
3 # Creating a NumPy array
4 arr=np.array([1, 2, 3, 4, 5])
5 print(arr)
6
7 # Performing mathematical operations
8 squared_arr=np.square(arr)
9
10 print(squared_arr)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

TERMINAL

```
PS C:\Users\Lenovo\Desktop\ex2> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python311/python.exe c:/Users/Lenovo/Desktop/ex2/NUM.PY
[ 1  4  9 16 25]
PS C:\Users\Lenovo\Desktop\ex2> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python311/python.exe c:/Users/Lenovo/Desktop/ex2/NUM.PY
[1 2 3 4 5]
[ 1  4  9 16 25]
PS C:\Users\Lenovo\Desktop\ex2>
```

Python Python Python

Ln 8, Col 13 Spaces: 4 UTF-8 CRLF Python 3.11.4 64-bit Go Live Prettier 28°C ENG 21:18

3 PANDAS:

pandas is a powerful library for data manipulation and analysis. It introduces two key data structures: Series (1-dimensional labeled array) and DataFrame (2-dimensional table). Pandas is built on top of NumPy and often used for handling and analyzing structured data.

The screenshot shows the Visual Studio Code interface with a dark theme. In the Explorer sidebar, there are several files: Welcome, ex1.py, ex3, NUM.PY, pan.py, ex2.py, ex1.py, ex2.py, ex3, flavors_of_cocoa.csv, NUM.PY, pan.py, and Toyota.csv. The file pan.py is currently open in the editor. The code in pan.py is:

```
1 import pandas as pd
2
3 # Creating a pandas Series
4 data=pd.Series([10, 20, 30, 40, 50], index=['a', 'b', 'c', 'd', 'e'])
5
6 print(data)
7
8 # Accessing elements and performing operations
9 doubled_data=data * 2
10
11 print(doubled_data)
```

The terminal below the editor shows the output of the script:

```
PS C:\Users\Lenovo\Desktop\ex2> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python311/python.exe c:/Users/Lenovo/Desktop/ex2/pan.py
a    10
b    20
c    30
d    40
e    50
dtype: int64
a    20
b    40
c    60
d    80
e   100
dtype: int64
```

The status bar at the bottom indicates the script was run at 21:22.

Pandas can also work seamlessly with NumPy arrays. Forexample, you can create a DataFrame using a NumPy array:

The screenshot shows the Visual Studio Code interface with a dark theme. In the Explorer sidebar, there are several files: Welcome, ex1.py, ex3, NUM.PY, pan.py, ex2.py, ex1.py, ex2.py, ex3, flavors_of_cocoa.csv, NUM.PY, pan.py, pan2.py, and Toyota.csv. The file pan2.py is currently open in the editor. The code in pan2.py is:

```
1 import pandas as pd
2 import numpy as np
3
4 # Creating a DataFrame from a NumPy array
5 data=np.array([[1, 2, 3], [4, 5, 6]])
6 df=pd.DataFrame(data, columns=['A', 'B', 'C'])
7
8 print(df)
```

The terminal below the editor shows the output of the script:

```
PS C:\Users\Lenovo\Desktop\ex2> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python311/python.exe c:/Users/Lenovo/Desktop/ex2/pan2.py
   A   B   C
0  1   2   3
1  4   5   6
PS C:\Users\Lenovo\Desktop\ex2>
```

The status bar at the bottom indicates the script was run at 21:25.

In the example above, pandas DataFrame df is created using a NumPy array, showcasing the interoperability between the two libraries. NumPy provides the underlying numerical operations, while pandas offers high-level data manipulation capabilities, making the powerful combination in data analysis workflows.

2 Introduction

2.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various **Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end etc.**



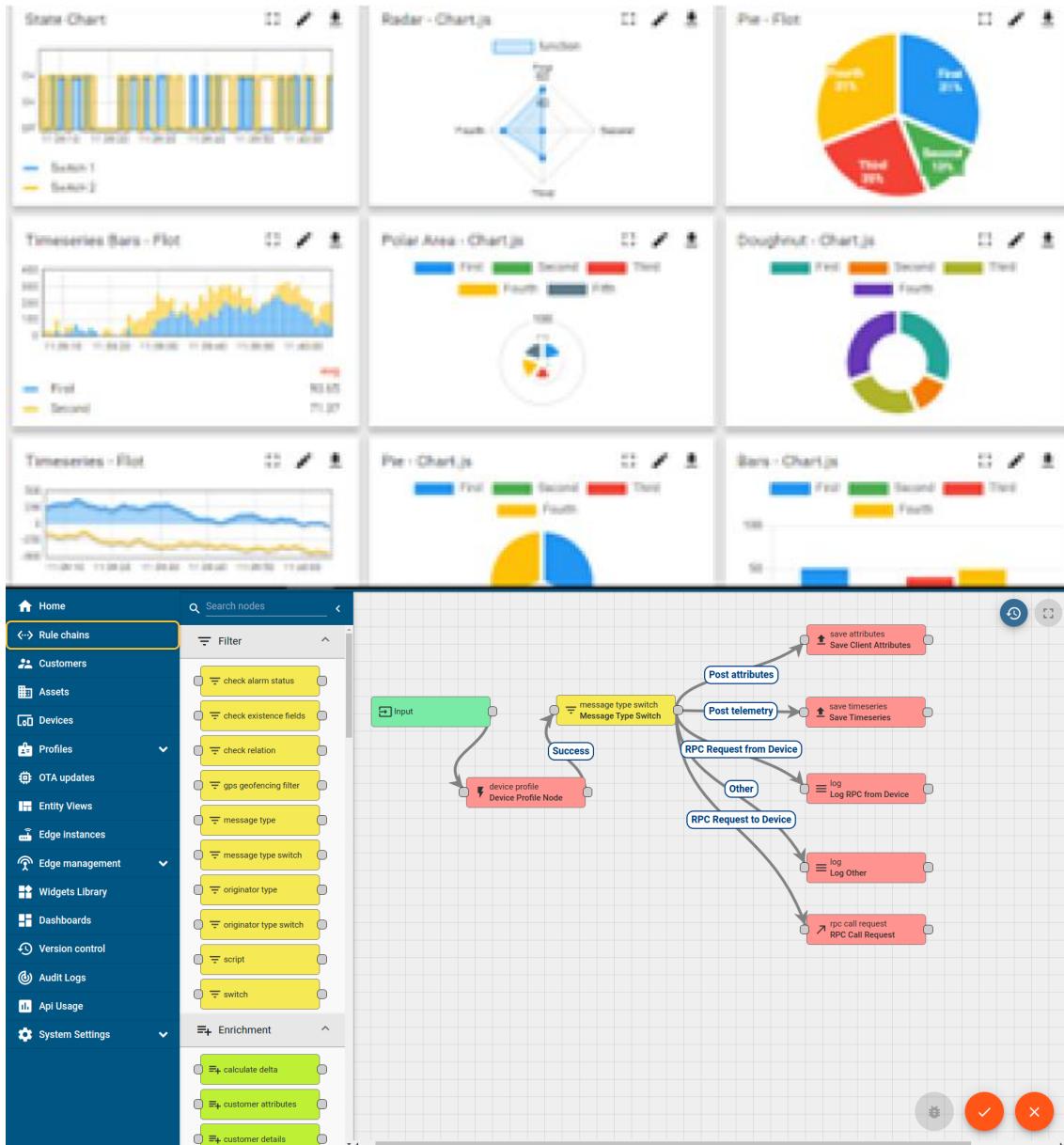
i. UCT IoT Platform([UCT Insight](#))

UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
- It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine



FACTORY WATCH

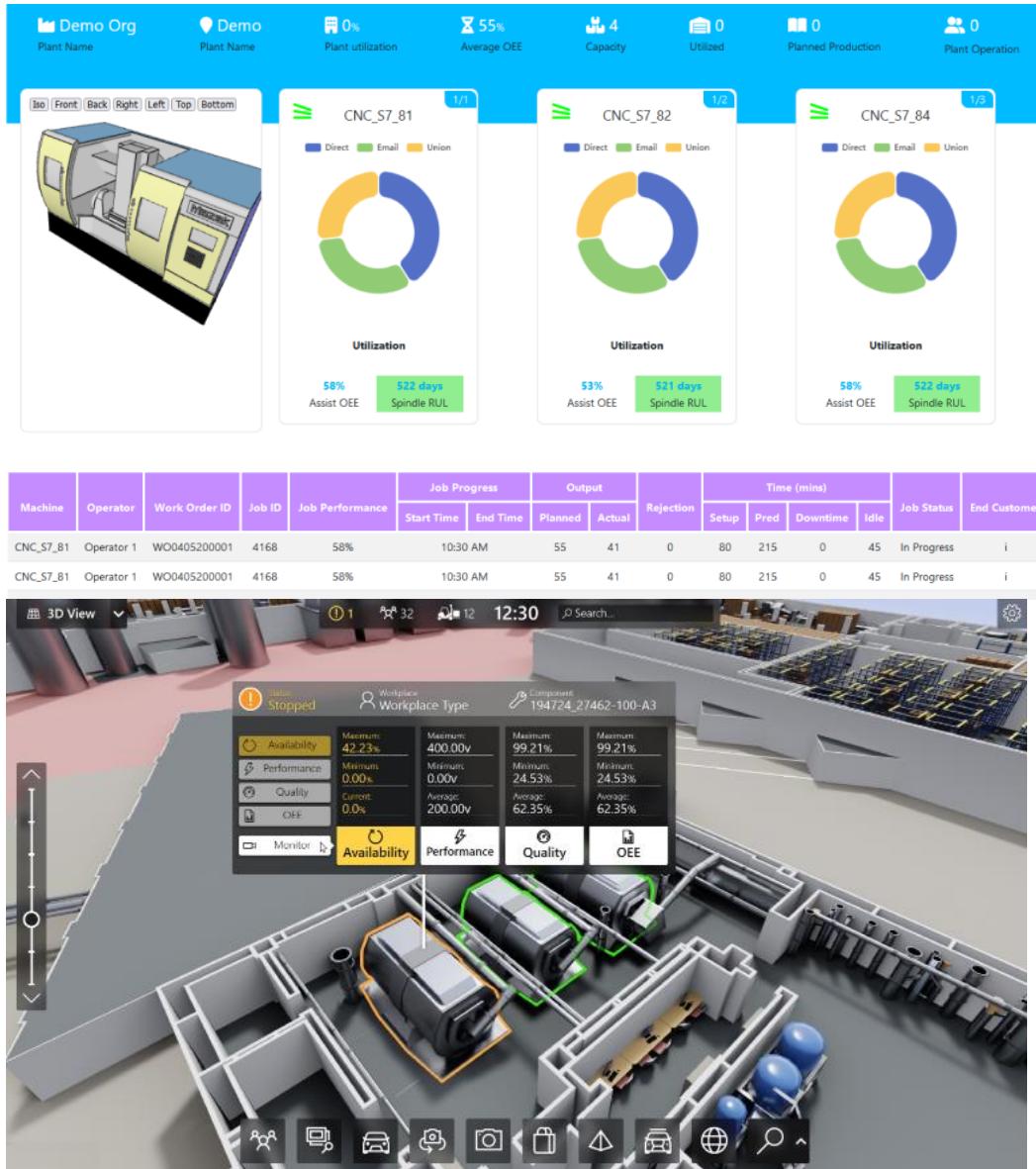
ii. Smart Factory Platform (FACTORY WATCH)

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleashed the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.

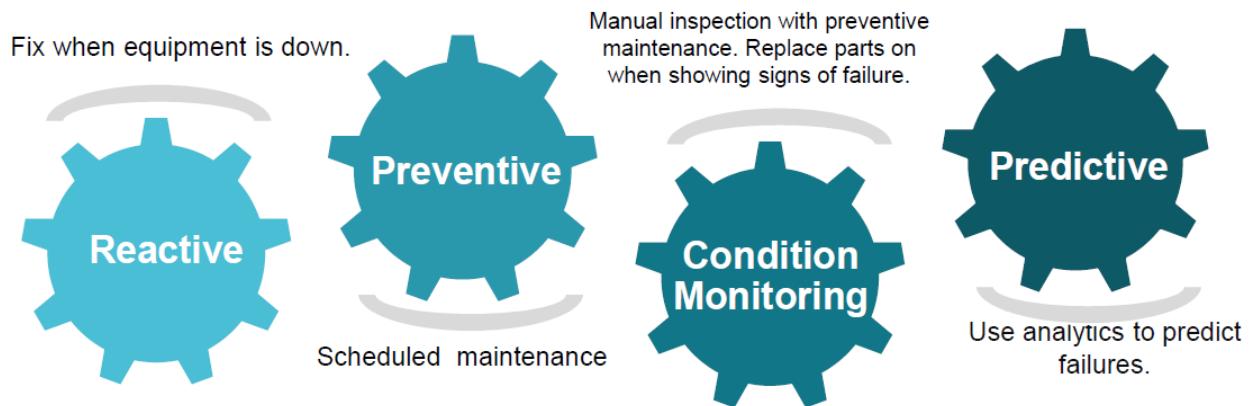


based Solution

UCT is one of the early adopters of LoRAWAN technology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



2.1 About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



2.2 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.3 Objectives of this Internship program

The objective for this internship program was to

- ☛ get practical experience of working in the industry.
- ☛ to solve real world problems.
- ☛ to have improved job prospects.
- ☛ to have Improved understanding of our field and its applications.
- ☛ to have Personal growth like better communication and problem solving.

3 MY PROJECT : URL SHORTENER

First of all, let us understand what is a URL shortener. Users have relied on connections as their primary form of transit since the internet's inception. URL shortening is a Web method for making a Uniform Resource Locator (URL) significantly shorter while still directing to the desired website. This is accomplished by the use of a redirect, which points to a web page with a lengthy URL. They let users navigate from one website to another in a matter of seconds by just clicking on a URL. Unfortunately, connections typically become lengthy and complex when tracking parameters are added and site structures expand to handle hundreds of pages. Long URLs became a concern as social media grew in popularity. Twitter used to limit messages to a maximum of 140 characters, and every character in a link was tallied. As a result, a long URL may take up the entirety of your tweet. URL shortener programs have evolved as a response to this difficulty of sharing.

3.1 Benefits of URL shortener

- The importance of content in any marketing plan cannot be overstated. You'll need the correct URLs to distribute that material. An URL shortener guarantees that the relevant messages reach your target audience without taking up too much room in your social media postings.
- Make it simpler for people to share your material by doing the following: Customers may learn everything they need to know about your site by using simple and branded URLs. Reduced URLs with random letter and number combinations are no longer necessary.
- Make your URLs more attractive: Shorter URLs are more appealing from an aesthetic standpoint. Although it may not appear to be significant, a shorter URL may be the deciding factor in encouraging someone to click on your links.
- Allows for traffic monitoring: Bit.ly includes tracking monitors that track your tweet or post's sharing activity over time.

4 PROJECT IMPLEMENTATION :

```
#pip install pyshorteners
```

```
from tkinter import*
```

```
import pyshorteners
```

```
root= Tk()
```

```
root.title("URL Shortener link")
```

```
root.geometry("800x300")
```

```
def myUrl():
```

```
    url_entry=url.get()
```

```
    result=pyshorteners.Shortener().tinyurl.short(url_entry)
```

```
    urlEntry.insert(END,result)
```

```
Label(root, text="Generate Short URL",font=("Georgia 25 bold"), fg="Purple").pack(pady=10)
```

```
frame1=Frame(root)
```

```
Label(frame1, text="Paste URL here: ", font=("Georgia 15 bold")).pack(side=LEFT)
```

```
url = Entry(frame1, width="40", font=("Georgia 15 bold"))
```

```
url.pack()
```

```
[  
  
frame1.pack(pady=10)  
  
  
Button(root , text="Gererate Link", font=("Georgia 15 bold"), command=myUrl).pack(pady=10)  
  
  
frame2=Frame(root)  
Label(frame2, text="Copy URL: ", font=("Georgia 15 bold")).pack(side=LEFT)  
  
  
urlEntry=Entry(frame2,width="25",fg="blue",font=("Georgia 15 bold"))  
urlEntry.pack()  
frame2.pack(pady=10)  
  
  
root.mainloop()
```

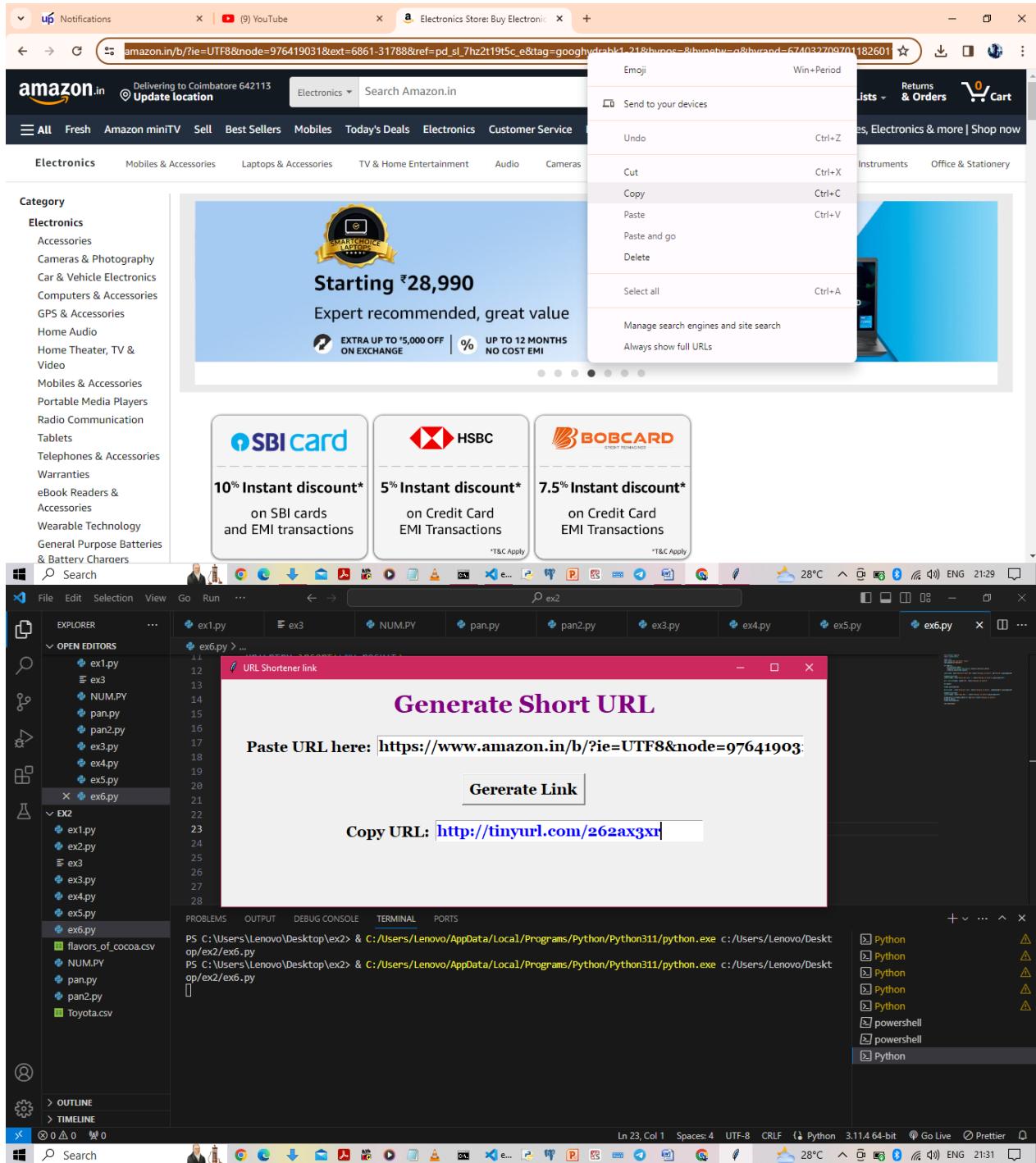
The screenshot shows a code editor interface with two windows side-by-side, both displaying the same Python script. The script uses the Tkinter library to create a simple GUI for generating short URLs.

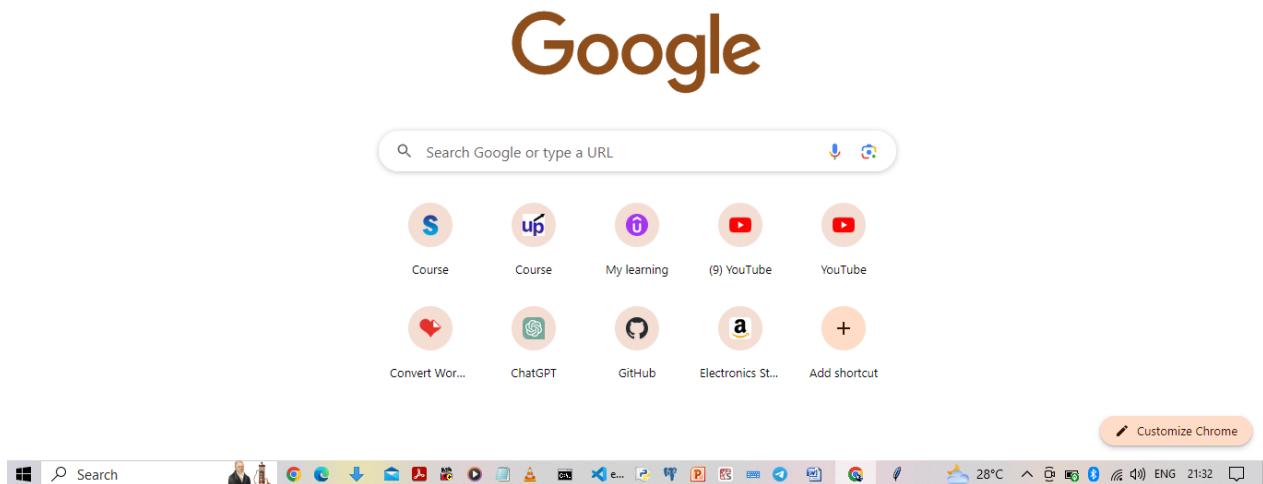
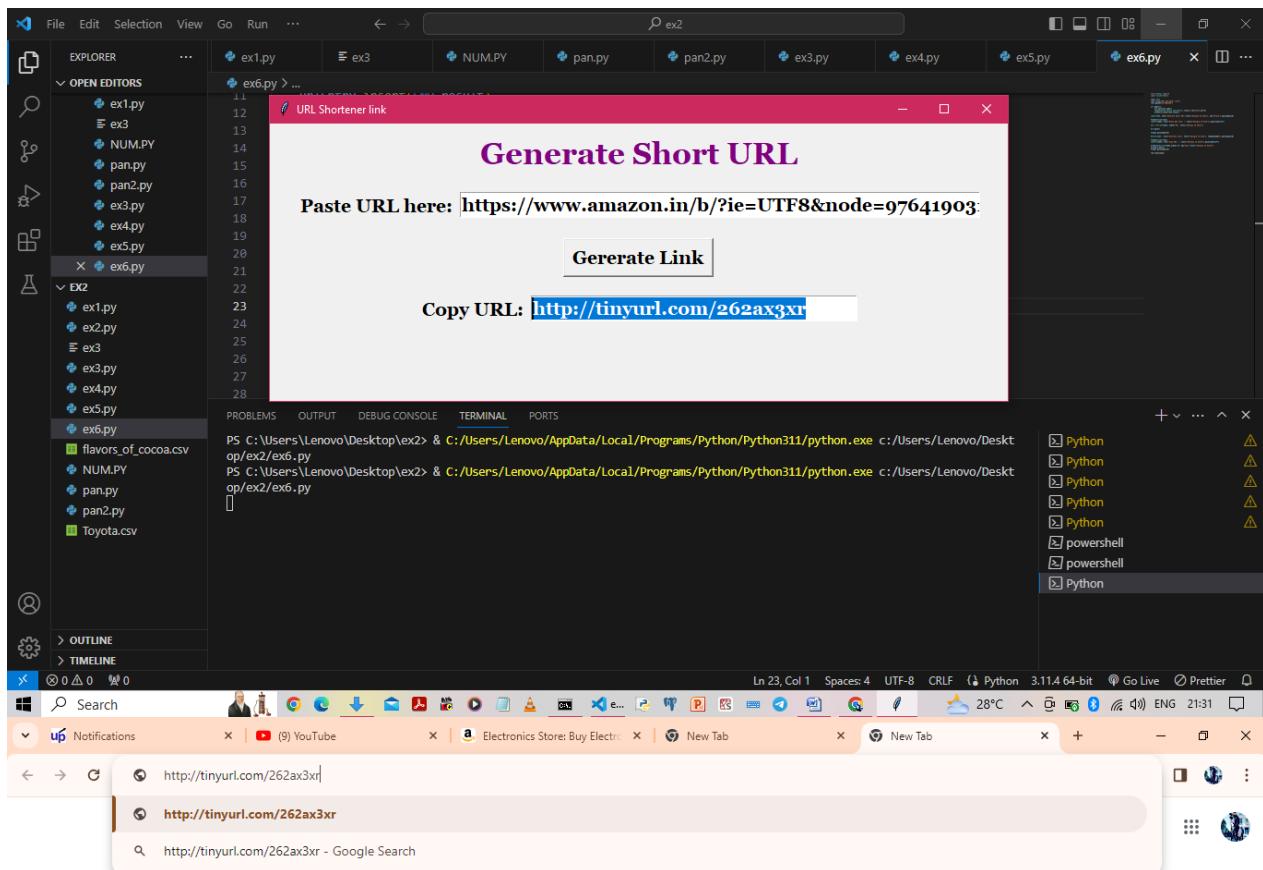
```

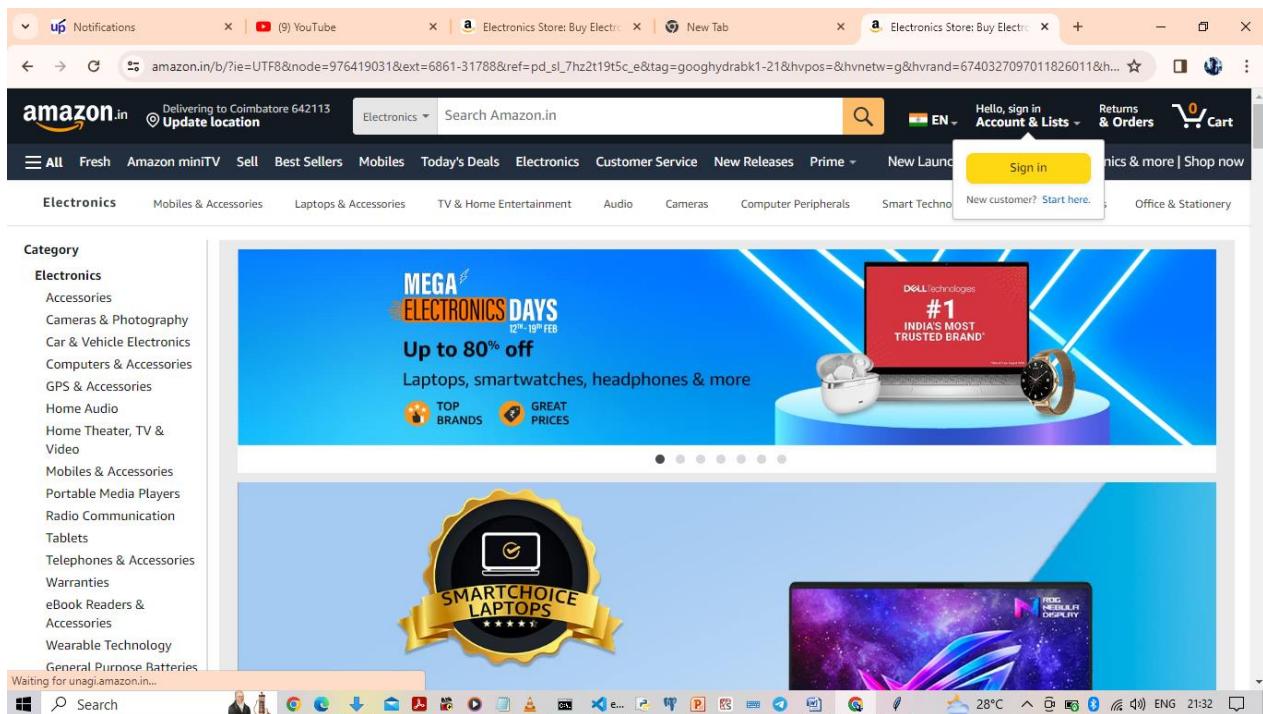
1  from tkinter import*
2  import pyshorteners
3
4  root= Tk()
5  root.title("URL Shortener link")
6  root.geometry("800x300")
7
8  def myUrl():
9      url_entry=url.get()
10     result=pyshorteners.Shortener().tinyurl.short(url_entry)
11     urlEntry.insert(END,result)
12
13 Label(root, text="Generate Short URL",font=("Georgia 25 bold"), fg="Purple").pack(pady=10)
14
15 frame1=Frame(root)
16 Label(frame1, text="Paste URL here: ", font=("Georgia 15 bold")).pack(side=LEFT)
17
18 url = Entry(frame1, width="40", font=("Georgia 15 bold"))
19
20 url.pack()
21
22 frame1.pack(pady=10)
23
24 Button(root , text="Generate Link", font=("Georgia 15 bold"), command=myUrl).pack(pady=10)
25
26 frame2=Frame(root)
27 Label(frame2, text="Copy URL: ", font=("Georgia 15 bold")).pack(side=LEFT)
28
29 urlEntry=Entry(frame2,width="25",fg="blue",font=("Georgia 15 bold"))
30 urlEntry.pack()
31 frame2.pack(pady=10)
32
33 root.mainloop()

```

The code defines a function `myUrl()` that takes a URL from an entry field, shortens it using `pyshorteners`, and inserts the result back into the entry field. It creates a window with a title bar, a main frame, and a sub-frame for the URL entry and button. The button calls the `myUrl()` function when clicked.







5 MY Learnings and Outcomes :

Strengthened understanding of basic programming concepts like variables, data types, loops, and conditional statements. Learned how to structure and organize code for readability and maintainability. Improved proficiency in Python syntax and language features. Acquired knowledge about Python's data structures, including lists, tuples, dictionaries, and sets. Developed skills in debugging Python code and identifying common errors. Gained problem-solving techniques to overcome challenges in coding. Gained hands-on experience working on real-world projects, possibly contributing to the development or enhancement of existing applications.

5.1 Code submission ([Github link](#))

5.2 Report submission ([Github link](#)) : first make placeholder, copy the link

