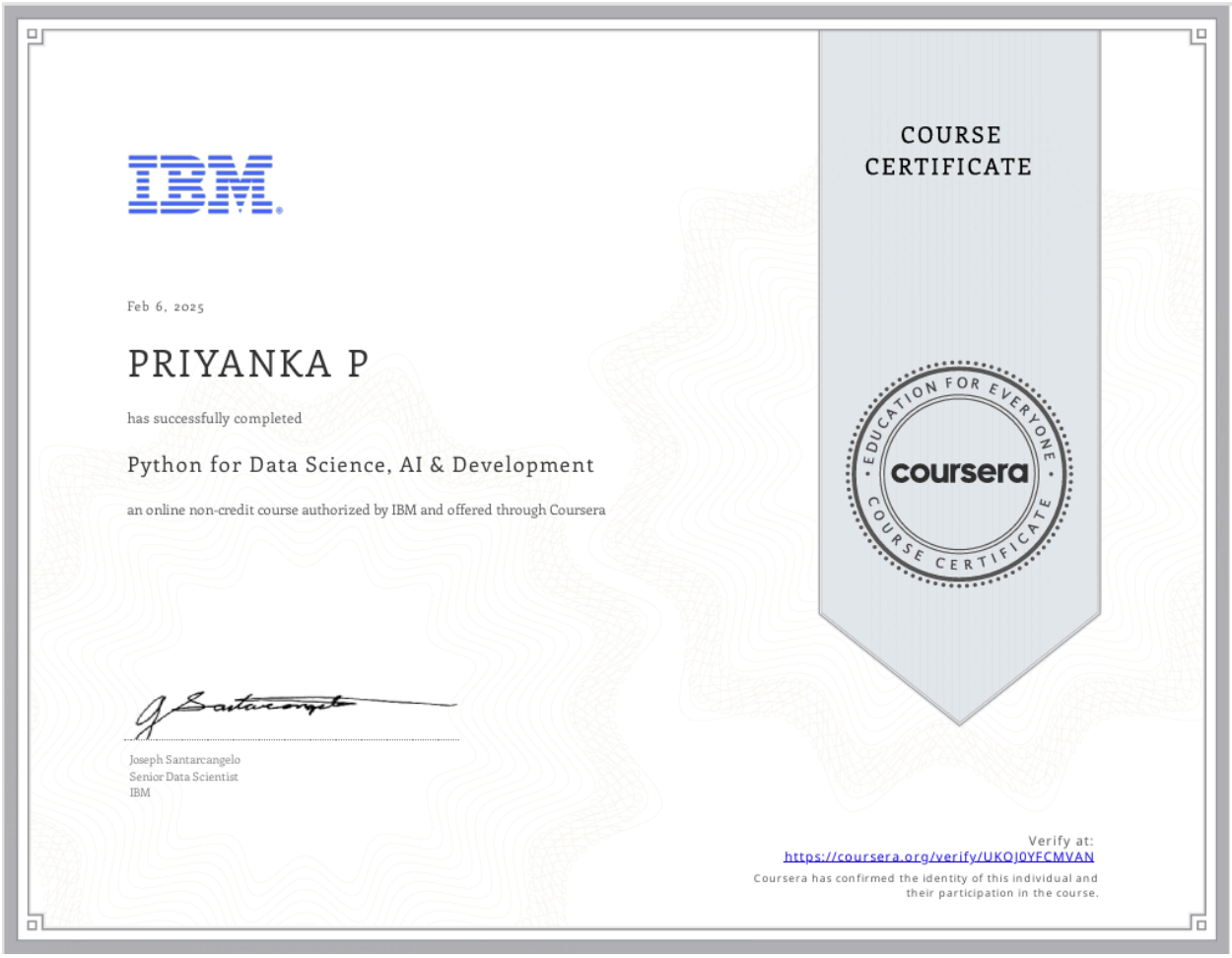

Python for Data Science, AI & Development

Module 1: Introduction to Python

What is Python?

- Python is a high-level, interpreted programming language.
- Used for web development, data science, automation, and more.
- Known for its simplicity and readability.

Key Features

- Interpreted: No need for compilation.
- Dynamically Typed: No need to declare variable types.

- Object-Oriented: Supports classes and objects.
- Extensive Libraries: Built-in modules for various tasks.

Basic Syntax

```
# Printing output
print("Hello, World!")

# Variables and Data Types
x = 10        # Integer
y = 3.14      # Float
name = "John" # String
is_valid = True # Boolean
```

Module 2: Data Structures

Lists

- Ordered, mutable collection.

```
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")  # Add element
fruits.remove("banana")   # Remove element
print(fruits[0])          # Access element
```

Tuples

- Ordered, immutable collection.

```
tuple_example = (1, 2, 3)
print(tuple_example[1])
```

Dictionaries

- Key-value pairs.

```
person = {"name": "Alice", "age": 25}
print(person["name"])
person["city"] = "New York"  # Add new key-value pair
```

Sets

- Unordered, unique collection.

```
set_example = {1, 2, 3, 3, 4}
set_example.add(5)
print(set_example)
```

Module 3: Control Flow & Functions

Conditional Statements

```
age = 18
if age >= 18:
    print("Adult")
else:
    print("Minor")
```

Loops

- For Loop

```
for i in range(5):
    print(i)
```

- While Loop

```
x = 0
while x < 5:
    print(x)
    x += 1
```

Functions

- Reusable code blocks.

```
def greet(name):
    return "Hello " + name
print(greet("Alice"))
```

Exception Handling

- Handling runtime errors.

```
try:
   result = 10 / 0
except ZeroDivisionError:
   print("Cannot divide by zero")
```

Module 4: File Handling & Data Processing

Reading Files

```
with open("file.txt", "r") as file:
   content = file.read()
print(content)
```

Writing to Files

```
with open("file.txt", "w") as file:
   file.write("Hello, World!")
```

Pandas for Data Handling

- Reading CSV

```
import pandas as pd
df = pd.read_csv("data.csv")
print(df.head())
```

- Writing CSV

```
df.to_csv("output.csv", index=False)
```

Module 5: APIs & Web Scraping

REST APIs

- APIs allow applications to communicate over the web.
- Uses HTTP methods: GET, POST, PUT, DELETE.

Using Python Requests Library

```
import requests
response = requests.get("https://api.example.com/data")
print(response.json())
```

Web Scraping with BeautifulSoup

```
from bs4 import BeautifulSoup
html = "<html><body><h1>Hello</h1></body></html>"
soup = BeautifulSoup(html, "html.parser")
print(soup.h1.text)
```

Module 6: NumPy & Data Science

NumPy Arrays

- Faster than lists, used for numerical computing.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr.mean())
```

2D Arrays

```
matrix = np.array([[1, 2, 3], [4, 5, 6]])
print(matrix[0, 1])
```

Module 7: Object-Oriented Programming (OOP)

Classes & Objects

```
class Car:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def display(self):
        print(self.brand, self.model)

car1 = Car("Toyota", "Corolla")
car1.display()
```

Module 8: Working with Different File Formats

CSV Files

```
df = pd.read_csv("data.csv")
```

JSON Files

```
import json
with open("data.json") as f:
    data = json.load(f)
print(data)
```

XML Files

```
import xml.etree.ElementTree as ET
tree = ET.parse("data.xml")
root = tree.getroot()
print(root.tag)
```