SOC Midterm Report

Anshika Yadav

Project Code: 41: Facial Recognition

June 27, 2025

Contents

1 variables.py

This script demonstrates variable declaration, printing, input, typecasting, multiple assignment, and various swapping techniques.

```python
name = "Anshika"
age = 21
height = 5.4
is_student = True

print("Name:", name)
print("Age:", age)
print("Height:", height)
print("Student:", is_student)

your_name = input("Enter your name: ")
your_age = input("Enter your age: ")  # input returns string
your_age = int(your_age)

print(f"Hello {your_name}, next year you will be {your_age + 1} years old.")

print("Type of your_name:", type(your_name))
print("Type of your_age:", type(your_age))

a, b, c = 10, 20, 30
print("Values of a, b, c:", a, b, c)

# Swapping using temp
a = 5
b = 6
temp = a
a = b
b = temp
print(a, b)

# Swapping using addition/subtraction
a = a + b
b = a - b
a = a - b
print(a, b)

# Swapping using XOR
a = a ^ b
b = a ^ b
a = a ^ b
print(a, b)

# Swapping using Pythonic way
a, b = b, a
print(a, b)
```

2 datatypes.py

Covers strings, lists, tuples, sets, and dictionaries.

```python
name = "Python Programming"
print("String:", name)
print("First 6 letters:", name[:6])
print("Uppercase:", name.upper())
print("Replaced:", name.replace("Python", "C++"))

fruits = ["apple", "banana", "mango"]
print("List:", fruits)
fruits.append("orange")
print("After appending:", fruits)
fruits.remove("banana")
print("After removing banana:", fruits)
for fruit in fruits:
    print("-", fruit)

coordinates = (10, 20)
print("Tuple:", coordinates)
print("X coordinate:", coordinates[0])

unique_numbers = {1, 2, 3, 2, 1}
print("Set:", unique_numbers)
unique_numbers.add(4)
unique_numbers.discard(2)
print("Modified Set:", unique_numbers)

student = {
    "name": "Anshika",
    "age": 21,
    "branch": "Chemical Engineering"
}
student["age"] = 22
student["college"] = "IIT Bombay"
for key, value in student.items():
    print(f"{key} : {value}")
```

3 conditionals.py

Demonstrates if-elif-else logic and nested conditions.

```python
x = 8
if x % 2 == 0:
    print("even")
if x > 5:
    print("great")
else:
    print("not great")

a = 5
if a == 1:
    print("a is 1")
elif a == 2:
    print("a is 2")
elif a == 3:
    print("a is 3")
elif a == 4:
    print("a is 4")
else:
```

```
    print("wrong input")
```

4 loops.py

Shows usage of for loops, while loops, continue/break, and nested loops.

```
x = ['anshika', 65, 2.5]
for i in x:
    print(i)

b = 'Anshika'
for i in b:
    print(i)

for i in [2,6,'laddoo']:
    print(i)

for i in range(20, 11, -2):
    print(i)

# While loop
av = 5
x = int(input("How many candies you want?"))
i = 1
while i <= x:
    if i > av:
        break
    print("candy")
    i += 1

for i in range(1, 31):
    if i % 3 == 0:
        continue
    print(i)

# Nested loops - patterns
for i in range(4):
    for j in range(4):
        print("# ", end="")
    print()

for i in range(4):
    for j in range(i+1):
        print("# ", end="")
    print()

for i in range(4):
    for j in range(4-i):
        print("# ", end="")
    print()

for i in range(4):
    for j in range(i, 4):
        print(j+1, end="")
    print()
```

5 functions.py

Explains defining functions, parameters, default args, return values, and global variables.

```python
def greet():
    print("Hello! Welcome to GitHub.")
greet()

def add(a, b):
    result = a + b
    print(f"Sum of {a} and {b} is {result}")
add(3, 7)

def multiply(x, y):
    return x * y
product = multiply(4, 5)
print("Multiplication Result:", product)

def greet_user(name="Guest"):
    print(f"Hello, {name}!")
greet_user("Anshika")
greet_user()

total = 0
def calculate_sum(a, b):
    total = a + b
    print("Inside function, total =", total)
calculate_sum(5, 10)
print("Outside function, total =", total)

counter = 0
def increment_counter():
    global counter
    counter += 1
increment_counter()
print("Counter after increment:", counter)
```

6 Introduction

This document provides a detailed explanation of important Python modules for beginners. Each section includes code examples and explanations. These modules include:

- `math` – mathematical functions and constants
- `datetime` – date and time operations
- `numpy` – numerical computations and arrays
- `pandas` – data analysis and manipulation
- `matplotlib` – visualization and plotting

7 Math Module

The `math` module provides access to mathematical functions like square roots, power, logarithmic, and trigonometric calculations.

7.1 Code and Explanation

```
import math  # Loads the math module
num = 16
print(f"Square root of {num} is {math.sqrt(num)}")
```

This code computes the square root of 16 using the `math.sqrt()` function.

```
x = 5.7
print(f"Ceiling of {x} is {math.ceil(x)}")
print(f"Floor of {x} is {math.floor(x)}")
```

The `ceil()` function rounds up and `floor()` rounds down.

Other functions include:

- `math.pow(x, y)` – raises x to the power y
- `math.exp(x)` – returns e^x
- `math.log(x)` – natural logarithm
- `math.log10(x)` – base-10 log
- `math.sin()`, `math.cos()` – trigonometric
- Constants: `math.pi`, `math.e`

8 Datetime Module

This module deals with date and time manipulation.

8.1 Examples

```
import datetime
now = datetime.datetime.now()
print("Current Date and Time:", now)
```

The `now()` function returns the current date and time. You can extract individual components like:

```
print("Year:", now.year)
print("Month:", now.month)
print("Day:", now.day)
```

You can also do arithmetic using `timedelta`:

```
today = datetime.date.today()
one_week = datetime.timedelta(weeks=1)
print("Next week:", today + one_week)
```

9 NumPy Module

NumPy is used for numerical computing. Arrays are its backbone.

9.1 Creating Arrays

```
import numpy as np
arr1 = np.array([1, 2, 3, 4])
arr2 = np.array([[1, 2], [3, 4]])
```

You can perform operations directly:

```
print(arr1 + 5)
print(np.sum(arr1))
print(np.mean(arr1))
print(np.max(arr2))
```

9.2 Other Utilities

```
arr = np.linspace(0, 10, 5)
print(arr)

arr = np.arange(1, 15, 4)
print(arr)

arr = np.logspace(1, 40, 5)
print(arr)
```

These generate sequences linearly, with step, or logarithmically spaced.

10 Pandas Module

Pandas is used for structured data – Series and DataFrames.

10.1 Working with Series

```
import pandas as pd
marks = pd.Series([85, 90, 95], index=['Math', 'Science', 'English'])
print(marks)
```

10.2 Working with DataFrames

```
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [20, 21, 19],
    'Branch': ['CS', 'EE', 'ME']
}
df = pd.DataFrame(data)
print(df)
```

You can analyze and manipulate data easily:

```
print(df['Name'])       # Column
print(df.iloc[0])       # Row
print(df['Age'].mean()) # Statistics
```

11 Matplotlib Module

This is used for plotting data visually.

11.1 Line Plot

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 35]
plt.plot(x, y, label="Line Plot", color='green')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Line Graph Example")
plt.legend()
plt.grid(True)
plt.show()
```

11.2 Bar Plot

```
subjects = ['Math', 'Science', 'English']
scores = [85, 90, 80]
plt.bar(subjects, scores, color='skyblue')
plt.title("Subject Scores")
plt.ylabel("Marks")
plt.show()
```

11.3 Pie Chart

```
labels = ['Python', 'C++', 'Java']
sizes = [40, 35, 25]
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.title("Programming Language Usage")
plt.show()
```

12 Conclusion

This document has shown how to use built-in and third-party Python modules with simple examples. These modules are critical for developing efficient code and handling data, math, and visualization tasks.

Practice these modules and try modifying the code examples to enhance your understanding.