# Python Programming Notes

**Unit 1: Introduction, Expressions, and Statements**

**Detailed Explanation:**

**1. Python Introduction:**

- Created by Guido van Rossum in 1991
- Interpreted language - code executed line by line
- High-level language - human-readable syntax
- Cross-platform - runs on Windows, Mac, Linux

**2. Variables and Data Types:**

- **Variables:** Containers for storing data values

```
name = "John"      # string
age = 25           # integer
height = 5.9       # float
is_student = True  # boolean
```

- **Data Types:**
  - int: whole numbers (10, -5, 0)
  - float: decimal numbers (3.14, -2.5)
  - str: text in quotes ("hello", 'Python')
  - bool: True or False
  - None: represents absence of value

**3. Expressions and Operators:**

- **Arithmetic:** + - * / // % **
- **Comparison:** == != > < >= <=
- **Logical:** and or not
- **Assignment:** = += -= *= /=

**4. Type Conversion:**

- **Implicit:** Python automatically converts

```
x = 5 + 2.5  # int + float = float (7.5)
```

- **Explicit:** Manual conversion

```
num_str = "10"
num_int = int(num_str)  # string to integer
```

**Video Links:**

- **English:** Python Full Course for Beginners
- **Hindi:** Python Basics in Hindi

**Unit 2: Control Flow and Loops**

**Detailed Explanation:**

**1. Conditional Statements:**

- **if-elif-else:** Multiple conditions

```
marks = 85
if marks >= 90:
    print("Grade A")
elif marks >= 75:
    print("Grade B")
else:
    print("Grade C")
```

**2. Boolean Operators:**

- **and:** Both conditions must be True
- **or:** At least one condition True
- **not:** Reverse the condition

**3. Loops:**

- **for loop:** Iterate over sequences

```
# Print numbers 0 to 4
for i in range(5):
    print(i)

# Iterate through list
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

- **while loop:** Repeat until condition false

```
count = 0
while count < 5:
    print(count)
    count += 1  # Important: update counter
```

**4. Loop Control Statements:**

- **break:** Exit loop completely
- **continue:** Skip current iteration
- **pass:** Placeholder for empty block

**Video Links:**

- **English:** Python Conditions and Loops
- **Hindi:** Control Statements in Hindi

---

**Unit 3: Functions and Modular Programming**

**Detailed Explanation:**

**1. Function Definition:**

```
def function_name(parameters):
    """Docstring - describes function"""
    # function body
    return value
```

**2. Types of Arguments:**

- **Positional:** Matched by position
- **Keyword:** Specified by parameter name
- **Default:** Predefined values
- **Variable-length:** *args and **kwargs

**3. Variable Scope:**

- **Local:** Inside function only
- **Global:** Accessible everywhere
- **nonlocal:** In nested functions

**4. Recursion:**

- Function calling itself
- Must have base case to stop

```python
def factorial(n):
    if n == 1:  # base case
        return 1
    else:
        return n * factorial(n-1)
```

**5. Lambda Functions:**

- Anonymous one-line functions

```python
square = lambda x: x * x
add = lambda a, b: a + b
```

**Video Links:**

- **English:** Python Functions Complete Guide
- **Hindi:** Functions in Python Hindi

---

**Unit 4: Strings and Collections**

**Detailed Explanation:**

**1. Strings:**

- **Indexing:** Access individual characters

```python
text = "Python"
print(text[0])  # 'P'
print(text[-1]) # 'n' (last character)
```

- **Slicing:** Extract substring

```python
text = "Hello World"
print(text[0:5])   # "Hello"
print(text[6:])    # "World"
print(text[::-1])  # "dlroW olleH" (reverse)
```

- **Methods:** upper(), lower(), strip(), split(), replace()

**2. Lists:**

- Ordered, mutable collection

```python
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")      # Add item
fruits.remove("banana")      # Remove item
fruits[0] = "mango"          # Modify item
```

**3. Tuples:**

- Ordered, immutable collection

```python
coordinates = (10, 20)
# coordinates[0] = 15  # ERROR - tuples are immutable
```

**4. Dictionaries:**

- Key-value pairs

```
student = {
    "name": "Alice",
    "age": 20,
    "grade": "A"
}
print(student["name"])  # Access value
student["age"] = 21     # Modify value
```

**5. Sets:**

- Unordered, unique elements

```
unique_numbers = {1, 2, 3, 2, 1}  # {1, 2, 3}
```

**Video Links:**

- **English:** Python Data Structures
- **Hindi:** Python Collections Hindi

**Unit 5: Files and Exception Handling**

**Detailed Explanation:**

**1. File Operations:**

- **Modes:** 'r' (read), 'w' (write), 'a' (append), 'r+' (read+write)

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

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

**2. Exception Handling:**

```
try:
    num = int(input("Enter number: "))
    result = 10 / num
except ValueError:
    print("Please enter valid number")
except ZeroDivisionError:
    print("Cannot divide by zero")
else:
    print("Result:", result)
finally:
    print("Execution completed")
```

**3. Custom Exceptions:**

```python
class CustomError(Exception):
    pass

try:
    age = -5
    if age < 0:
        raise CustomError("Age cannot be negative")
except CustomError as e:
    print(e)
```

**Video Links:**

- **English:** File Handling & Exceptions
- **Hindi:** File Handling in Hindi

---

**Unit 6: Object-Oriented Programming**

**Detailed Explanation:**

**1. Classes and Objects:**

```python
class Person:
    # Constructor
    def __init__(self, name, age):
        self.name = name    # instance variable
        self.age = age

    # Method
    def introduce(self):
        return f"I'm {self.name}, {self.age} years old"

# Create object
person1 = Person("John", 25)
print(person1.introduce())
```

**2. Inheritance:**

```python
class Student(Person):  # Inherits from Person
    def __init__(self, name, age, student_id):
        super().__init__(name, age)
        self.student_id = student_id

    # Method overriding
    def introduce(self):
        return f"I'm student {self.name}, ID: {self.student_id}"
```

**3. Encapsulation:**

```python
class BankAccount:
    def __init__(self):
        self.__balance = 0  # Private variable

    def deposit(self, amount):
        self.__balance += amount

    def get_balance(self):
        return self.__balance
```

**4. Polymorphism:**

```
def make_introduction(person):
    print(person.introduce())  # Works for both Person and Student
```

**Video Links:**

- **English:** Python OOP Tutorial
- **Hindi:** OOP in Python Hindi

---

**Unit 7: Advanced and Real-world Applications**

**Detailed Explanation:**

**1. Regular Expressions:**

```
import re

text = "Contact: john@email.com, sarah@test.org"
emails = re.findall(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b', text)
print(emails)  # ['john@email.com', 'sarah@test.org']
```

**2. GUI with Tkinter:**

```
import tkinter as tk

def on_click():
    label.config(text="Button Clicked!")

window = tk.Tk()
window.title("My App")

label = tk.Label(window, text="Hello World")
label.pack()

button = tk.Button(window, text="Click Me", command=on_click)
button.pack()

window.mainloop()
```

**3. Working with APIs:**

```
import requests

response = requests.get("https://api.github.com/users/octocat")
data = response.json()
print(data['name'], data['public_repos'])
```

**4. Popular Libraries:**

- **NumPy:** Scientific computing with arrays
- **Pandas:** Data manipulation and analysis

- **Matplotlib:** Creating visualizations and plots

**5. Mini Projects:**

- **Calculator:** Basic arithmetic operations
- **To-Do List:** Add, remove, mark complete tasks
- **Weather App:** Fetch data from weather API
- **Expense Tracker:** Record and categorize expenses

**Video Links:**

- **English:** Python Advanced Topics
- **Hindi:** Python Projects in Hindi

---

**Practice Resources:**

- **Online Compiler:** Replit
- **Coding Practice:** HackerRank Python
- **Documentation:** Python Official Docs