Python Programming Notes

Computer Science - Class 9

Subject: Computer Science

Level: Beginner / Class 9

Academic Year: 2026-27

Table of Contents

1. Introduction to Programming

Programming is the process of giving instructions to a computer to perform specific tasks. Think of it like giving a recipe to a chef. The chef follows the steps to create a dish, and the computer follows code to execute a program.
Key Concepts:

- **Algorithm:** A step-by-step procedure to solve a problem.
- **Source Code:** The instructions written by the programmer.
- **Compiler/Interpreter:** Special programs that translate human language (code) into machine language (0s and 1s).

> Summary: Programming is the bridge between human logic and computer execution.

2. What is Python? Features and Advantages

Python is a high-level, general-purpose programming language. It was created by **Guido van Rossum** and released in 1991. Today, it is one of the most popular languages in the world. **Features of Python:**

- **Simple and Easy to Learn:** Its syntax is very close to English.
- **Interpreted:** Code is executed line-by-line, which makes debugging easy.
- **Case-Sensitive:** Variable 'apple' is different from 'Apple'.
- **Large Library Support:** Tons of pre-written code for complex tasks like AI, data analysis, and web development.

Advantages:

- Beginner-friendly.
- Runs on all platforms (Windows, Mac, Linux).
- Open-source and free.

> Summary: Python is powerful yet simple, making it the perfect first language for students.

3. Installing Python and Setting Up Environment

To start coding in Python, you need an 'Interpreter' and an 'Editor'. **Steps to Install:**

1. Visit `python.org`.
2. Download the latest version for your OS.
3. While installing, make sure to check the box **"Add Python to PATH"**.

Environment Options:

- **IDLE:** The built-in editor that comes with Python.
- **VS Code:** A professional code editor.
- **Online Compilers:** Websites like Replit or Programiz if you don't want to install software.

> Summary: Once installed, you can use the IDLE shell to run individual lines or a script to run a full program.

4. Basics of Python Syntax

Syntax refers to the rules of the language. If you break these rules, Python will show a **SyntaxError**. **Key Rules:**

- **Indentation:** Python uses spaces to define blocks of code (unlike { } in other languages). Usually, 4 spaces or 1 tab is used.
- **Comments:** Use the # symbol for notes. Python ignores these. Perfect for explaining your logic!

```
# This is a comment
print("Hello, Class 9!") # This prints text to the screen
```

Summary: Correct indentation is mandatory in Python; otherwise, your program won't run.

5. Variables and Data Types

A **Variable** is like a container or a box that stores data. You can give it a name and put a value inside. **Naming Rules:**

- Must start with a letter or underscore.
- Cannot start with a number.
- Only contains letters, numbers, and underscores (A-z, 0-9, and _).

Common Data Types:

Type	Description	Example
`int`	Integers (whole numbers)	10, -5, 100
`float`	Decimal numbers	3.14, -0.5, 2.0
`str`	String (text in quotes)	"Hello", 'Python'
`bool`	Boolean (True or False)	True, False

```
age = 15          # int
height = 5.6      # float
name = "Rahul"    # str
is_student = True # bool
```

Summary: Variables store data of different types like numbers, text, or logical values.

6. Input and Output in Python

Programs need to talk to users. We use `print()` to show output and `input()` to take data from the user. **Output using print():**

```
print("Hello World")
print("I am", 14, "years old") # Using commas to separate
values
```

Input using input(): By default, `input()` always treats whatever you type as a **String**.

```
name = input("Enter your name: ")
age = int(input("Enter your age: ")) # int() converts string
to integer
```

> Summary: Use types casting like int() or float() when taking numerical input from a user.

7. Operators in Python

Operators are symbols used to perform operations on variables and values.

Arithmetic Operators:

- + (Addition)
- - (Subtraction)
- * (Multiplication)
- / (Division - results in a float)
- // (Floor Division - results in a whole number)
- % (Modulus - gives the remainder)
- ** (Exponentiation - power)

Comparison Operators: (These return True or False)

- == (Equal to)
- != (Not equal to)
- > (Greater than), < (Less than)

Logical Operators:

- and (True if both are true)
- or (True if at least one is true)
- not (Reverses the state)

> Summary: Operators allow us to calculate, compare, and connect logic in our code.

8. Conditional Statements

Conditionals allow a program to make decisions. It's like saying "If it rains, I will take an umbrella; otherwise, I will wear a hat." **The if-elif-else Structure:**

```
marks = int(input("Enter your marks: "))

if marks ⩾ 90:
    print("Grade: A")
elif marks ⩾ 75:
    print("Grade: B")
elif marks ⩾ 40:
    print("Grade: C")
else:
    print("Grade: F (Fail)")
```

Important Rules:

- The `if` statement must have a condition followed by a colon (`:`).
- The code inside the block MUST be indented.
- `elif` (short for else-if) is used for checking multiple conditions.

> Summary: Use 'if' for choice and 'indentation' to show what happens when the choice is picked.

9. Loops in Python

Loops are used to repeat a block of code multiple times. This saves us from writing the same code again and again. **1. The `while` Loop:** Repeats as long as a condition is **True**.

```
count = 1
while count ⩽ 5:
    print("Counting:", count)
    count = count + 1 # Increment to avoid infinite loop
```

2. The `for` Loop: Used for iterating over a sequence (like a list or a range of numbers).

```
# Using range(start, stop)
for i in range(1, 6):
    print("Number:", i)
```

Stop and Continue:

- `break`: Stops the loop immediately.
- `continue`: Skips the current turn and goes to the next one.

> Summary: Loops automate repetition. 'while' depends on a condition, 'for' works on a set range or sequence.

10. Introduction to Functions

A function is a reusable block of code that only runs when it is called. You can pass data (parameters) into it. **Defining and Calling a Function:**

```
# Definition
def greet_student(name):
    print("Hello", name, "! Welcome to Python class.")

# Calling the function
greet_student("Amit")
greet_student("Sara")
```

Why use functions?

- Reduces duplication.
- Makes code organized.
- Easier to fix bugs in one place.

> Summary: Functions are like 'mini-programs' inside your main program that you can use over and over.

11. Lists and Tuples

These are used to store multiple items in a single variable. **1. Lists (Mutable):** Items are ordered and can be changed.

```
fruits = ["apple", "banana", "cherry"]
fruits[1] = "orange" # Changes banana to orange
fruits.append("mango") # Adds mango to the end
```

2. Tuples (Immutable): Items are ordered but **CANNOT** be changed once created. Use rounded brackets.

```
dimensions = (1920, 1080)
# dimensions[0] = 1000 # This would cause an ERROR
```

> Summary: Lists are for collections that change; Tuples are for data that should stay constant.

12. Simple Python Programs (Examples)

Here are 5 beginner-friendly programs for your practice: **#1: Calculate Area of a Rectangle**

```
length = float(input("Enter length: "))
width = float(input("Enter width: "))
area = length * width
print("The area is:", area)
```

#2: Check if Number is Even or Odd

```
num = int(input("Enter a number: "))
if num % 2 == 0:
    print("It is Even")
else:
    print("It is Odd")
```

#3: Sum of First N Natural Numbers

```
n = int(input("Enter N: "))
total = 0
for i in range(1, n + 1):
    total += i
print("The sum is:", total)
```

#4: Factorial of a Number

```
num = int(input("Enter a number: "))
fact = 1
for i in range(1, num + 1):
    fact *= i
print("Factorial is:", fact)
```

#5: Convert Celsius to Fahrenheit

```
c = float(input("Celsius temperature: "))
f = (c * 9/5) + 32
print("Fahrenheit:", f)
```

13. Common Errors and Debugging Tips

Don't panic when your code doesn't work! Every programmer makes mistakes. **Types of Errors:**

- **Syntax Error:** Missing a colon `:`, unclosed quotes `""`, or wrong indentation.
- **Runtime Error:** Code is valid but things go wrong while running (e.g., dividing by zero).
- **Logical Error:** Program runs but gives the wrong output (e.g., using `+` instead of `-`).

Debugging Tips:

1. Read the **Traceback** (error message) carefully — it tells you the line number!
2. Use `print()` statements to check variable values at different steps.
3. Check your indentation.
4. Comment out parts of your code to isolate the problem.

Final Revision Section

Quick-Fire Revision:

- **Creator:** Guido van Rossum (1991).
- **Comments:** Use #.
- **Case-Sensitive:** age vs Age.
- **Input:** Always a `str` unless converted (cast).
- **Loop:** `while` for conditions, `for` for ranges/lists.
- **Mutable:** Lists can be changed.
- **Immutable:** Tuples cannot be changed.

> Best of luck with your coding journey! Practice makes perfect.