

Name:-Vaibhav kumar gupta

Date:-21-02-2025

Python:-

Python Basics

Python is a high-level, interpreted programming language known for its **simplicity, readability, and versatility**. It is widely used in web development, data science, automation, and more.

Key Features

- ✓ **Easy to learn** – Simple syntax
- ✓ **Interpreted** – No need to compile
- ✓ **Dynamically typed** – No need to declare variable types
- ✓ **Huge libraries** – NumPy, Pandas, TensorFlow, etc.

Code:-

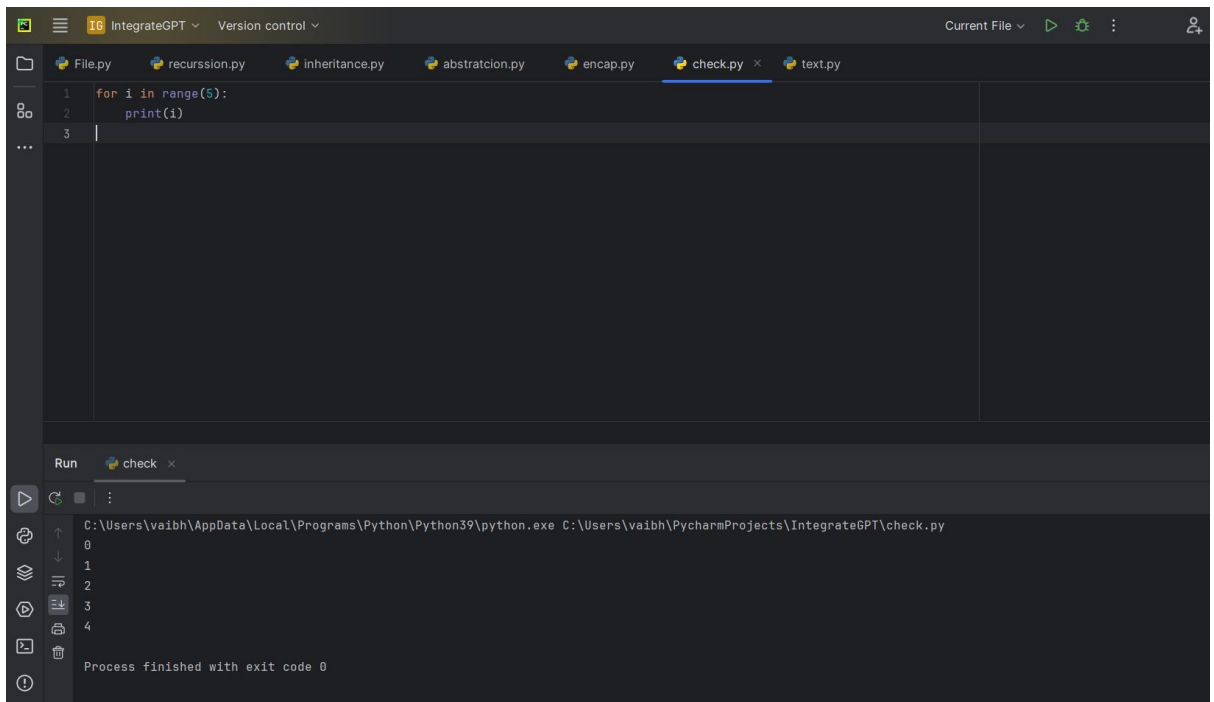
```
# Python Hello World  
print("Hello, World!")
```

Loops in Python

Loops in Python allow **repeating a block of code multiple times**.

Types of Loops

1. **For Loop** – Iterates over a sequence (list, tuple, dictionary, string, etc.)



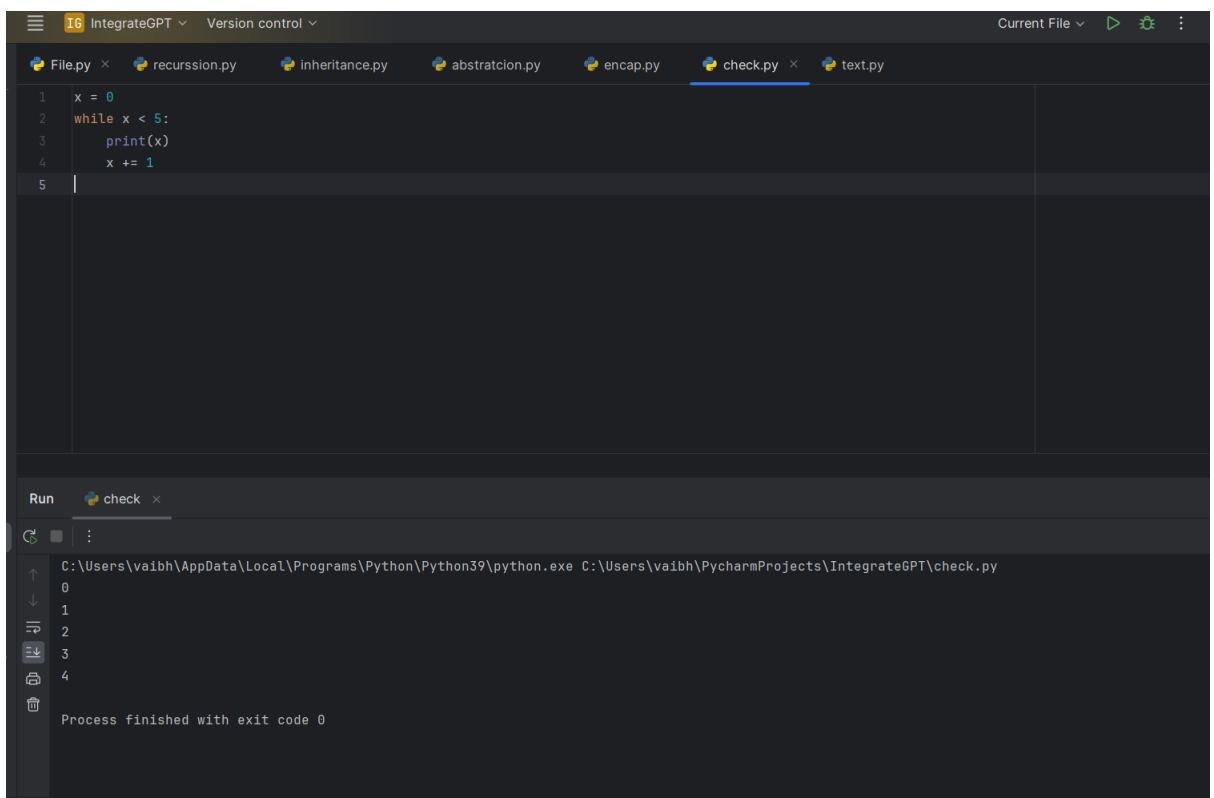
The screenshot shows the PyCharm IDE with a project named 'IntegrateGPT'. The 'check.py' file is open and contains a for loop that prints numbers 0 through 4. The Run console shows the output of the program, which is the numbers 0, 1, 2, 3, and 4, each on a new line. The process finished with exit code 0.

```
1 for i in range(5):
2     print(i)
3
```

Run check x

```
C:\Users\vaibh\AppData\Local\Programs\Python\Python39\python.exe C:\Users\vaibh\PycharmProjects\IntegrateGPT\check.py
0
1
2
3
4
Process finished with exit code 0
```

2. While Loop – Repeats as long as a condition is true



The screenshot shows the PyCharm IDE with a project named 'IntegrateGPT'. The 'check.py' file is open and contains a while loop that prints numbers 0 through 4. The Run console shows the output of the program, which is the numbers 0, 1, 2, 3, and 4, each on a new line. The process finished with exit code 0.

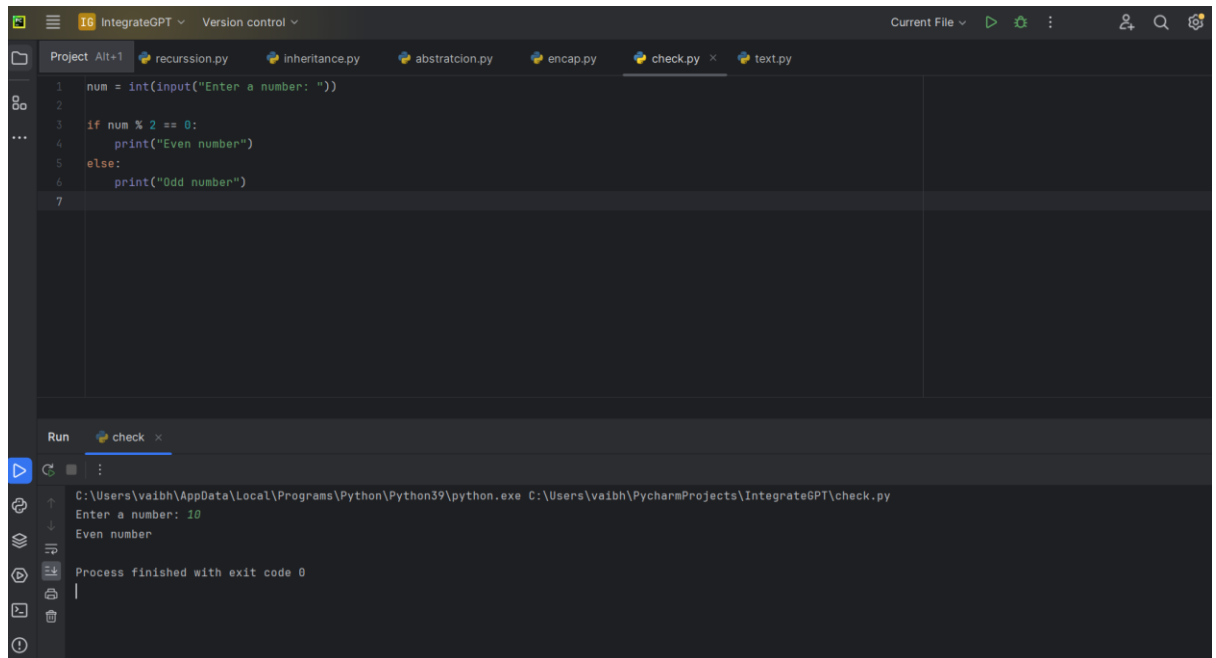
```
1 x = 0
2 while x < 5:
3     print(x)
4     x += 1
5
```

Run check x

```
C:\Users\vaibh\AppData\Local\Programs\Python\Python39\python.exe C:\Users\vaibh\PycharmProjects\IntegrateGPT\check.py
0
1
2
3
4
Process finished with exit code 0
```

if-else Statements in Python

if-else is a conditional statement that allows a program to make decisions.



The screenshot shows a Python IDE with a project named 'IntegrateGPT'. The file explorer on the left shows several Python files: recursion.py, inheritance.py, abstraction.py, encap.py, check.py, and text.py. The main editor window displays the code for check.py:

```
1 num = int(input("Enter a number: "))
2
3 if num % 2 == 0:
4     print("Even number")
5 else:
6     print("Odd number")
7
```

Below the editor, the 'Run' panel shows the execution of check.py. The output is:

```
C:\Users\vaibh\AppData\Local\Programs\Python\Python39\python.exe C:\Users\vaibh\PycharmProjects\IntegrateGPT\check.py
Enter a number: 10
Even number
Process finished with exit code 0
```

.toml – Configuration File Format

TOML (Tom's Obvious, Minimal Language) is a simple configuration file format, similar to JSON but more readable.



Why Use .toml?



Human-friendly syntax



Used for project configurations (e.g., pyproject.toml)



Easier to edit than JSON/YAML



Example: pyproject.toml

```
1  [build-system]
2  requires = ["setuptools", "wheel"]
3  build-backend = "setuptools.build_meta"
4
5  [project]
6  name = "my_project"
7  version = "0.1.0"
8  dependencies = [
9      "numpy",
10     "requests"
11 ]
12 |
```

Bazel – Build System

Bazel is a build tool developed by Google for efficiently compiling and managing large projects.

Why Use Bazel?

- ✓ Fast builds – Uses caching and parallel execution
- ✓ Scalability – Ideal for large projects
- ✓ Multi-language support – Supports Python, Java, C++, Go, etc.
- ✓ Reproducibility – Ensures consistent builds

Basic Bazel Workflow

Define a Build File (BUILD or WORKSPACE)

Run Bazel Commands (bazel build //target)

Execute the Binary (bazel run //target)

Example: Bazel Build File

```
cc_binary(  
  name = "hello-world",  
  srcs = ["hello.cpp"],  
)  
|
```

Build & Run

```
bazel build //:hello-world  
bazel run //:hello-world  
|
```