Assignment 1:

Code Review and Error Correction

1.**Code Snippet 1**

```
def add_numbers(a, b)
return a + b
print(add_numbers(5, 10))
```

Code Review:- **Missing colon (:) at the end of the function definition line.**

Correct code:-

```
def add_numbers(a, b):
return a + b
print(add_numbers(5, 10))
```

Explanation:-

In Python, every function definition line must end with a colon (:) to indicate the start of the function block.
Without it, Python will throw a SyntaxError.

2.**Code Snippet 2**

```
name = "Alice
print("hello, " + name)
```

Code Review:- Missing closing quotation mark in the string "Alice.

Correct code:-

```
name = "Alice"
print("hello, " + name)
```

Explanation:-

In Python, string literals must be enclosed in matching quotes.

3.**Code Snippet 3**

```
for i in range(5)
print("Number:", i)
```

Code Review:- Missing colon (:) after for loop statement

Correct code:-

```
for i in range(5):
print("Number:", i)
```

Explanation:-

In Python, a colon (:) is required after a loop declaration to indicate the start of the loop body.

4.**Code Snippet 4**

```
my_list = [1, 2, 3, 4, 5]
print("The fifth element is: " + my_list[5])
```

Code Review:-

Lists in Python are zero-indexed, so my_list[5] is out of range. The last element’s index is 4.

Correct code:-

```
my_list = [1, 2, 3, 4, 5]
print("The fifth element is: " + str(my_list[4]))
```

Explanation:-

In Python, list indexing starts at 0. For a list of length 5, valid indices are 0 to 4.

Since "The fifth element is: " is a string, we must convert the integer my_list[4] to a string using str() before concatenation.

5.**Code Snippet 5**

```
def greet(name):
print("Hello " + name)
greet("Bob")
```

Code Review:-

no error in this code.

Correct code:-

```
def greet(name):
print("Hello " + name)
greet("Bob")
```

6.**Code Snippet 6**

```
age = input("Enter your age: ")
if age >= 18:
print("You are eligible to vote.")
else:
print("You are not eligible to vote.")
```

Code Review:-

input() in Python returns a string, so comparing it directly with an integer (18) causes a TypeError.

The value must be converted to an integer before comparison.

Correct code:-

```
age = int(input("Enter your age: "))
if age >= 18:
print("You are eligible to vote.")
else:
print("You are not eligible to vote.")
```

Explanation:-

Input() always returns data as a string. For numerical comparisons, it needs to be converted to an integer using int().

Without conversion, Python will not allow the comparison between str and int.

7.**Code Snippet 7**

```
def multiply(a, b):
result = a * b
return result
print(multiply(4, 5))
```

Code Review:-

return statement is placed outside the function body due to incorrect indentation

Correct code:-

```
def multiply(a, b):
result = a * b
```

```
return result
print(multiply(4, 5))
```

Explanation:-

- **The return statement must be indented to be part of the function body.**
- **Without proper indentation, Python treats it as outside the function, which either causes an IndentationError or makes the function return None.**

--

8.**Code Snippet 8**

```
count = 10
while count > 0
print(count)
count -= 1
print("Countdown complete!")
```

Code Review:-

The loop body (print(count) and count -= 1) must be indented to be part of the while loop.

Correct code:-

```
count = 10
while count > 0:
print(count)
```

```
count -= 1
print("Countdown complete!")
```

Explanation:-

In Python, indentation defines code blocks. The statements inside the while loop must be indented to indicate they belong to that loop.

The print("Countdown complete!") is outside the loop to ensure it only executes once after the loop ends.

TASK 2

History of Python:

1980s: Created by Guido van Rossum in the

Netherlands.

1991: Python 1.0 released.

2000: Python 2.0 introduced new features (list

comprehensions, garbage collection).

2008: Python 3.0 released with major changes.

Today: One of the most popular languages

globally.

Why Python is Popular:

- Easy to learn and write.
- Large standard library.
- Cross-platform compatibility.
- Strong community support.
- Rich ecosystem of third-party packages.

Functions in Python:

- A block of reusable code.
- Improves modularity and reduces redundancy.

Syntax:

```
def greet(name):
print("Hello,", name)
greet("Alice")
```

Types of functions:

- Built-in (e.g., len(), print())
- User-defined

Modules in Python:

- A file containing Python definitions and statements.

- Helps organize code logically.
- Syntax for importing:
- import math
- print(math.sqrt(16))
- Types:
- Built-in modules (math, os, sys)
- Custom modules

Benefits of Functions and Modules:

- Code reusability.
- Easier debugging.
- Better organization.
- Scalability for larger projects.

Conclusion:

- Python is beginner-friendly yet powerful.
- Functions and modules are essential for clean, reusable, and maintainable code.
- Python continues to grow in popularity across industries.