Functions Class Notes and Theory

CLASS THEORY: FUNCTIONS

1. Introduction to User-Defined Functions

A **user-defined function** is a block of code that:

1. Takes zero or more **parameters** (inputs).
2. Executes specific logic.
3. Optionally **returns** a result.

Why Use Functions?

- **Reusability**: Prevent repeated code blocks.
- **Organization**: Split complex tasks into simpler sub-tasks.
- **Maintainability**: Update logic in one place; all callers benefit.

2. Defining and Calling a Function

2.1. Function Definition

In `functions.py`, the `compareStudents()` function is defined like this:

```
def compareStudents(StudentA, StudentB, StudentC):
    if StudentA < StudentB:
        print("StudentA is less than StudentB")
        return 0
    elif StudentA > StudentB:
        print("StudentA is greater than StudentB")
        return 1
```

4. **Keyword** `def` indicates a function definition.
5. **Function Name**: `compareStudents`.
6. **Parameters**: `StudentA, StudentB, StudentC`.

7. **Colon** (`:`) signals the start of the function body.

Inside the Function

- A simple **if-elif** structure compares `StudentA` to `StudentB` .
- Depending on the comparison, it prints a message and **returns** either `0` or `1` .
- Note: `StudentC` is currently **unused** in the logic, but it's part of the function's parameters.

2.2. Calling the Function

Below the definition, the code calls `compareStudents()` multiple times:

```
studentOneMarks = 10
studentTwoMarks = 20
StudentThreeMarks = 30

compareStudents(studentOneMarks, studentTwoMarks)
compareStudents(studentOneMarks, StudentThreeMarks)
compareStudents(studentTwoMarks, studentOneMarks)
```

When calling:

8. We provide **arguments** matching the parameter order in the definition.
9. The function then executes, possibly printing or returning a value.

3. Parameters, Arguments, and Return Values

3.1. Parameters vs. Arguments

- **Parameters**: The function's placeholders (`StudentA, StudentB, StudentC`) in the definition.
- **Arguments**: The actual values passed to the function (e.g., `studentOneMarks`).

3.2. Return Values

- `compareStudents()` returns `0` or `1` based on the comparison. This return **exits** the function immediately.
- If no return was used, the function would default to returning `None` .

Note: We might print the returned value or store it if needed:

```
result = compareStudents(studentOneMarks, studentTwoMarks)
print("Function result:", result)
```

4. Example Usage and Extension

10. **Comparison Logic**
    - The function's code could be extended to compare `StudentA` with `StudentC` or do more complex checks.
11. **Even/Odd Checking**
    - The script hints at checking if a number (`Number = 1`) is even/odd. This could be another function:

```
def isEven(num):
    return num % 2 == 0
```

12. **Avoiding Repetition**
    - If repeated logic (like comparing students or checking even/odd) occurs, a function is the best place to house that code for reusability.

5. Scope and Variables

5.1. Local Scope

- Variables **inside** the function (e.g., `StudentA`) are local to that function.
- Once the function ends, those variables are destroyed or go out of scope.

5.2. Global Scope

- Variables like `studentOneMarks` and `studentTwoMarks` are in the file's global scope.
- They remain accessible until the program ends or they are changed.

6. Summary

- `compareStudents()` is a clear example of a **user-defined** function: it takes parameters, performs logic, and returns a value.
- Even though `compareStudents()` includes `StudentC` , the current logic does not use it—this might indicate planned functionality or a placeholder.
- Functions enable code reusability and help organize tasks like comparing student marks or checking even/odd numbers.

CLASS NOTES: FUNCTIONS

13. **Definition Syntax**

```
def functionName(param1, param2, ...):
    # function body
    return some_value
```

14. **Example from `functions.py`**

```
def compareStudents(StudentA, StudentB, StudentC):
    if StudentA < StudentB:
        print("StudentA is less than StudentB")
        return 0
    elif StudentA > StudentB:
        print("StudentA is greater than StudentB")
        return 1
```

15. **Calling the Function**

```
compareStudents(studentOneMarks, studentTwoMarks)
```

16. **Parameters vs. Arguments**

- **Parameters**: placeholders in the definition (`StudentA, StudentB, StudentC`).
- **Arguments**: real values passed in (`studentOneMarks, studentTwoMarks, StudentThreeMarks`).

17. **Return Value**

- `compareStudents()` returns either `0` or `1` , ending the function immediately.

18. **Unused Parameter**

- `StudentC` is present but not used in the `if-elif` . Could be utilized in future logic.

19. **Scope**

- **Local**: Inside `compareStudents()` , the names `StudentA, StudentB, StudentC` are local.
- **Global**: `studentOneMarks, studentTwoMarks, StudentThreeMarks` exist at the module level.

20. **Additional Example**: Even/Odd Check

```
def isOddOrEven(num):
    if num % 2 == 0:
        return "Even"
    else:
        return "Odd"
```