1. **Even when new values are introduced into an expression, the output remains the same unless the function is introduced again.**

E.g >>> carrots = 24

>>> rabbits = 8

>>> crs\_per\_rab = carrots/rabbits

Rabbits = 12

Output = 3.0

Unless you re-introduce;

>>> crs\_per\_rab = carrots/rabbits

>>> **print**(crs\_per\_rab)

Output = 2

1. **Flow of execution**

This is the order in which a code or statements are executed.

Execution begins at the first statement of the program.

Statements are executed one at a time from the top.

Statements inside the function are not executed until the function is called.

Function calls are like a bypass; other than going to the next statement, it jumps the first line of the called function, executes it, and comes back to where it left off.

**For example;**

**Example 1;**

2-print("welcome")

3-for x in range(3):

4-print(x)

5-print("Good morning college")

#The numbers are representatives of lines of code.

**Output**

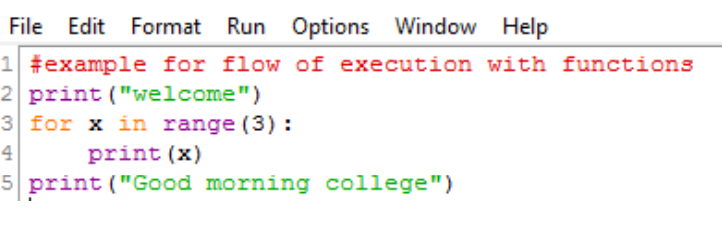
welcome

0

1

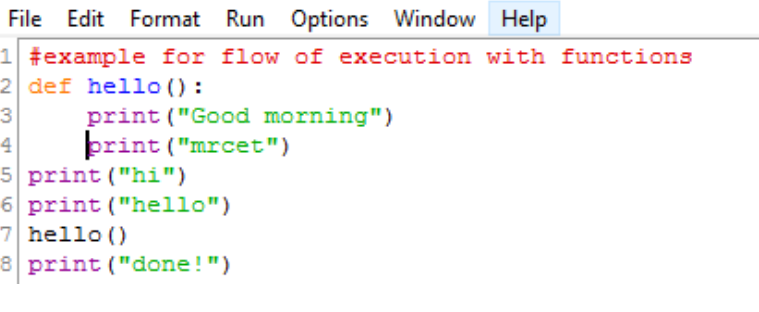
2

Good morning college



**Order of execution is 2,3,4,3,4,3,4,5**

**Example 2;**



Output

hi

hello

Good morning

mrcet

done!

**Flow/order of execution is: 2,5,6,7,2,3,4,7,8**

1. **When Reading a program, don’t read from top to bottom, follow the execution flow. Read the def statements from top to bottom, and skip the function's statements until the function is called.**

Example in code;

def greet(name):

print(f"Hello, {name}!")

def main():

greet("Alice")

greet("Bob")

main()

**Function Definitions:** When the interpreter starts reading the program, it first encounters the greet function definition. At this point, it notes that there's a function named greet but doesn't execute its body. The same happens with the main function definition.

**Function Call:** After defining both functions, the interpreter reaches the main() function call. It then jumps to the main function's body to execute its statements.

**Inside** **main:** The greet function is called twice within the main: first with the argument "Alice" and then with "Bob." Each time greet is called, the interpreter executes its body, printing the greeting message.

**Output**

Hello, Alice!

Hello, Bob!

**Key takeaways**

* The interpreter processes the code from top to bottom but only executes function bodies when those functions are called.
* When reading a program, it's essential to follow this flow of execution:
  + Acknowledge function definitions as you encounter them.
  + Skip reading the function's body until you reach a point where the function is called.
  + Upon encountering a function call, refer back to the function's definition to understand its behavior.

By following the flow of execution, you can better understand how a Python program operates and how data moves through the code.