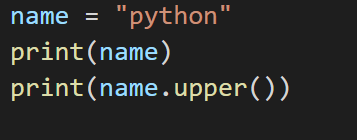
# **upper, lower, swapcase, capitalize and casefold, title String Functions in Python**

**upper()**

This method is used to convert all character of a string into uppercase.

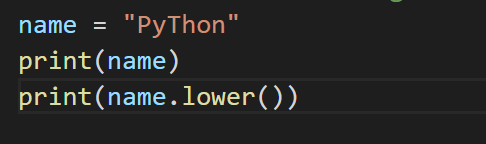
Syntax: string.upper()



**lower()**

This method is used to convert all character of a string into lowercase.

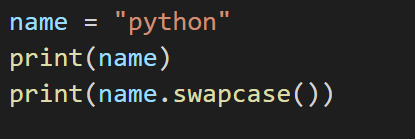
Syntax: string.lower()



**swapcase()**

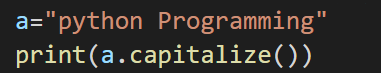
This method is used to convert all character of a string into uppercase and vice versa.

Syntax: string.swapcase()



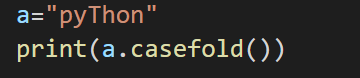
**capitalize()**

the **capitalize()** function returns a string where the first character is upper case and the rest is lower case.



**casefold()**

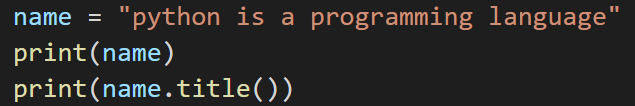
the **casefold()** function returns a string where all the characters are lower case.



**title()**

This method is used to convert the string in such that each word in string will start with a capital letter and remaining will be small letter.

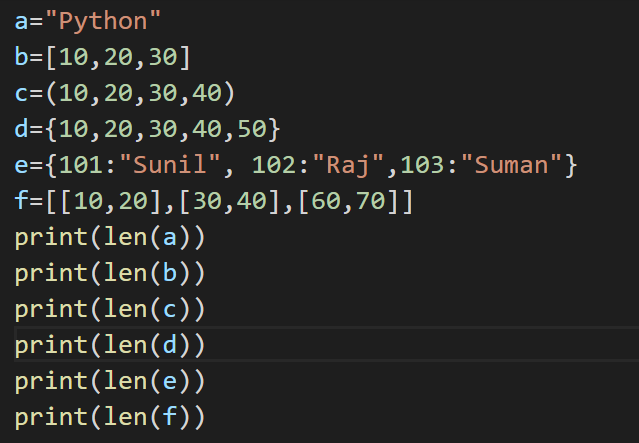
Syntax: string.title()



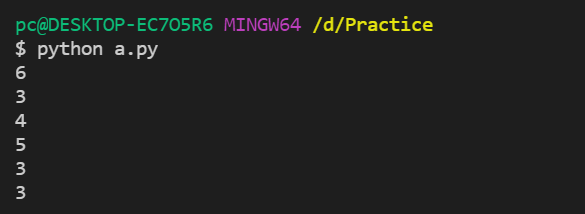
**Len() Function**

This function returns the length (the number of items) of an object. The argument may be a sequence (such as a string, tuple, list) or a collection (such as dictionary, set).

Syntax: len(arg)



**OUTPUT**



**Conditional Statement**

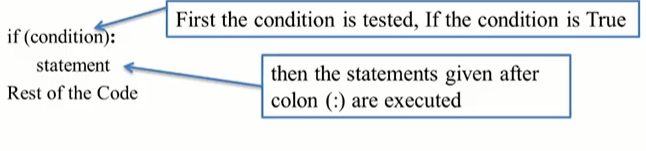
There comes, situations in real life when we need to make some decisions and based on these decisions, we decide what should we do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions, we will execute the next block of code. Decision-making statements in programming languages decide the direction of the flow of program execution. Decision making is the most important aspect of almost all the programming languages. As the name implies, decision making allows us to run a particular block of code for a particular decision. Here, the decisions are made on the validity of the particular conditions. Condition checking is the backbone of decision making.

While writing code in any language, you will have to control the flow of your program. This is generally the case when there is decision making involved - you will want to execute a certain line of codes if a condition is satisfied, and a different set of code in case it is not. In Python, you have the **if**, **elif** and the **else** statements for this purpose.

In Python, **if else elif statement** is used for decision making.

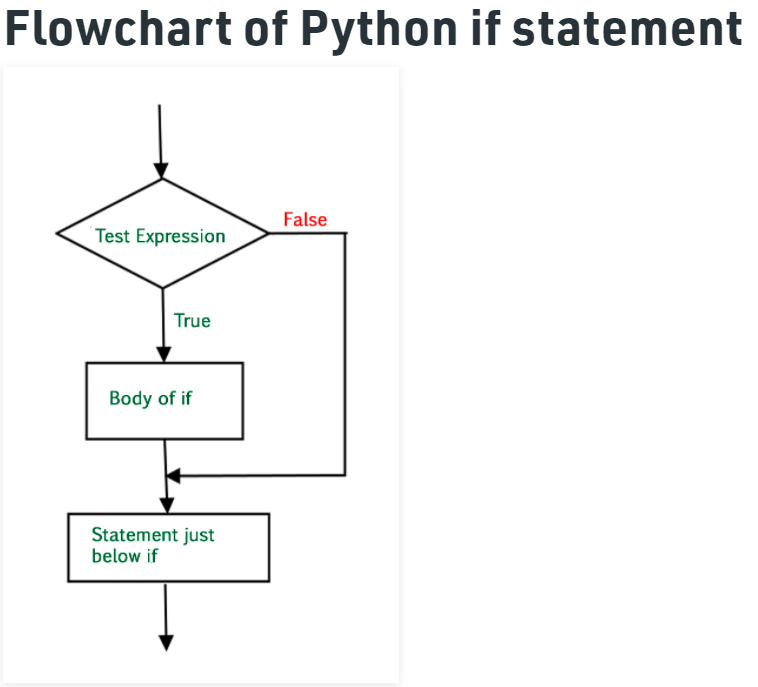
**if statement**

if statement is the simplest decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e., if a certain condition is true then a block of statement is executed otherwise not. This is the simplest example of a conditional statement. The syntax is:

The block of lines indented the same amount after the colon (:) will be executed whenever the condition is TRUE. The colon (:) is important because it separates the condition from the statements to be executed after the evaluation of the condition.

If there is single statement it can be written in one line.

**Example:** if(condition): Statement



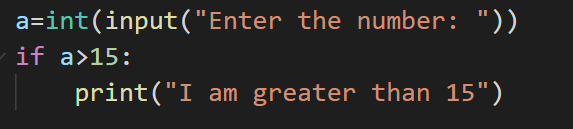
Here, the condition after evaluation will be either true or false. if statement accepts boolean values – if the value is true then it will execute the block of statements below it otherwise not. As we know, python uses indentation to identify a block. So, the block under an if statement will be identified as shown in the below example:

**if condition:**

statement1

statement2

**Write a python program to input one number from the user. Print “I am greater than 15” if the number is greater than 15?**



**if-else**

The if statement alone tells us that if a condition is true, it will execute a block of statements and if the condition is false, it won’t. But what if we want to do something else if the condition is false. Here comes the *else* statement. We can use the *else* statement with *if* statement to execute a block of code when the condition is false.

**Syntax**:

if (condition):

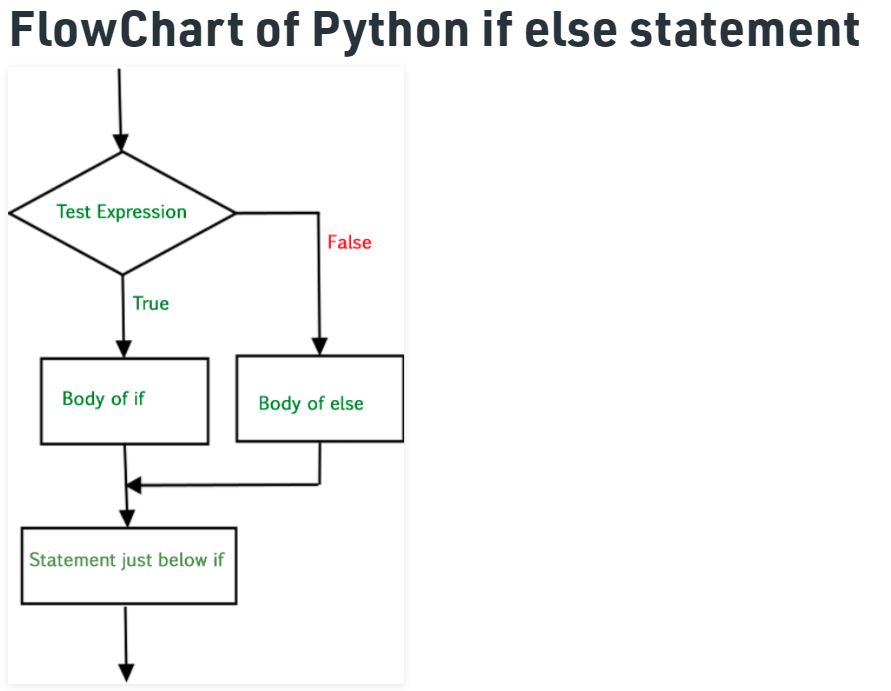
# Executes this block if

# condition is true

else:

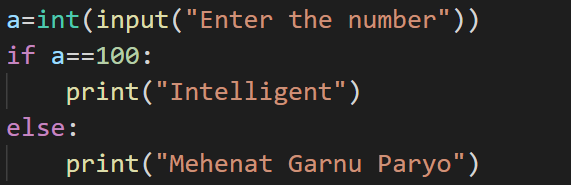
# Executes this block if

# condition is false

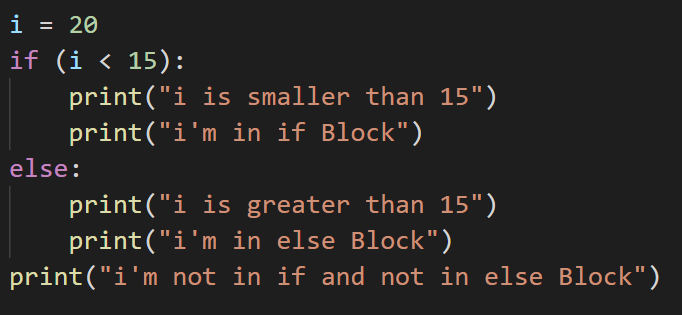


**Write a python program to input one number from the user. Print “Intelligent” if the number is 100, otherwise print “Mehenat Garnu Paryo”.**

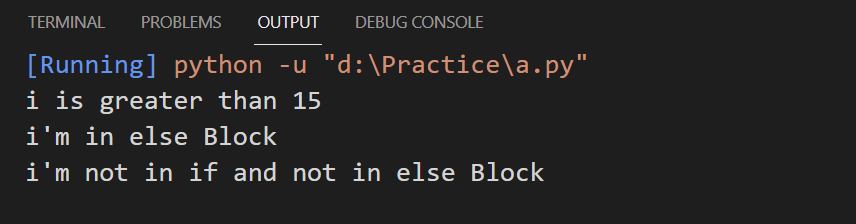
**Ans:**



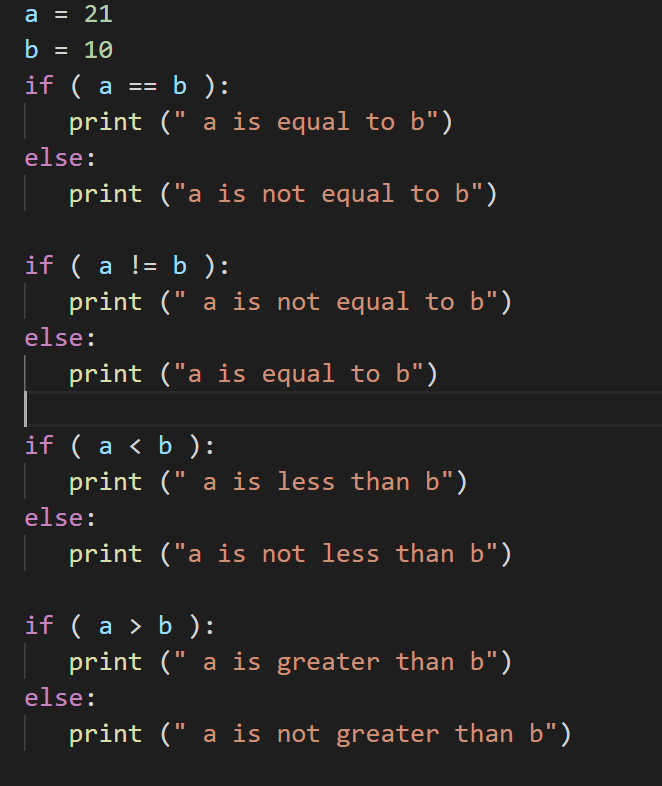
**Example: Python if else Statement**

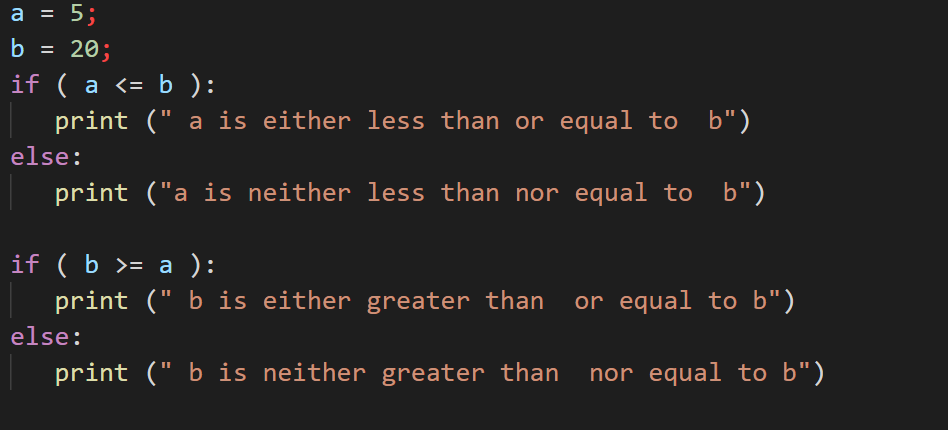


**OUTPUT**



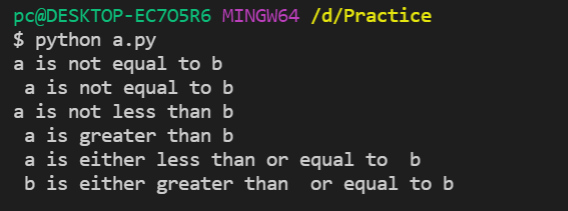
The block of code following the else statement is executed as the condition present in the if statement is false after calling the statement which is not in block (without spaces).





**OUTPUT**

When you execute the above program it produces the following result −



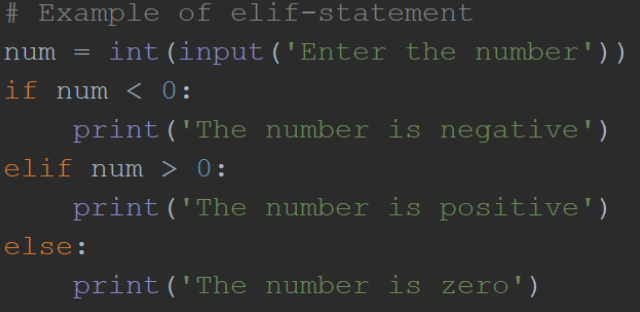
## elif statement

The elif statement enables us to check multiple conditions and execute the specific block of statements depending upon the true condition among them. We can have any number of elif statements in our program depending upon our need. However, using elif is optional.

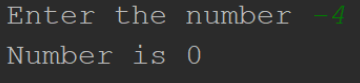
The syntax of the elif statement is given below



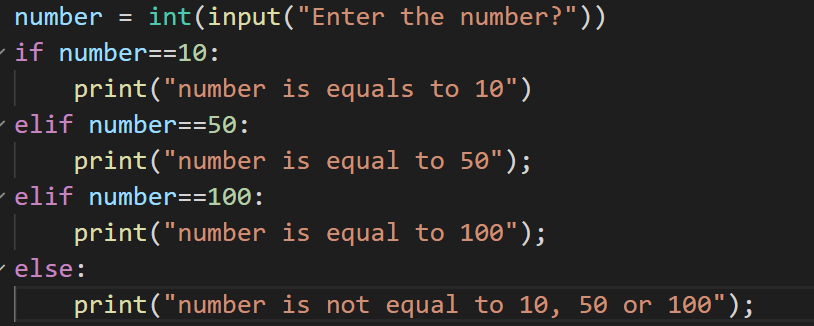
**Example:**



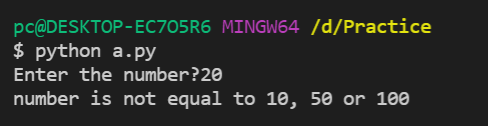
**OUTPUT**



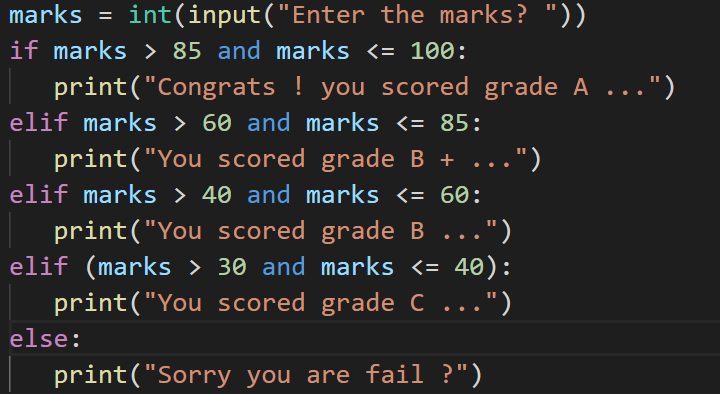
**Example 2:**



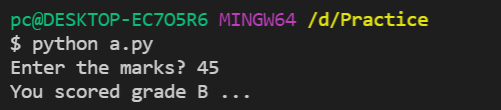
**OUTPUT**



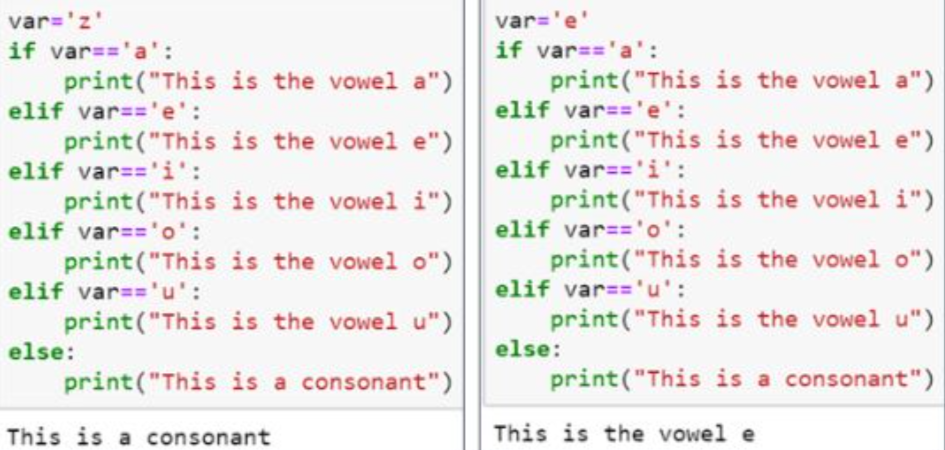
**Example 3:**



**OUTPUT**

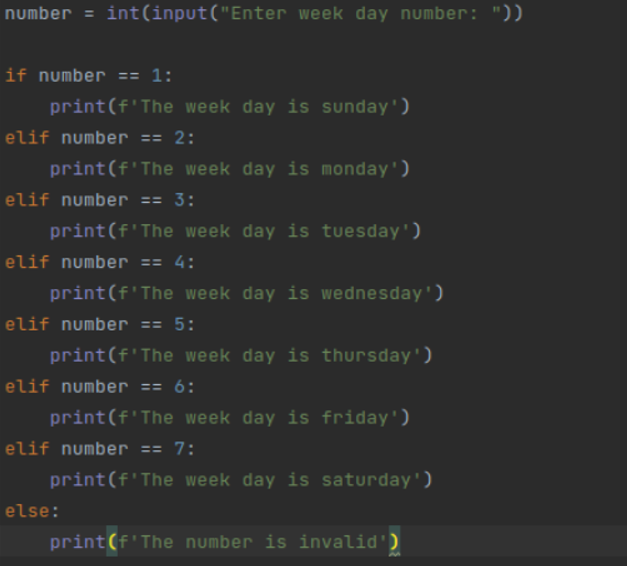


The program below uses the if-elif-else ladder to check if a letter is a vowel or a consonant.



**Question:** Write a program that asks the user for a number in the range of 1 to 7. The program should display the corresponding day of the week, where 1=sunday, 2=monday,3=tuesday,4=wednesday,5=thursday,6=friday,7=saturday. The program should display an error message if the user enters a number that is outside the range of 1 to 7.

**OUTPUT**



**Nested if Statement**

When an if a statement is present inside another if statement, it is called a nested IF statement. This situation occurs when you have to filter a variable multiple time.

### **Syntax:**

if (condition1):

    #Executes if condition1 is true

    if (condition2):

        #Executes if condition2 is true

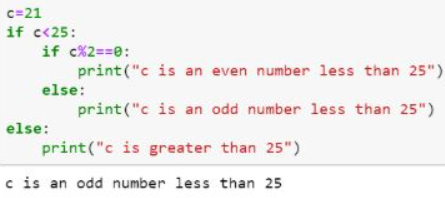
    #Condition2 ends here

#Condition1 ends here

In nested IF statements, you should always take care of the indentation to define the scope of each statement. You can have as many levels of nesting as required, but it makes the program more complex to read and understand. Therefore, you should always try to minimize the use of nested IF statements.

The following is another example that shows how nested IF works: We have a number, and we’re going to check if the number is greater or less than 25. If the number is less than 25, we’ll check if it is an odd number or an even number. If the number is greater than 25, we will print that the number is greater than 25.

**Example 1:**



**Example 2:**

