The Magic of Python

- The “>>>” is a Python *prompt* indicating that Python is ready for us to give it a command. These commands are called *statements*.
- >>> print("Hello, world“)

  Hello, world
- >>> print(2+3)

  5

```
>>> print("2+3=", 2+3)
        2+3= 5
>>>
```

Keywords

- predefined and reserved words in python that have special meanings. Keywords are used to define the syntax of the coding.

```
and       as        assert    break     class     continue  def
del       elif      else      except    finally   for       from
global    if        import    in        is        lambda    nonlocal
not       or        pass      raise     return    try       while
with      yield     false     none      true
```

Identifiers

- provide a name for a given program element with a sequence of one or more characters.
- Each variable name is an identifier.
- Rules to be followed:
  - Can only contain **alphanumeric** characters and **underscores** (a-z, A-Z, 0-9, _)
  - Must start with letters but **cannot start with a number**
  - Names are **case-sensitive**
  - Name cannot be a keyword
  - Should be one word with **no spaces** in between

- **Python does not allow special characters**
- Python is case sensitive, so rollNumber and RollNumber are two different identifiers.

Valid Identifiers		Invalid Identifiers	Reason Invalid
totalSales		'totalSales'	quotes not allowed
totalsales		total sales	spaces not allowed
salesFor2010		2010Sales	cannot begin with a digit
sales_for_2010		_2010Sales	should not begin with an underscore

Literals

- numbers, text, or other data that represent values to be stored in variables.
- Raw data assigned to variables, or, constants used while programming
  - Numeric Literals, digits 0-9, sign character and decimal point.
    - Age=22
    - Value=22.32
    - Angle=-90

- String Literals
  - delimited (surrounded) by a matching pair of either single (') or double (") quotes.

```
'A'                          - a string consisting of a single character
'jsmith16@mycollege.edu'     - a string containing non-letter characters
"Jennifer Smith's Friend"    - a string containing a single quote character
' '                          - a string containing a single blank character
''                           - the empty string
```

- Boolean Literals, assign boolean values
- Literal Collections, defines lists, tuples, dictionary, sets
- Special Literals, none literal
  - To assign null value to a variable.

Basic Input, Output, and String Formatting in Python

- input() always returns a string

```
>>> name = input("What is your name? ")
What is your name? Amrita
>>> name
'Amrita'
```

- If you want a number to read, convert the string to the appropriate type with the built-in int(), float() or complex() functions.

Python

```
>>> number = input("Enter a number: ")
Enter a number: 50
>>> print(number + 100)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be str, not int

>>> number = int(input("Enter a number: "))
Enter a number: 50
>>> print(number + 100)
150
```

Python f-strings

- To use formatted string literals, begin a string with f or F before the opening ‘ or “.
- Inside this f string, write Python expressions between { }, which can refer the variable values.

```
>>> year = 2016
>>> event = 'Referendum'
>>> f'Results of the {year} {event}'
'Results of the 2016 Referendum'
```

Python

```
>>> name = input("What is your name? ")
What is your name? Winston

>>> age = int(input("How old are you? "))
How old are you? 24

>>> print(name)
Winston

>>> print(age)
24
```

Python

```
>>> f"Hello, {name}. In 50 years, you'll be {age + 50}."
Hello, Winston. In 50 years, you'll be 74.
```

split() Method

- Split a string to a list of strings using a separator.

```
text = 'geeks for geeks'

# Splits at space
print(text.split())

word = 'geeks, for, geeks'

# Splits at ','
print(word.split(','))

word = 'geeks:for:geeks'

# Splitting at ':'
print(word.split(':'))

word = 'CatBatSatFatOr'

# Splitting at t
print(word.split('t'))
```

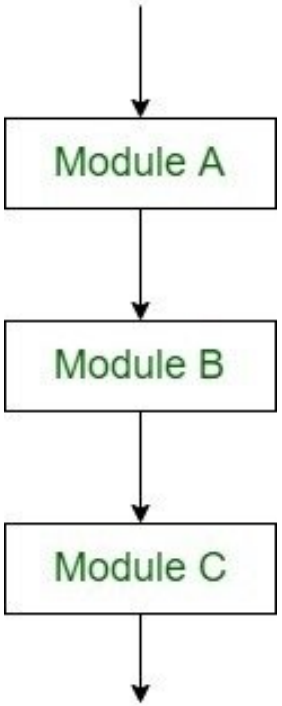
Output :

```
['geeks', 'for', 'geeks']
['geeks', ' for', ' geeks']
['geeks', 'for', 'geeks']
['Ca', 'Ba', 'Sa', 'Fa', 'Or']
```

Control Structures

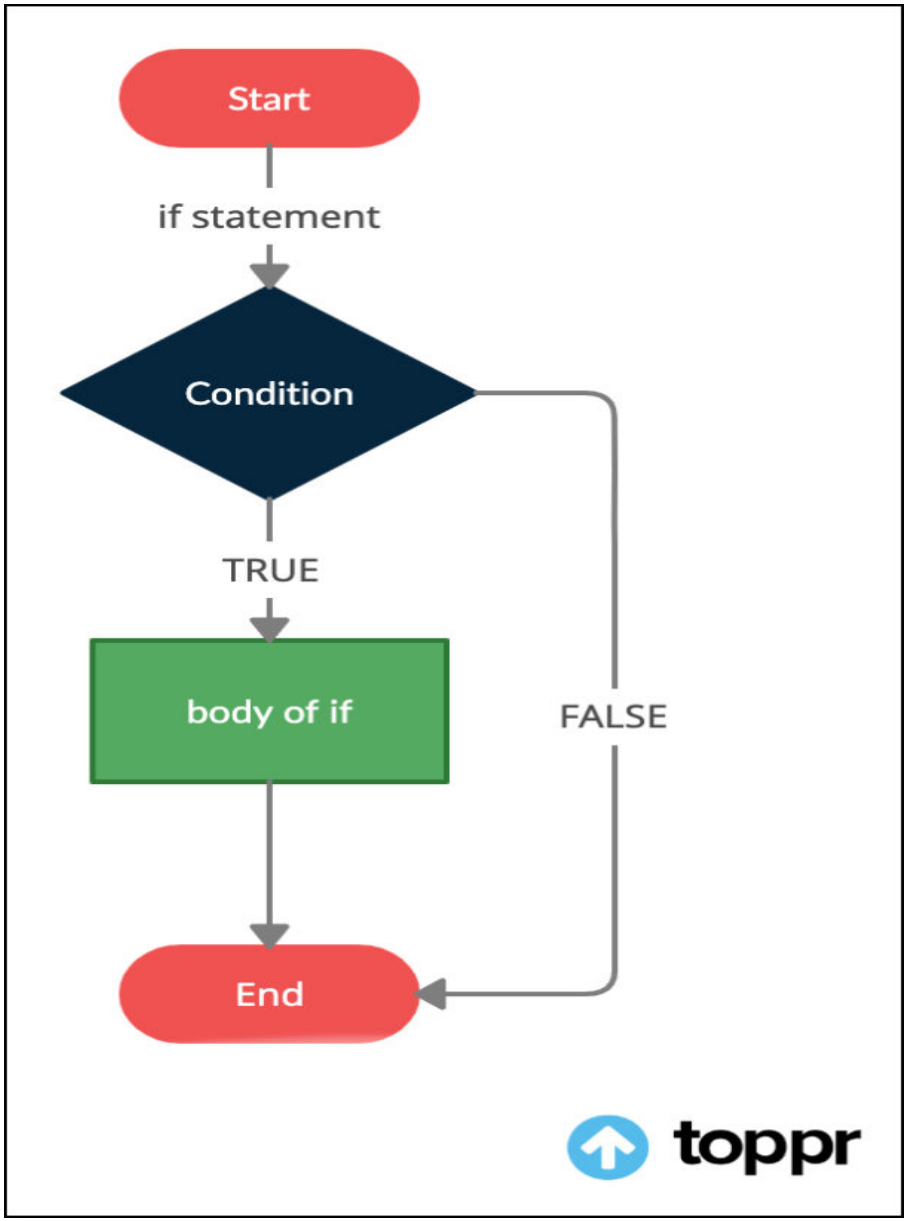
- the way to specify flow of control in programs.
- Any algorithm or program can be more clear and understood if they use self-contained modules called as logic or control structures.
- Defines the program flows direction based on certain parameters or conditions.
- Three basic types of flow of control, known as: Sequence flow, Selection flow and Iteration flow.

- Sequential Flow:
  - follows a serial or sequential flow in which the flow depends on the series of instructions given to the computer.



- Conditional Flow
  - involves a number of conditions or parameters which decides the execution of program statements.
  - ***Python if statement*** *along with its variants is used for the decision-making process.*
  - *The* ***Python if statement*** *is used to determine whether or not a specific statement or set of statements will be performed.*
- Syntax: simple if

```
if condition:
    statements(s)
```


Start
if statement
Condition
TRUE
body of if
FALSE
End
toppr

- **Note –** The indentation of the body of the loop in Python indicates the content of the if statement. It is necessary to indent the content after the ‘if statement’ has been declared. The body begins with an indentation and ends with the first un-indented line.
- **Python supports the standard mathematical logical conditions:**
  - Equals: a == b
  - Not Equals: a != b
  - Less than: a < b
  - Less than or equal to: a <= b
  - Greater than: a > b
  - Greater than or equal to: a >= b

- Example- Simple if

```
# Python program to illustrate if the number is even using if statement
num = int(input('Enter a number:'))
if (num%2 == 0): print('Number is Even')
```

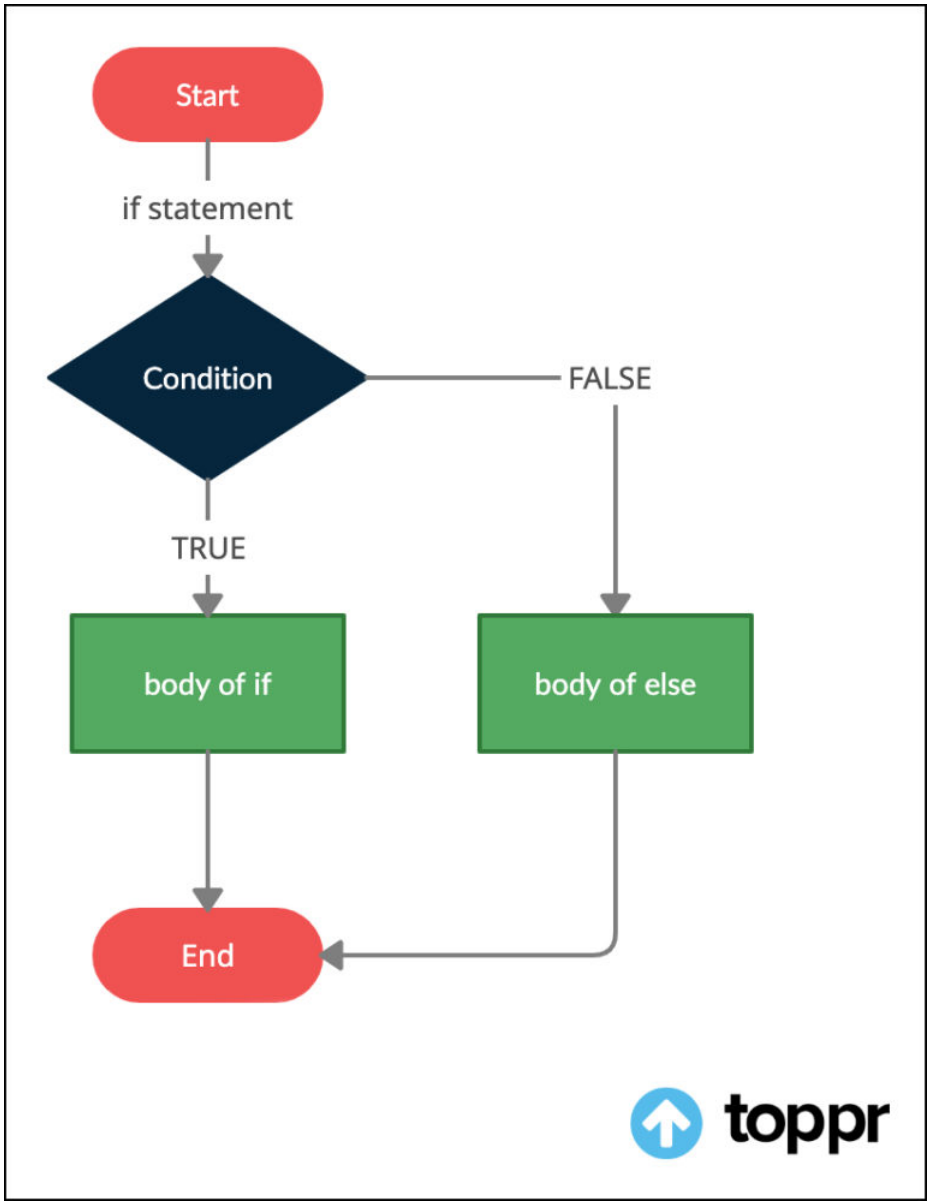
Output:

- Enter a number:10 Number is Even
- Enter a number:11
- Enter a number:

Python if…else statements

- Syntax:

```
if condition:
        statements(s)
else:
        statements(s)
```



- # Python program to compare between subject marks. history = 45 science = 57

```
if (history > science):
        print('History marks greater than Science')
else:
        print('Science marks greater than History')
```

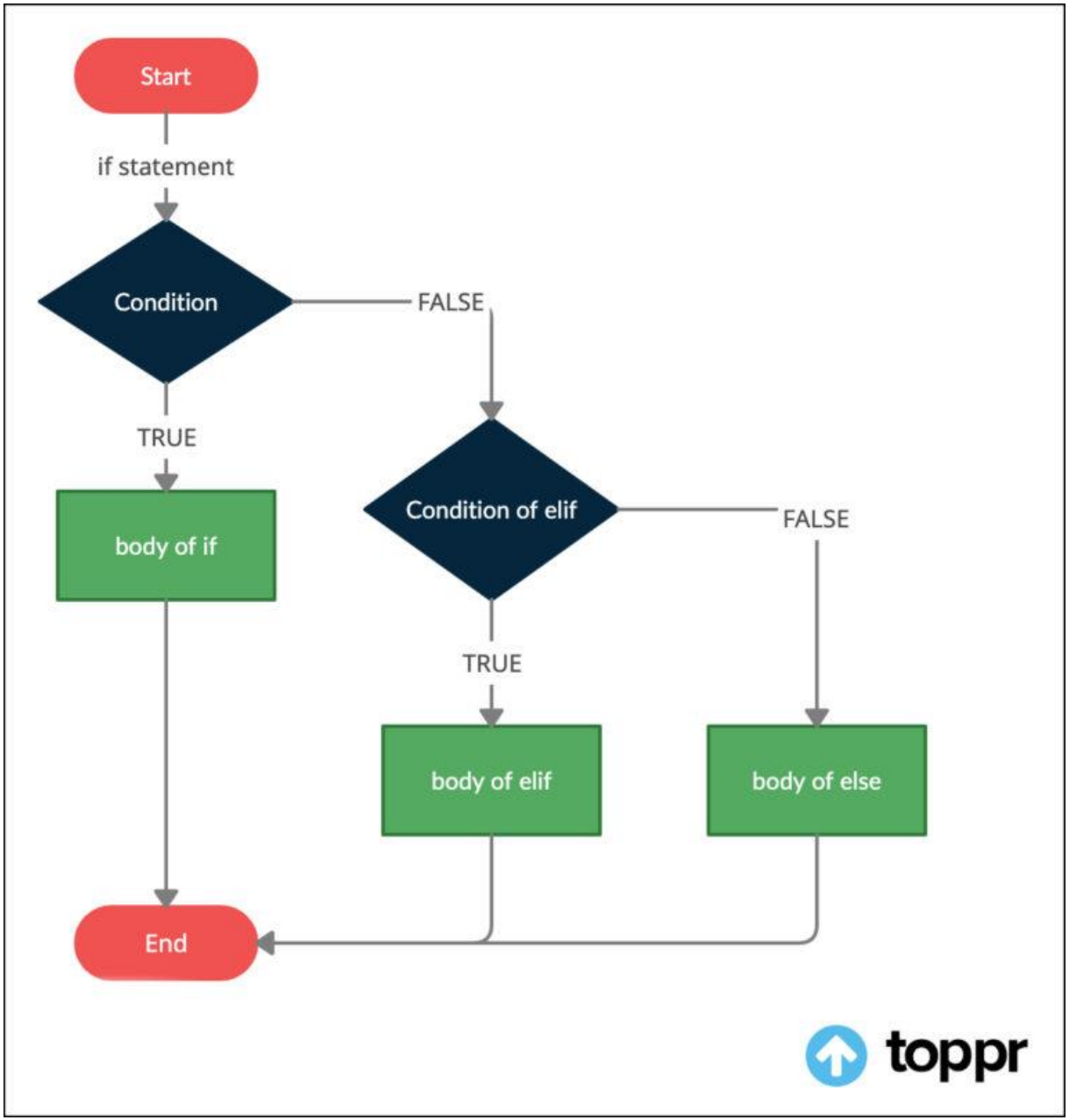
Output

```
        Science marks greater than History
```

Python if…elif…else statement

- A user can choose from a variety of alternatives here.
- Many else if conditions can be declared.
- If the first condition does not satisfy then the next condition is executed. But if this condition also does not satisfy then the next condition is executed.
- Syntax:

```
if condition1:
        statements(s) # body of if
elif condition2:
        statements(s) # body of elif
elif condition3:
        statements(s) # body of elif
else: statements(s) # body of else
```


Start
if statement
Condition
FALSE
TRUE
body of if
Condition of elif
FALSE
TRUE
body of elif
body of else
End
toppr

- # Python program to check student grade based on the marks

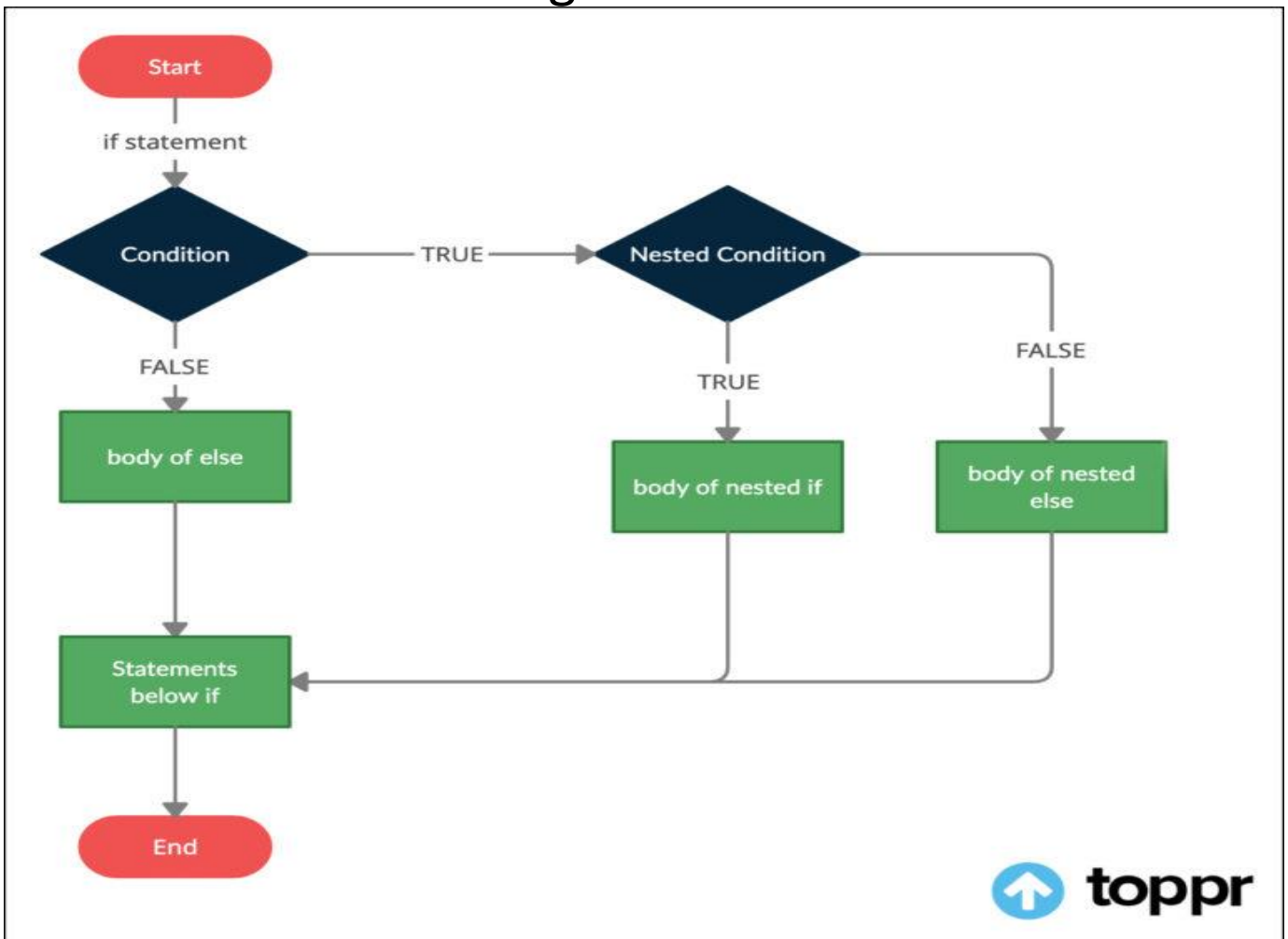
```
marks = int(input('Enter marks from 0-50: '))
if (marks<20):
        print('Student Failed')
elif (marks>20 and marks<40):
        print('Student passed with B Grade')
else:
        print('Student passed with A Grade') Output
```

- Output

```
        Enter marks from 0-50: 35
        Student passed with B Grade
```

Python Nested if statement

- We can have an if…else statement inside another if…else statement. This is called as nesting of loops.
- Python provides us with the nesting method.



- # Python program to check sign of a user entered number

```
val = int(input('Enter a number: '))
if val>=0:
        if val == 0:
                print('Number is Zero')
        else:
                print('Number is positive')
else:
        print('Number is negative')
```

- Output 1

  Enter a number: 24

  Number is positive

- Output 2

  Enter a number: 0

  Number is Zero