Lecture 2

PYTHON ESSENTIAL

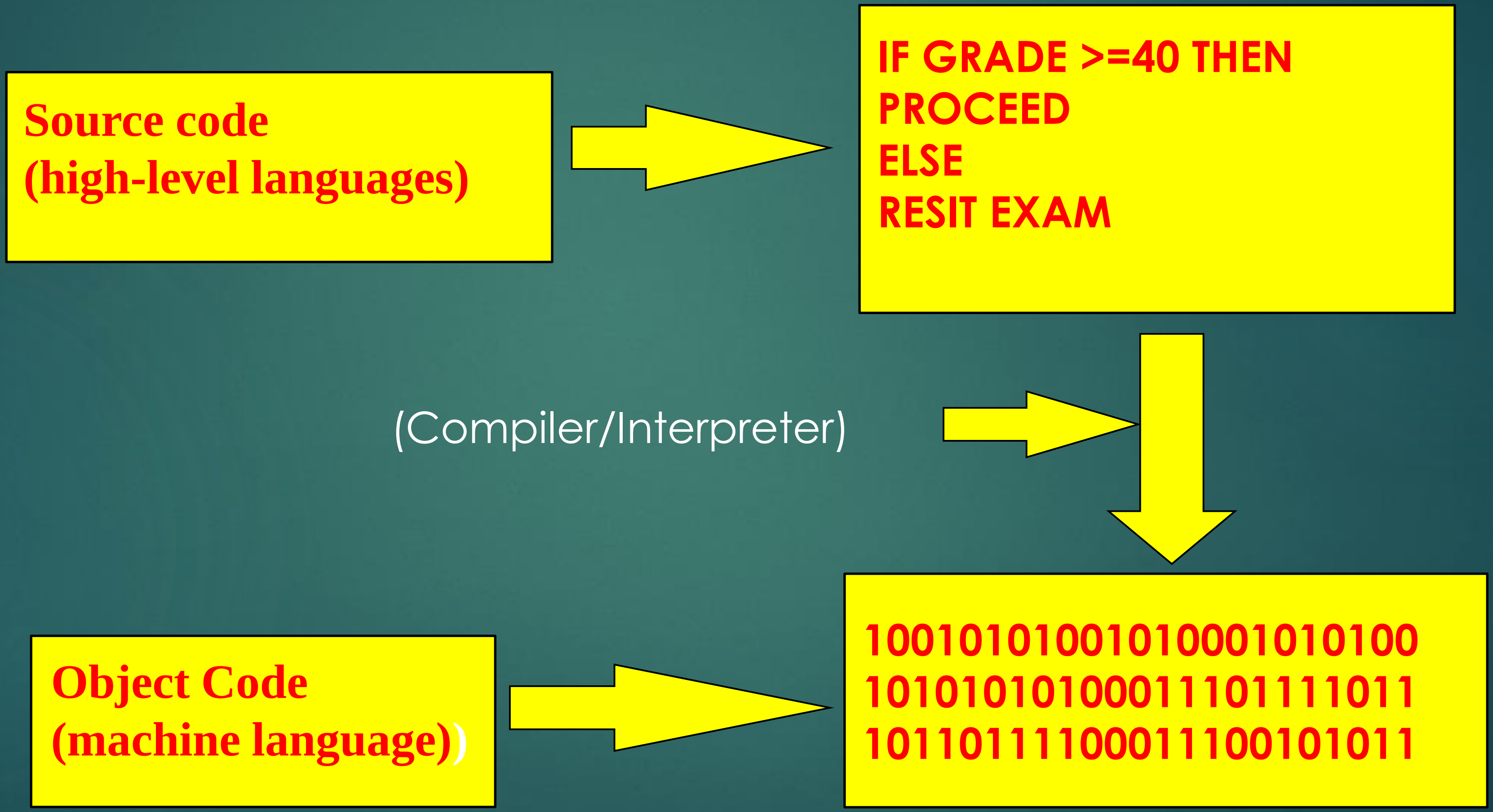
BY

MOHAMMED ABDELFATTAH ALI

Objectives

- Review
- Compilation vs. Interpretation
- Basic Syntax
- Variable Types
- Basic Operators
- Conditional Statements
- Nested IF statements

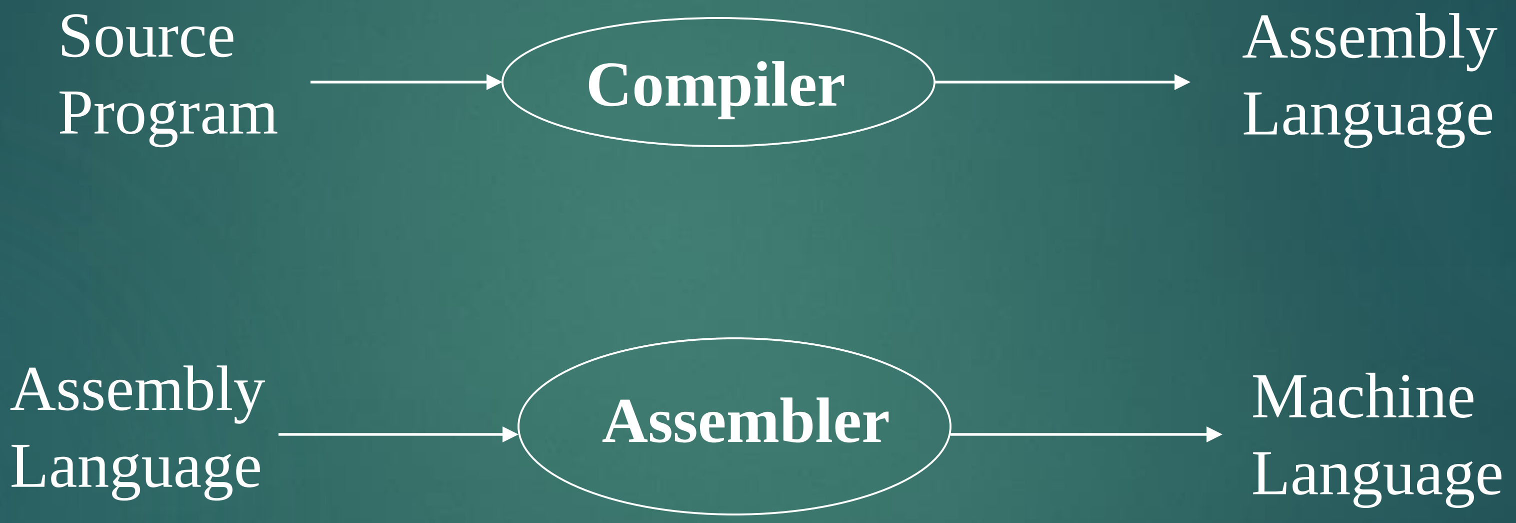
Translation process



```
IF GRADE >=40 THEN
PROCEED
ELSE
RESIT EXAM
```

```
10010101001010001010100
10101010100011101111011
10110111100011100101011
```

Compilation into Assembly Language

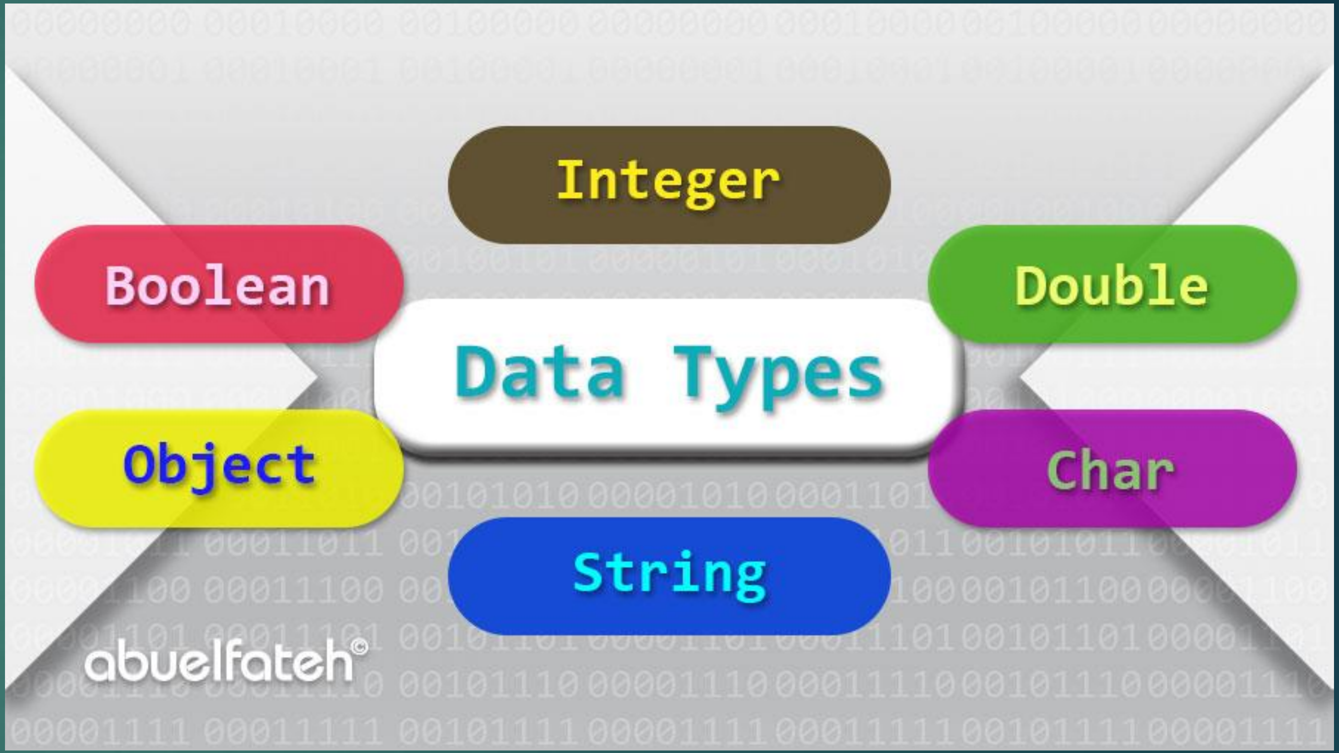


Compilation vs. Interpretation

- Compilation:
  - Syntax errors caught before running the program
  - Better performance
  - Decisions made once, at compile time
- Interpretation:
  - Better diagnostics (error messages)
  - More flexibility
  - Supports **late binding** (delaying decisions about program implementation until runtime)
    - Can better cope with PLs where type and size of variables depend on input
  - Supports creation/modification of program code on the fly (e.g. Lisp, Prolog)

Types of Data (Value, Built-In)

- Numeric Data
- Character data (alphanumeric)
- Boolean data (TRUE/FALSE)
- Other Data Types (Enum, … etc.)



Data Hierarchy

1.	**Parenthesis**	**( )**
2.	**Exponentiation**	**^**
3.	**Multiplication/Division**	*** /**
4.	**Addition/Subtraction**	**+ -**

Hierarchy of Operations Example

3 * (6 + 2) / 12 – (7 – 5) ^ 2 * 3	( ) first
= 3 * 8 / 12 – 2 ^ 2 * 3	^ next
= 3 * 8 / 12 – 4 * 3	Mult/Div (L to R)
= 24 / 12 – 4 * 3	Mult/Div (L to R)
= 2 – 12	Add/Subtr
= -10	

Basic Syntax

Basic Syntax

- If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**. However in Python version 2.4.3.

```
>>> print "Hello, Python!"

>>> print ("Hello, Python! ")
```

Python Identifiers

- A Python identifier is a name used to identify a variable, function, class, module or other object.
- An identifier starts with a letter **A to Z** or **a to z** or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

- Python does not allow punctuation characters such as **@, $, and %** within identifiers.

- Python is a case sensitive programming language.
- Thus, **Manpower** and **manpower** are two **different identifiers** in Python.

Reserved Words

- The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names.
- All the Python keywords contain lowercase letters only.

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Lines and Indentation

- Python provides no braces to indicate blocks of code for class and function definitions or flow control.
- Blocks of code are denoted by line indentation, which is rigidly enforced.
- The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
   print("True“)
else:
   print("False")
```

```
if True:
print("Answer")
print("True")
else:
print("Answer")
print("False")
```

Multi-Line Statements

- Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

```
total = item_one + \
        item_two + \
        item_three
print (total)
```

- Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

Quotation in Python

- Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.
- The triple quotes are used to span the string across multiple lines. For example, all the following are legal –

```
word = 'word'

sentence = "This is a sentence."

paragraph = """This is a paragraph. It is

made up of multiple lines and sentences."""
```

Comments in Python

- A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

```
# First comment

print("Hello, Python! ") # second comment
```

- This produces the following result –

```
Hello, Python!
```

- You can type a comment on the same line after a statement or expression –

```
name = "Madisetti" # This is again comment
```

- You can comment multiple lines as follows –

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

```
'''
This is a multiline
comment.
'''
```

Multiple Statements

- Multiple Statements on a Single Line

```
x=5; y=6; z=10; print(x+y+z)
```

- Multiple Statement Groups as Suites
- A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.
- Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example –

```
if expression :
   suite
elif expression :
   suite
else :
   suite
```

Variable Types

Variable Types

- Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Assigning Values to Variables

- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

- The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable. For example –

```
counter = 100        # An integer assignment
miles   = 1000.0     # A floating point
name    = "John"     # A string

print (counter)
print (miles)
print (name)
```

```
100
1000.0
John
```

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example –

```
a = b = c = 1

print (a)
print(b)
print(c)
```

Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location. You can also assign multiple objects to multiple variables. For example –

```
a,b,c = 1,2,"john"

print (a)
print(b)
print(c)
```

Standard Data Types

- Python has five standard data types –

    Numbers

    String

    List

    Tuple

    Dictionary

Python Numbers

```
var1 = 1
var2 = 10
```

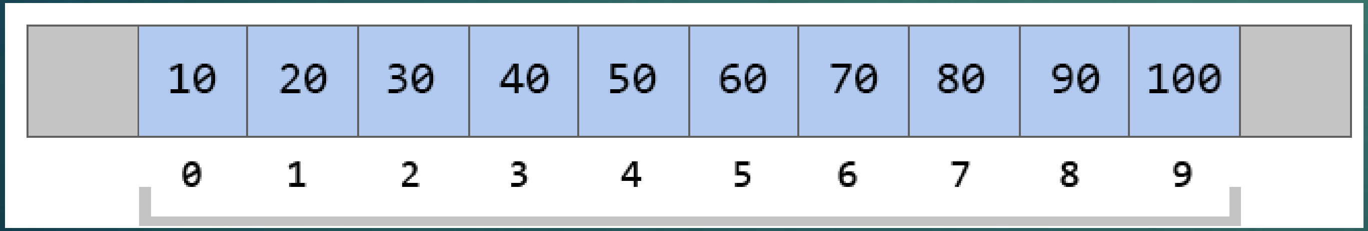
Here are some examples of numbers –

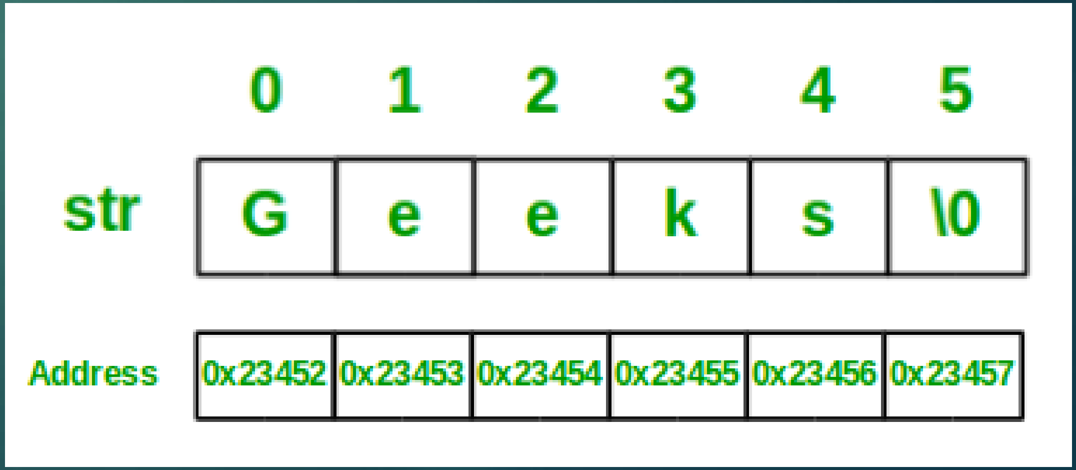
int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAEl	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J
0x69	-4721885298529L	70.2-E12	4.53e-7j

Python Strings

- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator. For example –





Python Strings

```
str = 'Hello World!'

print (str)          # Prints complete string                        Hello World!
print (str[0])       # Prints first character of the string          H
print (str[2:5])     # Prints characters starting from 3rd to 5th    llo
print (str[2:])      # Prints string starting from 3rd character     llo World!
print (str * 2)      # Prints string two times                       Hello World!Hello World!
print (str + "TEST") # Prints concatenated string                    Hello World!TEST
```

Data Type Conversion

Function & Description
int(x [,base]) Converts x to an integer. base specifies the base if x is a string.
float(x) Converts x to a floating-point number.
chr(x) Converts an integer to a character.
str(x) Converts object x to a string representation.

Basic Operators

Python Arithmetic Operators

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0

Python Comparison Operators

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	**(a == b) is not true.**
!=	If values of two operands are not equal, then condition becomes true.	**(a != b) is true.**
<>	If values of two operands are not equal, then condition becomes true.	**(a <> b) is true. This is similar to != operator.**
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	**(a > b) is not true.**
<	If the value of left operand is less than the value of right operand, then condition becomes true.	**(a < b) is true.**
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	**(a >= b) is not true.**
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	**(a <= b) is true.**

Python Assignment Operators

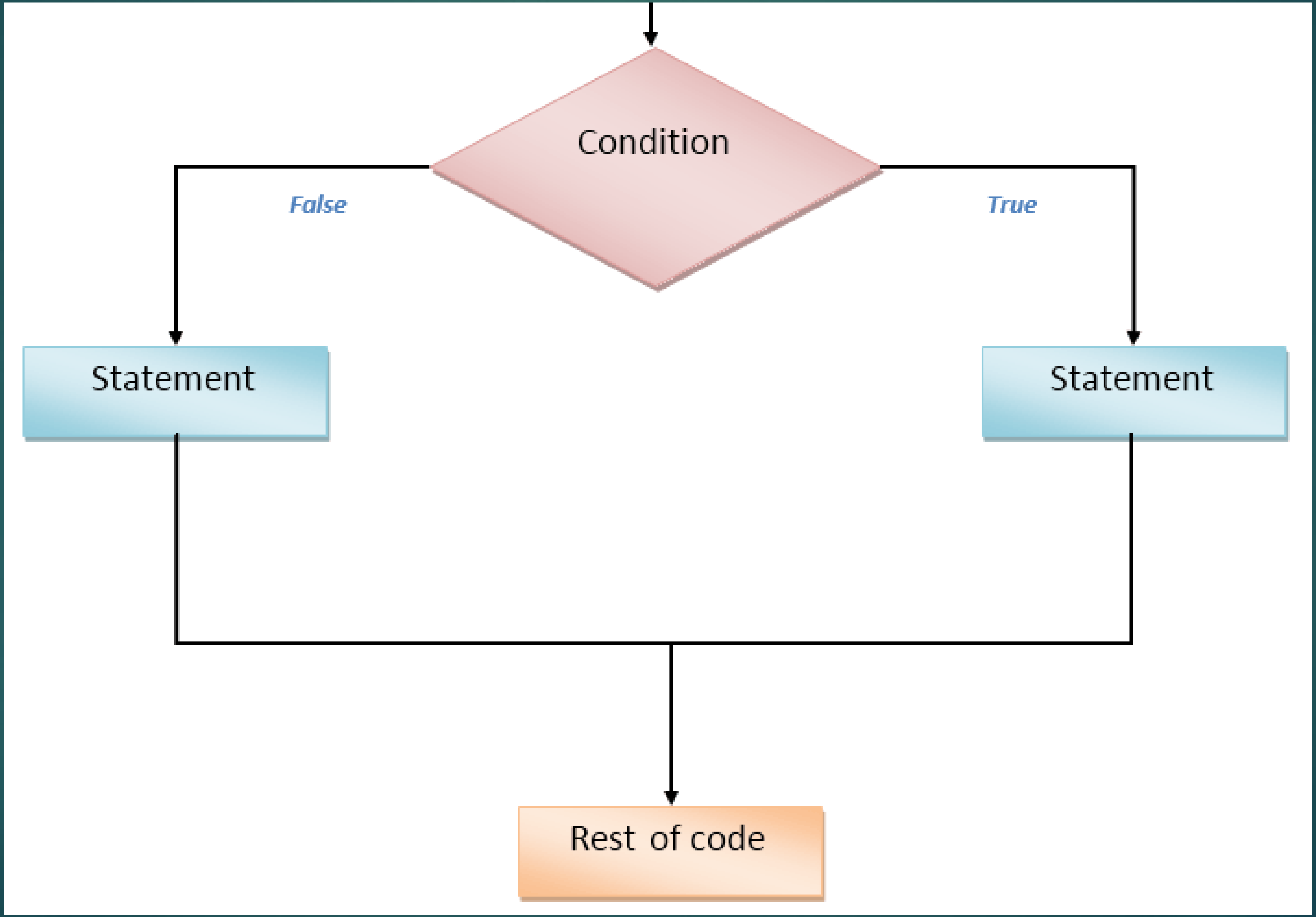
Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a

Other Operators

- Python Bitwise Operators
- Python Logical Operators
- Python Membership Operators
- Python Identity Operators

Conditional Statements

Conditional Statements

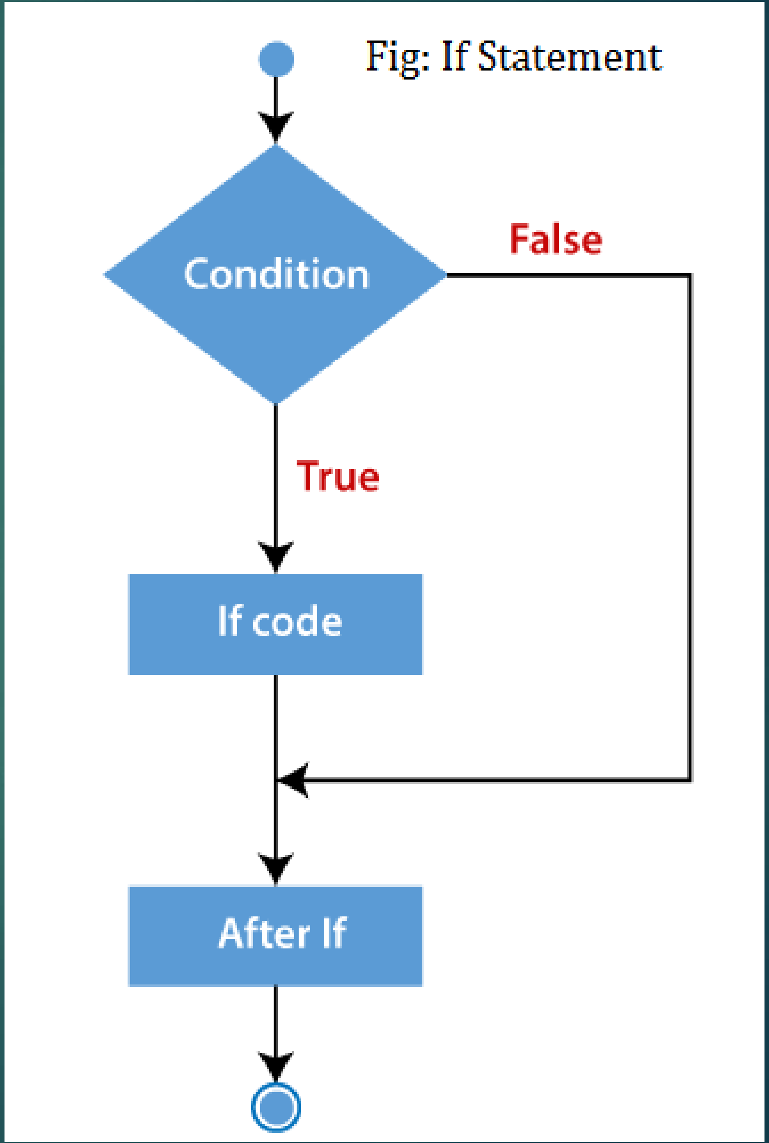


Conditional Statements

The **if** statement is a common branching structure.

– Evaluate a **boolean** expression.

- If **true**, execute some statements.
- If **false**, execute other statements.



Conditional Statements

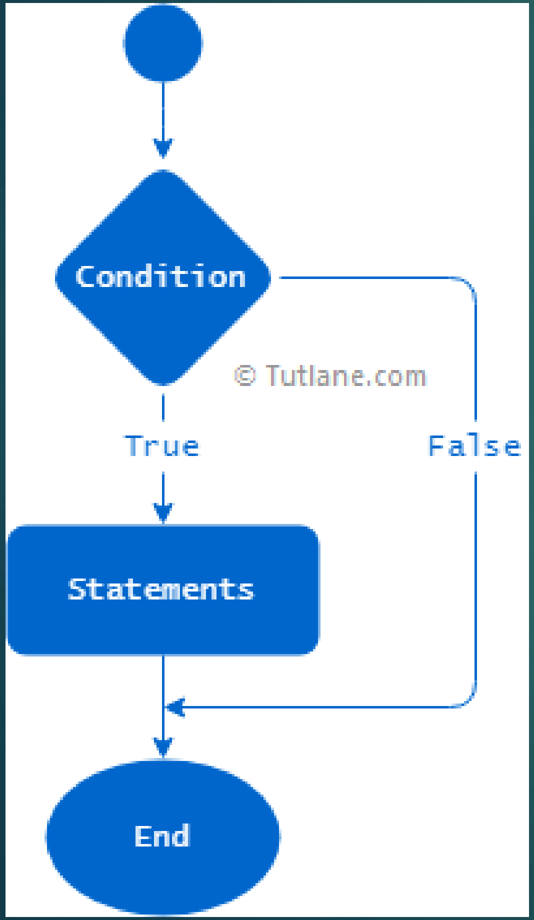
#	Statement & Description
1	An **if statement** consists of a boolean expression followed by one or more statements.
2	An **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is FALSE.
3	You can use one **if or else if** statement inside another **if or else if** statement(s).

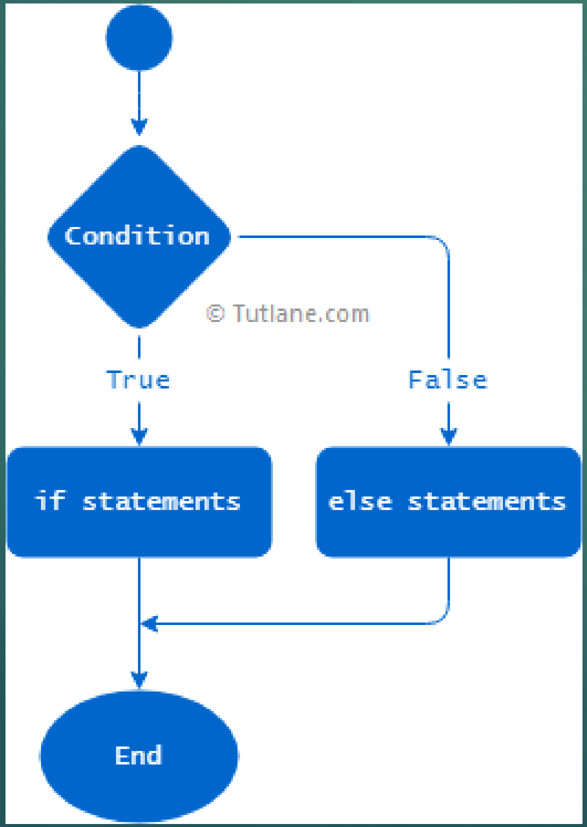
Conditional Statements

```
if expression :
   suite
```

```
if expression :
   suite
else :
   suite
```

```
if expression :
   suite
elif expression :
   suite
else :
   suite
```





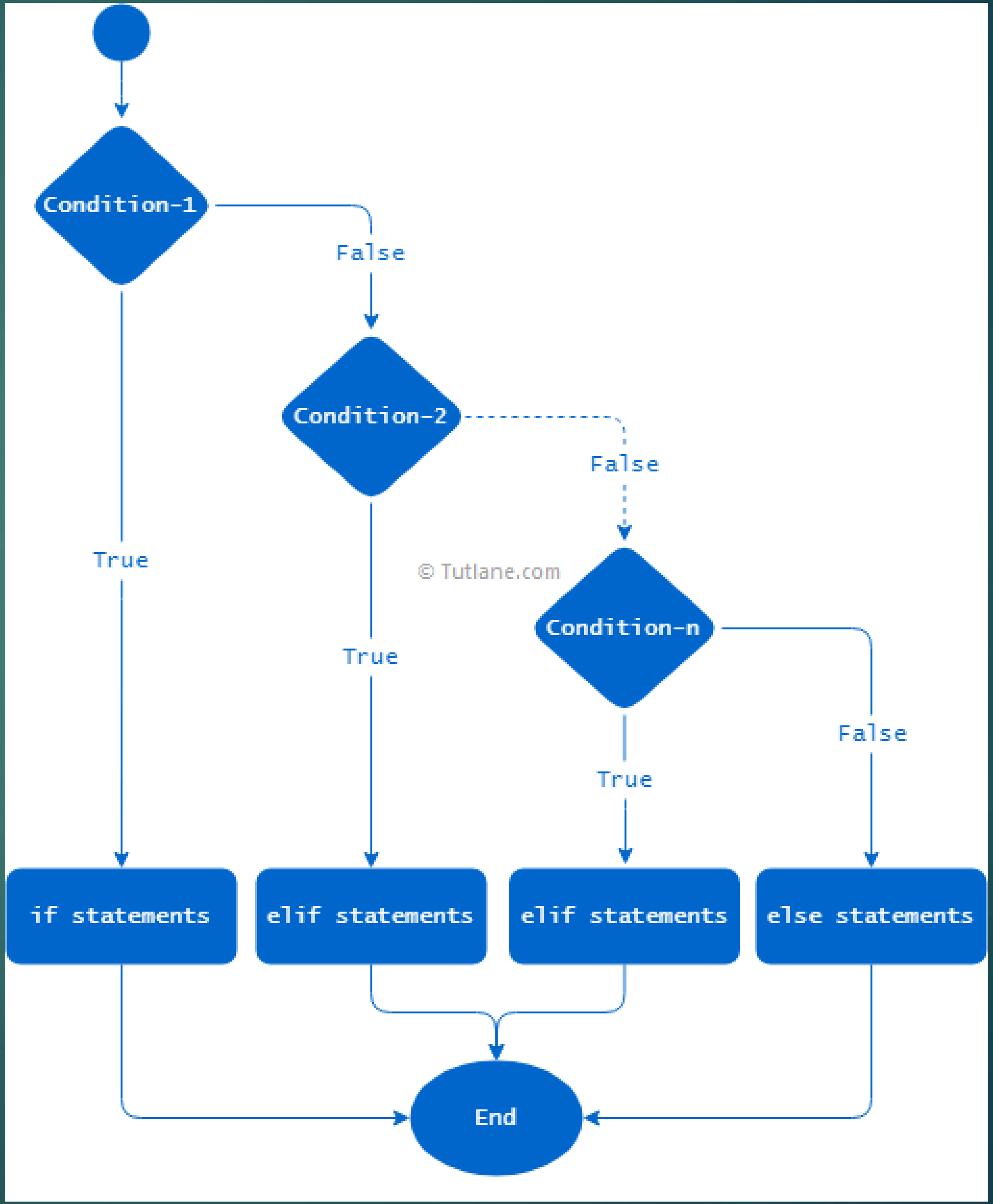
Conditional Statements

```
If (expression)
    {
    statements;
    }
else if (expression)
    {
    statements;
    }
else
    {
    statements;
    }
```

```
if expression :
    suite
elif expression :
    suite
else :
    suite
```

Conditional Statements

```
if expression :
    suite
elif expression :
    suite
else :
    suite
```



Nested IF statements

```
if expression1:

   statement(s)

   if expression2:
      statement(s)
   elif expression3:
      statement(s)
   else:
      statement(s)

else:
   statement(s)
```

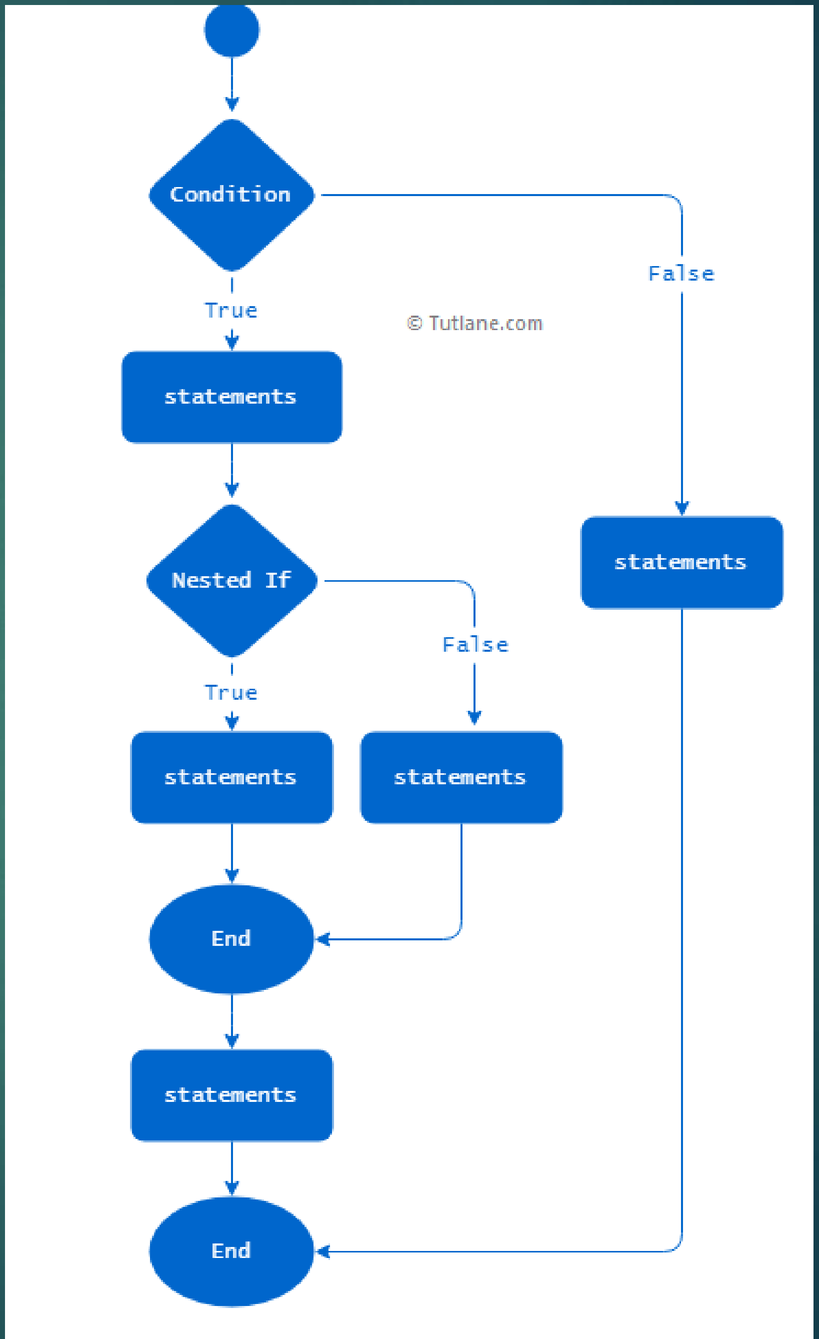
Nested IF statements

```
if expression1:

    statement(s)

    if expression2:
        statement(s)
    elif expression3:
        statement(s)
    else:
        statement(s)

else:
    statement(s)
```



Thanks