

## 01 WHAT IS A PROGRAM?

- Sequence or set of instructions in a programming language for a computer to execute.
- Programming: choosing the right data, suitable data structure and writing precise code for computer to understand
- Computers understand binary(0/1) language
- Translator(Interpreter/Compiler) converts understandable human languages to machine understandable form.
- SYNTAX - set of rules or grammar for writing computer programs

## WRITING FIRST PYTHON PROGRAM

REQUIREMENTS
• Python Installation
• Editor (.py)
• CMD (execution)

ALTERNATIVE
Integrated Development Environment (IDE) - a software application that helps programmers develop software code efficiently

OFFLINE IDEs	ONLINE IDEs
VS Code	ide.view ide.codingminutes
PyCharm	Google Colab

## 02 .PRINT()

Printing anything on screen	<code>print(message)</code>
Printing strings	<code>print("Hello World")</code> Anything enclosed within "" is a string
Printing mathematical results	<code>print(10 + 20)</code> <b>Output:</b> 30 The space between operand and operator doesn't matter

## 03. DATA TYPES

DESCRIPTION	SYNTAX
<b>Int (Integers)</b> • deals with positive or negative whole numbers. eg: 1, -4, 8, 0, 323434242 • No upper limit size in Python	<code>print(5)</code> <b>Output:</b> 5
<b>Float (Decimals)</b> • All real numbers that are not Integers. eg: .3, 0.123, -2.4 • Are 9 and 9.0 the same? Different in Python • 9.0 → Float    9 → Integer	<code>print(5.0)</code> <b>Output:</b> 5.0
<b>String</b> • Anything inside quotes • Double quotes("") or single quotes (''). • start and end with the same type of quote.	<code>print("Scaler")</code> <b>Output:</b> Scaler
<b>Boolean</b> • True or False value • Used for comparing	<code>print(True)</code> <b>Output:</b> True
<b>None</b> • Has only one value, None. • To represent nothing or an empty value, we use None.	<code>print(None)</code> <b>Output:</b> None

## TYPE()

To check data type of any object	<code>print(type("hello"))</code> <b>Output:</b> <class 'str'>
----------------------------------	---

## 04. VARIABLES

DEFINE	DEFINE
• containers to keep objects. • refers to a reserved memory location.	<code>input()</code> - takes input from user

## NAMING RULES

A combination of lowercase/ uppercase letters, digits, or an underscore.

- Lowercase letters (a to z)
- Uppercase letters (A to Z)
- Digits (0 to 9)
- Underscore (\_)

**NOTE:** Cannot begin with a digit → **1name** is invalid

## 05. OPERATORS

- symbols of operation.
- values on which operation is happening

Arithmetic Operators	operators such as +, -, *, /, //, **, % Return type of / is always floating point integers & floating = real numbers	<code>print(2+3)</code> <b>Output:</b> 5 <code>print(5/2)</code> <b>Output:</b> 2.5
Exponential Operator	$x^{**}y = x^y$	<code>print(2**3)</code> <b>Output:</b> 8
Floor Division	• $x//y = \text{floor}(x/y)$ • Returns biggest integer less than the value	<code>print(floor(5/2))</code> <b>Output:</b> 2
Modulus Operator	• $x \% y \rightarrow \text{remainder of } x/y$ • If x is '+ve' → remainder of x/y • If x is '-ve' → $y - (x \% y)$	<code>print(5%2)</code> <b>Output:</b> 1
Comparison Operators	• operators such as $>$ , $<$ , $\geq$ , $\leq$ , $\neq$ , $\neq$ • Compares values between different entities	<code>print(4 &gt; 5)</code> <b>Output:</b> False
Assignment Operator (=)	• assigns the RHS operand value to LHS operand.	<code>x = 5</code> <code>print(x)</code> <b>Output:</b> 5

## 06. CONTROL STATEMENTS

### Logical Operators

- and, or, not
- used when there are multiple conditions
- Precedence → not > and > or

DESCRIPTION	SYNTAX	EXAMPLE															
<b>and</b> • True if all conditions are True	(condition1) and (condition2)	<table border="1"> <tr><td>p</td><td>q</td><td>p and q</td></tr> <tr><td>T</td><td>T</td><td>T</td></tr> <tr><td>T</td><td>F</td><td>F</td></tr> <tr><td>F</td><td>T</td><td>F</td></tr> <tr><td>F</td><td>F</td><td>F</td></tr> </table>	p	q	p and q	T	T	T	T	F	F	F	T	F	F	F	F
p	q	p and q															
T	T	T															
T	F	F															
F	T	F															
F	F	F															
<b>or</b> • True if any one conditions are True	(condition1) or (condition2)	<table border="1"> <tr><td>p</td><td>q</td><td>p or q</td></tr> <tr><td>T</td><td>T</td><td>T</td></tr> <tr><td>T</td><td>F</td><td>T</td></tr> <tr><td>F</td><td>T</td><td>T</td></tr> <tr><td>F</td><td>F</td><td>F</td></tr> </table>	p	q	p or q	T	T	T	T	F	T	F	T	T	F	F	F
p	q	p or q															
T	T	T															
T	F	T															
F	T	T															
F	F	F															
<b>not</b> • Works with boolean operands • Inverts the current truth value	Ex: <code>print(not True)</code> => False	<table border="1"> <tr><td>p</td><td>not p</td></tr> <tr><td>T</td><td>F</td></tr> <tr><td>F</td><td>T</td></tr> </table>	p	not p	T	F	F	T									
p	not p																
T	F																
F	T																

## CONTROL STATEMENTS

- Decisions based on conditions
- execute a certain logical statement and decide whether to enable the control of the flow through a certain set of statements or not.

DESCRIPTION	SYNTAX	EXAMPLE
<b>if block</b> • Enters the block if condition is True One indent = 4/2 spaces	<code>if(something):</code> <code>print(something)</code> X=1 :	<code>x = 10</code> <code>if x &gt; 5:</code> <code>    print("Hello")</code> <b>Output:</b> Hello
<b>else block</b> • Executed when if and elif blocks fail	<code>if(something):</code> <code>print(something)</code> X=1 else: print(something else) :	<code>x = 10</code> <code>if x &lt; 5:</code> <code>    print("Hello")</code> <code>else:</code> <code>    print("Scaler")</code> <b>Output:</b> Scaler
<b>elif block</b> • Similar to if block, but executes when if block is not entered	<code>if expression1:</code> statement(s) <code>elif expression2:</code> statement(s) <code>elif expression3:</code> statement(s) else: statement(s)	<code>time = int(input())</code> <code>if time &gt; 9 and time &lt;= 12:</code> <code>    print("Good morning")</code> <code>elif time &gt; 12 and time &lt;= 17:</code> <code>    print("Good afternoon")</code> <code>elif (time &gt; 17 and time &lt;= 24):</code> <code>    print("Good night")</code> <code>else:</code> <code>    print("SOJAOO!")</code>

## 07. MATHS

- A number system is defined as a system of writing to express numbers.

### 1. BINARY NUMBER SYSTEM

- Represented in the form of 0/1
- base 2 number system
- Any number less than  $2^n$  can be represented by n digits
- Example: 1011 is a number in binary number system

### 2. DECIMAL NUMBER SYSTEM

- Represented by digits from 0 to 9
- base of 10 because it uses ten digits from 0 to 9
- Example: 987 is a number in decimal number system

### 3. OCTAL NUMBER SYSTEM (BASE 8)

### 4. HEXADECIMAL NUMBER SYSTEM (BASE 16)

#### CONVERSIONS

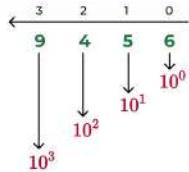
##### DECIMAL TO BINARY

- Representation:  $(value)_{base}$

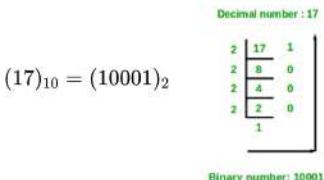
Example:  $(209)_{10} \rightarrow 209$  in base 10

- From left to right, the digits represent increasing powers of 10 starting from 0.

$$(9456)_{10} = 6 * 10^0 + 5 * 10^1 + 4 * 10^2 + 9 * 10^3$$



- Progressively divide by the base of the number system you want to convert it into and note down the remainder from end to start.



##### BINARY TO DECIMAL

- multiply powers of 2

## 08. RANGE

- Represents a continuous stretch of numbers
- Square brackets [] means the terminal values are inclusive (end-start+1 values)  
Example:  $[10, 15] = [10, 11, 12, 13, 14, 15]$
- in round brackets (), the terminal values are exclusive. (end-start-1 values)  
Example:  $(10, 15) = (11, 12, 13, 14)$

DESCRIPTION	SYNTAX	EXAMPLE
<b>bin()</b> • Provides binary value for any decimal number • Represented by '0b' prefix	<b>bin(decimal number)</b>	<code>print(bin(56))</code> <b>Output:</b> 0b111000
<b>log()</b> • returns the natural logarithm of a number	<b>math.log</b> (number, base)	<code>print(math.log(14, 5))</code> <b>Output:</b> 1.6397385131955606

## 09. ITERATION

- a sequence of instructions or code being repeated until a specific end result is achieved.

DESCRIPTION	SYNTAX	EXAMPLE
<b>while loop</b> • we can execute a set of statements as long as a condition is true.	3 parts for executing while loops <b>Initialisation</b> • Where to start from <b>Condition</b> • Termination of loop <b>Updation</b> • Update the iterator variable	<code># Initialisation i = 1 # Condition while i &lt; 6:     print(i)     # Updation     i += 1</code> <b>Output:</b> 1 2 3 4 5

### for loop

Need: While loop will run infinitely when we do not give an exit condition used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string)

for every element in range(start, end):  
Do Something

```
for i in range(1,6):
    print(i)
```

```
Output: 1
2
3
4
5
```

**range(number)**  
• includes start and excludes end  
• start 0 by default  
Does not work with float

```
# create range
range(5)
Output: range(0,5)

print(list(range(5)))
Output: [0,1,2,3,4]

print(list(range(2,5)))
Output: [2,3,4]

print(list(range(-9, -1)))
Output: [-9, -8, -7, -6, -5, -4, -3, -2]

print(list(range(2,10, 2)))
Output: [2,4,6,8]
```

## 11. JUMP STATEMENTS

DESCRIPTION	SYNTAX	EXAMPLE
<b>pass</b> • Acts as a placeholder • Empty block	<b>if something:</b> <b>pass</b>	<code>for i in range(5):     if i == 3:         pass     print(i)</code> <b>Output:</b> 0 1 2 3 4
<b>continue</b> • Code after this is ignored	<b>if something:</b> <b>continue</b>	<code>for i in range(5):     if i == 3:         continue     print(i)</code> <b>Output:</b> 0 1 2 4
<b>break</b> • Loop terminates	<b>if something:</b> <b>break</b>	<code>for i in range(5):     if i == 3:         break     print(i)</code> <b>Output:</b> 0 1 2

## 12. PATTERN PRINTING

- Nested loops used
- Outer loop runs for each row, inner loop runs for number of columns

#### EXAMPLE:

For n=4, the pattern is

```
n = int(input())
for i in range(n):
    for j in range(i+1):
        print("*", end = "")
    print()
```

\*

\*\*

\*\*\*

\*\*\*\*