

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

VS Code

PyCharm

ONLINE IDEs

ide.view

ide.codingminutes

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> Output: 30 The space between operand and operator doesn't matter

03. DATA TYPES

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

TYPE()

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

04. VARIABLES

DEFINE	DEFINE
<ul style="list-style-type: none"> • 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> output: 5 <code>print(5/2)</code> Output: 2.5
Exponential Operator	$x**y = x^y$	<code>print(2**3)</code> Output: 8
Floor Division	<ul style="list-style-type: none"> • $x//y = \text{floor}(x/y)$ • Returns biggest integer less than the value 	<code>print(floor(5/2))</code> Output: 2
Modulus Operator	<ul style="list-style-type: none"> • $x \% y \rightarrow$ remainder of x / y • If x is '+ve' → remainder of x / y • If x is '-ve' → $y - (x \% y)$ 	<code>print(5%2)</code> Output: 1
Comparison Operators	<ul style="list-style-type: none"> • operators such as >, <, >=, <=, ==, != • Compares values between different entities 	<code>print(4 > 5)</code> Output: False
Assignment Operator (=)	• assigns the RHS operand value to LHS operand.	<code>x = 5</code> <code>print(x)</code> Output: 5

06. CONTROL STATEMENTS

Logical Operators

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

- (condition1) and (condition2)
- (condition1) or (condition2)
- not(condition1)

DESCRIPTION	SYNTAX	EXAMPLE															
and <ul style="list-style-type: none">• True if all conditions are True	(condition1) and (condition2)	<table><tr><th>p</th><th>q</th><th>p and q</th></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															
or <ul style="list-style-type: none">• True if any one conditions are True	(condition1) or (condition2)	<table><tr><th>p</th><th>q</th><th>p or q</th></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															
not <ul style="list-style-type: none">• Works with boolean operands• Inverts the current truth value	Ex: print(not True) >> False	<table><tr><th>p</th><th>not p</th></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
if block <ul style="list-style-type: none"> • Enters the block if condition is True One indent = 4/2 spaces 	if (something): <code>print(something)</code> <code>X = 1</code> :	<code>x = 10</code> <code>if x > 5:</code> <code>print("Hello")</code> Output: Hello
else block <ul style="list-style-type: none"> • Executed when if and elif blocks fail 	if (something): <code>print(something)</code> <code>X = 1</code> else: <code>print(something else)</code> :	<code>x = 10</code> <code>if x < 5:</code> <code>print("Hello")</code> else: <code>print("Scaler")</code> Output: Scaler
elif block <ul style="list-style-type: none"> • Similar to if block, but executes when if block is not entered 	if expression1: statement(s) elif expression2: statement(s) elif expression3: statement(s) else: statement(s)	<code>time = int(input())</code> <code>if time > 9 and time <= 12:</code> <code>print("Good morning")</code> <code>elif time > 12 and time <= 17:</code> <code>print("Good afternoon")</code> <code>elif (time > 17 and time <= 24):</code> <code>print("Good night")</code> else: <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.

Decimal number : 17

2	17	1
2	8	0
2	4	0
2	2	0
	1	

$$(17)_{10} = (10001)_2$$

Binary number: 10001

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
bin() <ul style="list-style-type: none"> Provides binary value for any decimal number Represented by '0b' prefix 	<code>bin(decimal number)</code>	<pre>print(bin(56))</pre> Output: 0b111000
log() <ul style="list-style-type: none"> returns the natural logarithm of a number 	<code>math.log (number, base)</code>	<pre>print(math.log(14,5))</pre> Output: 1.6397385131955606

09. ITERATION

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

DESCRIPTION	SYNTAX	EXAMPLE
while loop <ul style="list-style-type: none"> we can execute a set of statements as long as a condition is true. 	3 parts for executing while loops Initialisation • Where to start from Condition • Termination of loop Updation • Update the iterator variable	<pre># Initialisation i = 1 # Condition while i < 6: print(i) # Updation i += 1</pre> Output: 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()

- includes start and excludes end
 - start 0 by default
- Does not work with float

`range(number)`

`range(start, end, step)`

Step → the jumps from start to next number and so on
By default it is 1

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
pass <ul style="list-style-type: none"> Acts as a placeholder Empty block 	<pre>if something: pass</pre>	<pre>for i in range(5): if i == 3: pass print(i)</pre> Output: 0 1 2 3 4
continue <ul style="list-style-type: none"> Code after this is ignored 	<pre>if something: continue</pre>	<pre>for i in range(5): if i == 3: continue print(i)</pre> Output: 0 1 2 4
break <ul style="list-style-type: none"> Loop terminates 	<pre>if something: break</pre>	<pre>for i in range(5): if i == 3: break print(i)</pre> Output: 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()
```

```
*
* *
* * *
* * * *
```