Introduction to Python

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

print()

Printing anything on screen	print(message)

Printing strings	print(“Hello World”) Anything enclosed within “” is a string
Printing mathematical results	print(10 + 20) > 30 The space between operand and operator doesn’t matter

Data Types

- The type of data we are dealing with

Description	Example
Int (Integers) - deals with **positive** or **negative** whole numbers. eg: 1, -4, 8, 0, 323434242 - No upper limit size in Python	print(5) >> 5
Float (Decimals) - 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	print(5.0) >> 5.0
String ● Anything inside quotes ● Double quotes("") or single quotes (''). ● start and end with the same type of quote.	print(“Scaler”) >> Scaler

Boolean - True or False value - Used for comparing	print(True) >> True
None ● Has only one value, None. ● To represent nothing or an empty value, we use None.	print(None) >> None

`type()` - To check data type of any object

```
print(type("hello"))
>> <class 'str'>
```

Variables

- containers to keep objects.
- refers to a reserved memory location.

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 (_)

Cannot begin with a digit → 1name is invalid

Taking Input

input() - takes input from user

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
		`print(2+3)`
		`>> 5`
		`print(5/2)`
		`>> 2.5`

Exponential Operator	- x**y = x^y	`print(2**3)`
		`>> 8`

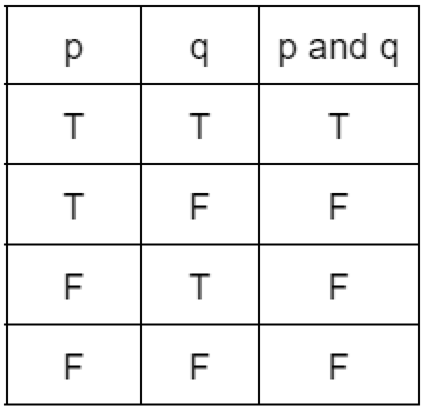
Floor Division	- x//y = floor(x/y)
	- Returns biggest integer less than the value
		`print(floor(5/2))`
		`>> 2`

Modulus Operator	- x % y -> remainder of x / y
	- If x is **'+ve'** -> remainder of x / y
	- If x is **'-ve'** -> y - (x % y)
		`print(5%2)`
		`>> 1`

Comparison Operators	- operators such as >, <, >=, <=, ==, !=
	- Compares values between different entities
		`print(4 > 5)`
		`>> False`

Assignment Operator (**=**)	- assigns the RHS operand value to LHS operand.	`x = 5`
		`print(x)`
		
		`>> 5`

Control Statements

Description	Syntax	Example
Logical Operators - and, or, not - used when there are multiple conditions - Precedence → not > and > or	-(condition1) and (condition2) - (condition1) or (condition2) - not(condition1)	
and - True if all conditions are True	(condition1) and (condition2)	

p	q	p and q
T	T	T
T	F	F
F	T	F
F	F	F

or - True if any one conditions are True	(condition1) or (condition2)	p · q · p or q T · T · T T · F · T F · T · T F · F · F
not - Works with boolean operands - Inverts the current truth value	- Ex: print(not True) >> False	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
- *Enters the block if condition is True*
One indent = 4/2 spaces
	if(something):
	    print(something)
	    X = 1
	    :
	    :
		`x = 10`
		`if x > 5:`
		  `print(“Hello”)`
		
		`>> Hello`

else block
- Executed when if and elif blocks fail
	if(something):
	    print(something)
	    X = 1
	else:
	    print(something else)
		`x = 10`
		`if x < 5:`
		  `print(“Hello”)`
		`else:`

	:
	:
		`print(“Scaler”)`
		
		`>> Scaler`

elif block
- 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)
		`time = int(input())`
		`if time > 9 and time <= 12:`
		`  print("Good morning")`
		`elif time > 12 and time <= 17:`
		`  print("Good afternoon")`
		`elif (time > 17 and time <= 24):`
		`  print("Good night")`
		`else:`
		`  print("SOJAOO!")`

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

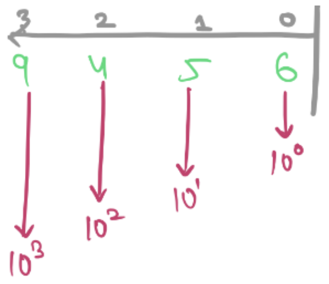
3. **Octal number system (Base 8)**

4. **Hexadecimal Number system (Base 10)**

Representation: $(\text{value})_{\text{base}}$

Example→ $(209)_{10}$ → 209 in base 10

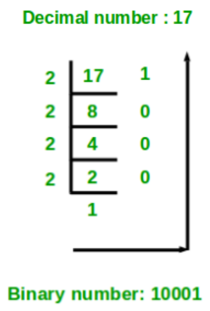
- From left to right, the digits represent increasing powers of 10 starting from 0.
- $(9456)_{10} = 6 * 10^0 + 5 * 10^1 + 4 * 10^2 + 9 * 10^3$



Conversions
Decimal to binary

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

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



Binary to Decimal

- multiply powers of 2

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()
- Provides binary value for any decimal number
- Represented by '0b' prefix
	bin(decimal number)	`print(bin(56))`
		
		`>> 0b111000`

log()
- returns the natural logarithm of a number
	math.log(number, base)	`print(math.log(14,5))`
		
		`>> 1.6397385131955606`

Iteration

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

Description	Syntax	Example

while loop
- 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
		`# Initialisation`
		`i = 1`
		
		`# Condition`
		`while i < 6:`
		`   print(i)`
		
		`   # Updation`
		`   i += 1`
		
		`>> 1`
		`2`
		`3`
		`4`
		`5`

for loop
 	
Need: While loop will run infinitely when we do not
	for every element in range(start, end):
	Do SOmething
		`for i in range(1,6):`
		`    print(i)`
		
		`>> 1`

give an exit condition
 	
- used for iterating *over a sequence (that is either a* ***list****, a* ***tuple****, a* ***dictionary****, a* ***set****, or a* ***string****)*
		`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)`
		`>> range(0,5)`
		
		`print(list(range(5)))`
		`>> [0,1,2,3,4]`
		
		`print(list(range(2,5)))`
		`>> [2,3,4]`
		
		`print(list(range(-9, -1)))`
		`>> [-9, -8, -7, -6, -5, -4, -3, -2]`
		
		`print(list(range(2,10, 2)))`
		`>>[2,4,6,8]`

Jump Statements

Description	Syntax	Example

pass
- Acts as a placeholder
- Empty block
	if something:
	  pass
		`for i in range(5):`
		`    if i == 3:`
		`        pass`
		`    print(i)`
		`>> 0 1 2 3 4`

continue
- Code after this is ignored
	if something:
	  continue
		`for i in range(5):`
		`    if i == 3:`

		`        continue`
		`    print(i)`
		`>> 0 1 2 4`

break
- Loop terminates
	if something:
	break
		`for i in range(5):`
		`    if i == 3:`
		`        break`
		`    print(i)`
		`>>0`
		`1`
		`2`

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()
```