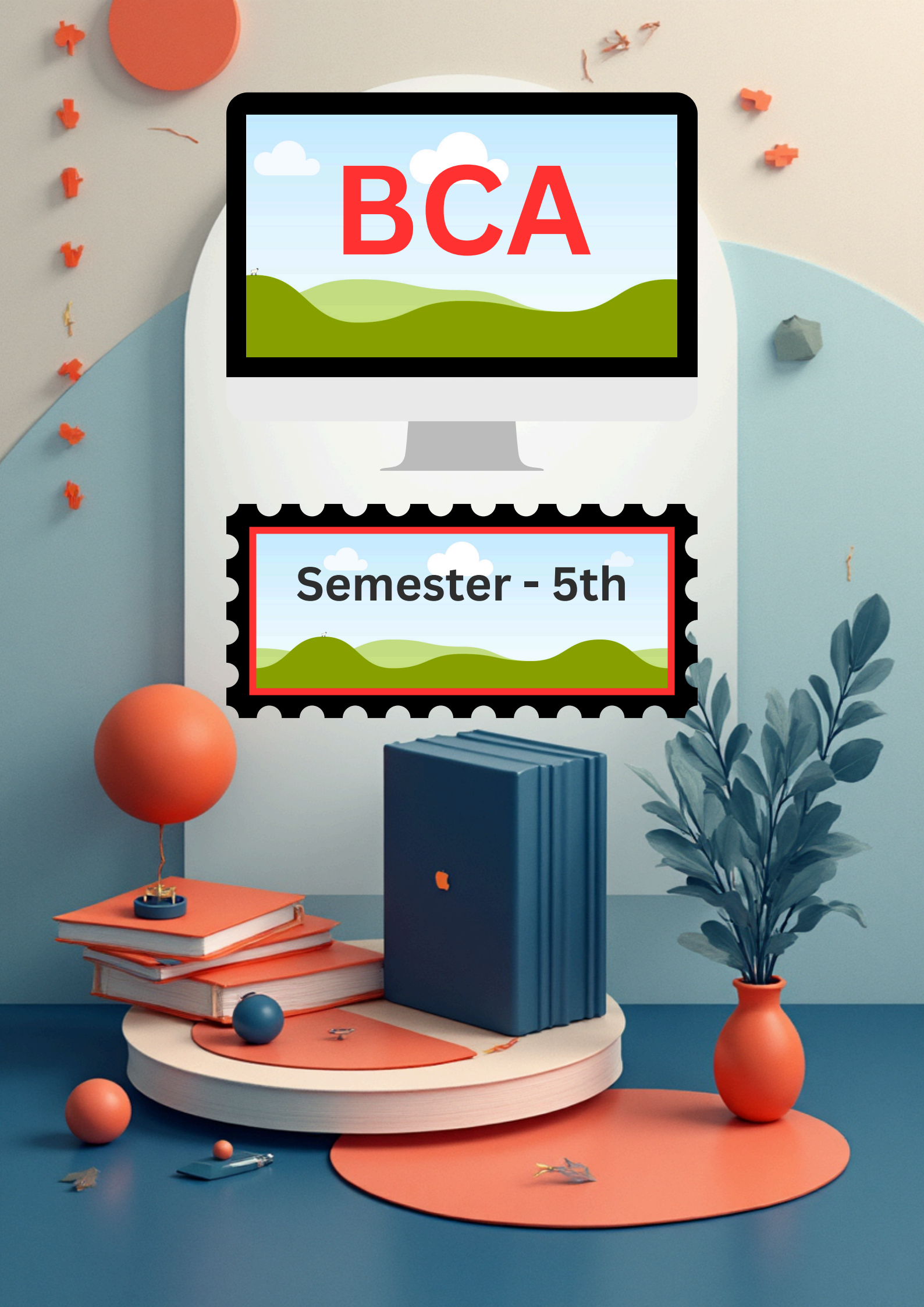
BCA
Semester - 5th

Artificial Intelligence through Python

Notes - 1

Contents

1.1 Installation and Working with Python 1.2 Understanding Python variables 1.3 Python basic Operators 1.4 Understanding python blocks 1.5 Declaring and using Numeric data types: int, float, complex 1.6 Using string data type and string operations 1.7 Defining list and list slicing 1.8 Use of Tuple data type 1.9 Conditional blocks using if, else and elif 1.10 Simple for loops in python 1.11 For loop using ranges, string, list and dictionaries 1.12 Use of while loops in python 1.13 Loop manipulation using pass, continue, break and else 1.14 Programming using Python conditional and loops block

Installation and Working with Python

Installation Steps

Go to Python website:
Visit https://www.python.org

Download Python:
Choose the latest version (like Python 3.x) suitable for your operating system (Windows, macOS, or Linux).

Install Python:
Click on “Install Now”
Make sure to tick the box “Add Python to PATH”

Verify Installation:
Open Command Prompt (CMD) or Terminal → type
python --version
It will show the installed version.

Working with Python

You can write Python code in:

a. IDLE (Python’s built-in editor): Comes automatically with Python.
b. Jupyter Notebook: Used in data science and AI for writing code interactively.
c. VS Code / PyCharm: Professional code editors.

Running Python Code

You can write and run a simple program like:
print("Hello, AI World!")

→ Output:
Hello, AI World!

Understanding Python Variables

Definition

A variable is a name used to store a value in memory.
It acts like a container for data.

Example
x = 10
name = "Python"

Here,
x stores integer 10
name stores string "Python"

Rules for Variables

1. Must start with a letter or underscore (_).
2. ✅ age, _count
3. ❌ 1num
4. Can contain letters, digits, underscores.
5. Case sensitive: Age and age are different.
6. No spaces allowed in variable names.

Dynamic Typing

Python is dynamically typed, meaning you don’t need to declare variable types.

```
a = 10      # int
a = "AI"    # string
```

Variable type can change anytime.

Python Basic Operators

Operators are symbols that perform operations on variables and values.

Type	Operators	Example	Explanation
Arithmetic	+ - * / % // **	a + b	Basic math
Comparison	⊗	a > b	Compares values
Assignment	⊗	a += 2	Assigns or updates
Logical	and, or, not	a > 10 and b < 5	Combines
Bitwise	`&	^ ~ << >>`	a & b
Membership	in, not in	"AI" in topic	Checks if present
Identity	is, is not	a is b	Checks memory

Example

```
a = 10
b = 3
print(a + b)   # 13
print(a // b)  # 3 (Floor division)
print(a ** b)  # 1000 (Power)
```

Understanding Python Blocks

Definition

A block is a group of statements that belong together — like inside an if, for, or function.

Important: Indentation

- Python uses indentation (spaces or tabs) to define blocks, not curly braces {} like C or Java.
- Proper indentation is mandatory.

Example

```
if True:
    print("Inside block")
    x = 10
print("Outside block")
```

➡️ The first two lines inside if are part of one block.
Note:
If indentation is wrong, Python gives IndentationError.

Declaring and Using Numeric Data Types: int, float, complex

Numeric Data Types in Python
Python has 3 main numeric types:

Type	Description	Example
int	Integer numbers	a = 10
float	Decimal or fractional	b = 10.5
complex	Numbers with real +	c = 3 + 4j

Examples

```
# Integer
x = 5
print(type(x))  # <class 'int'>

# Float
y = 5.25
print(type(y))  # <class 'float'>
```

```
# Complex
z = 3 + 2j
print(type(z)) # <class 'complex'>


Operations
a = 5
b = 2.0
c = 3 + 4j

print(a + b)  # 7.0 (int + float = float)
print(a + c)  # (8+4j)
```

✅ *Summary (For Exam Quick Revision)*

Topic	Key Points
Python Installation	Download → Install → Add to PATH
Variables	Containers for data, dynamically
Operators	Arithmetic, Comparison, Logical,
Blocks	Defined by **indentation**, used in if, for, function
Numeric Types	int, float, complex — support math

Using String Data Type and String Operations

👉 **What is a String?**

A string is a sequence of characters enclosed in single (‘ ’), double (“ ”), or triple (“”” ”””) quotes.

Example
name = "Python"
message = 'Artificial Intelligence'
note = """AI is the future"""

String Indexing
Each character in a string has a position (index):
P y t h o n
0 1 2 3 4 5

name = "Python"
print(name[0]) # P
print(name[-1]) # n

String Operations

Operation	Example	Output	Meaning
Concatenation	"AI" + "Python"	"AIPython"	Join two strings
Repetition	"AI" * 3	"AIAIAI"	Repeat string
Length	len("Python")	6	Total characters
Slicing	"Python"[0:3]	"Pyt"	Part of string
Membership	'A' in "AI"	TRUE	Check if present
Iteration	for c in "AI": print(c)	AI	Loop through

Common String Methods

Method	Use	Example
upper()	Converts to uppercase	"python".upper() →
lower()	Converts to lowercase	"AI".lower() → "ai"
title()	First letter capital	"hello ai".title()
replace(old, new)	Replace text	"AI".replace("A","P") →
split()	Split into list	"AI Python".split() →
strip()	Removes spaces	" AI ".strip() → "AI"

Defining List and List Slicing

👉 What is a List?

A list is an ordered, mutable (changeable) collection of items.
It can store different types of data (int, string, float, etc.).

Example

```
my_list = [10, 20, "AI", 5.5]
print(my_list)
```

Accessing Elements

```
print(my_list[0])   # 10
print(my_list[-1])  # 5.5
```

Modifying Lists

```
my_list[1] = 50
print(my_list)  # [10, 50, "AI", 5.5]
```

List Operations

Operation	Example	Output
Append	my_list.append(100)	Adds item at end
Insert	my_list.insert(1,	Adds at specific index
Remove	my_list.remove("AI")	Removes value
Pop	my_list.pop()	Removes last item
Length	len(my_list)	Returns size
Sort	list.sort()	Sorts list

List Slicing
Slicing = extracting part of a list.

```
numbers = [10, 20, 30, 40, 50, 60]
print(numbers[1:4])   # [20, 30, 40]
print(numbers[:3])    # [10, 20, 30]
print(numbers[::2])   # [10, 30, 50]
```

Use of Tuple Data Type

👉 What is a Tuple?
A tuple is similar to a list, but immutable (cannot be changed).

Example

```
my_tuple = (10, 20, 30, "AI")
print(my_tuple)
```

Accessing Tuple Elements

```
print(my_tuple[0])   # 10
print(my_tuple[-1])  # "AI"
```

Why Use Tuples?
Data protection: Data cannot be modified.
Faster than lists.
Can be used as dictionary keys (lists can't).

Tuple Operations

Operation	Example	Output
Indexing	my_tuple[1]	20
Slicing	my_tuple[1:3]	(20, 30)
Concatenation	(1,2)+(3,4)	(1,2,3,4)
Repetition	("AI",)*3	("AI","AI","AI")
Membership	"AI" in my_tuple	TRUE

Conditional Blocks using if, else and elif

👉 What is a Conditional Block?
Conditional blocks are used to make decisions in Python programs — they check conditions and execute code accordingly.

Syntax

```
if condition:
    statement1
elif another_condition:
    statement2
else:
    statement3
```

Example 1: if

```
x = 10
if x > 5:
    print("x is greater than 5")
```

✅ Output → x is greater than 5

Example 2: if...else

```
age = 18
if age >= 18:
    print("You are adult")
else:
    print("You are minor")
```

✅ Output → You are adult

Example 3: if...elif...else

```
marks = 75

if marks >= 90:
    print("Grade A")
elif marks >= 75:
    print("Grade B")
elif marks >= 60:
    print("Grade C")
else:
    print("Fail")
```

✅ Output → Grade B

Explanation (in simple words):
if → checks the first condition.
elif → used when there are multiple conditions.
else → runs when no condition is true.
Indentation is mandatory to group statements.

✅ ***Summary (For Exam Quick Revision)***

Topic	**Key Points**
String	Sequence of characters; supports
List	Mutable, ordered, can store mixed
Tuple	Immutable, ordered, faster than
If-Else-Elif	Used for decision making based on

Simple for Loops in Python

👉 What is a for loop?

A for loop is used to repeat a block of code for each item in a sequence (like a list, string, or range).

Syntax

```
for variable in sequence:
    # statements
```

Example

```
for i in range(5):
    print("Hello AI")
```

✅ Output:

```
Hello AI
Hello AI
Hello AI
Hello AI
Hello AI
```

Explanation (simple words):

Loop i runs from 0 to 4 (total 5 times). Each time, it prints “Hello AI”.

For Loop using ranges, string, list and dictionaries

Using range()

range(start, stop, step) → generates a sequence of numbers.

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

✅ Output → 1 2 3 4 5

```
for i in range(0, 10, 2):
    print(i)
```

✅ Output → 0 2 4 6 8

For loop with String

```
for ch in "PYTHON":
    print(ch)
```

✅ Output:

P
Y
T
H
O
N

For loop with List

```
fruits = ["apple", "banana", "mango"]
for f in fruits:
    print(f)
```

✅ Output:

apple
banana
mango

For loop with Dictionary

You can loop through keys, values, or both.

```
student = {"name": "John", "age": 20, "course": "AI"}

# Loop through keys
for key in student:
    print(key)

# Loop through values
for value in student.values():
    print(value)

# Loop through key & value both
for key, value in student.items():
    print(key, ":", value)
```

✅ Output:

```
name : John
age : 20
course : AI
```

Use of While Loops in Python

👉 What is a while loop?

A while loop repeats code as long as a condition is true.

Syntax

```
while condition:
    # statements
```

Example

```
i = 1
while i <= 5:
    print("AI", i)
    i += 1
```

✅ Output:

```
AI 1
AI 2
AI 3
AI 4
AI 5
```

Explanation (in simple words):
The loop runs while i <= 5. After every iteration, i increases by 1.

Infinite Loop Example

```
while True:
    print("Hello")
```

⚠️ This will run forever (use Ctrl + C to stop).

Loop Manipulation using pass, continue, break and else

pass

Used when you want no action in a loop (just a placeholder).

```
for i in range(5):
    pass  # does nothing
print("Loop completed")
```

continue

Skips the current iteration and moves to the next one.

```
for i in range(5):
    if i == 2:
        continue
    print(i)
```

✅ Output → 0 1 3 4
(When i=2, it skips printing.)

break

Stops the loop completely.

```
for i in range(10):
    if i == 5:
        break
    print(i)
```

✅ Output → 0 1 2 3 4

else (with loop)

The else block runs only if the loop finishes normally (without break).

```
for i in range(3):
    print(i)
else:
    print("Loop completed")
```

✅ Output:

```
0
1
2
Loop completed
```

Programming using Python Conditional and Loops Block

Let's see a few simple programs combining conditions and loops 👇

Example 1: Print even numbers from 1 to 10

```
for i in range(1, 11):
    if i % 2 == 0:
        print(i)
```

✅ Output → 2 4 6 8 10

Example 2: Sum of first 5 natural numbers

```
sum = 0
for i in range(1, 6):
    sum += i
print("Sum =", sum)
```

✅ Output → Sum = 15

Example 3: Check if number is positive or negative

```
num = int(input("Enter number: "))
if num > 0:
    print("Positive")
elif num < 0:
    print("Negative")
else:
    print("Zero")
```

Example 4: Print numbers using while loop

```
i = 1
while i <= 5:
    print(i)
    i += 1
```

Example 5: Use of break and continue

```
for i in range(1, 10):
    if i == 3:
        continue  # skips 3
    if i == 7:
        break     # stops at 7
    print(i)
```

✅ Output → 1 2 4 5 6

✅ *Summary (For Exam Quick Revision)*

Topic	Key Points
for loop	Used to iterate over range, list, string, dict
while loop	Runs till condition is true
pass	Do nothing (placeholder)
continue	Skip current iteration
break	Exit loop early
else (with loop)	Executes if loop completes normally
Conditional + Loops	Used together for logic-based programs

Practice Questions

1. What are the steps to install Python and how can we verify its installation?
2. Explain Python variables and write the rules for naming them.
3. What are Python basic operators? Give examples of each type.
4. What is indentation in Python? How are blocks defined using indentation?
5. Explain different numeric data types in Python with examples.
6. What is a string in Python? Write any five string operations or methods.
7. What is the difference between list and tuple in Python?
8. Write a Python program using if, elif, and else to display student grades based on marks.
9. What is the difference between for loop and while loop? Give one example of each.
10. Explain the use of break, continue, pass, and else statements in Python loops with suitable examples.

Check the answer in the Practice Questions section on our website.

Visit Website