

# Advanced Python Programming (4321602) - Winter 2023 Solution

Milav Dabgar

January 24, 2024

## Question 1

### પ્રશ્ન 1(a) [3 ગુણ]

What is Dictionary? Explain with example.

#### જવાબ

Dictionary એ Python માં key-value pairs નું collection છે જે mutable અને ordered છે.  
Dictionary ના ગુણધર્મો:

ગુણધર્મ	વર્ણન
Mutable	Values બદલી શકાય છે
Ordered	Insertion order જળવાઈ રહે છે (Python 3.7+)
Indexed	Keys દ્વારા access થાય છે
No Duplicates	Duplicate keys allow નથી

```
1 # Dictionary ઉદાહરણ
2 student = {
3     "name": "Raj",
4     "age": 20,
5     "course": "IT"
6 }
7 print(student["name"]) # Output: Raj
8
```

- **Key-Value Structure:** દરેક element પાસે key અને value હોય છે
- **Fast Access:** Data access O(1) time complexity માં થાય છે
- **Dynamic Size:** Runtime એ size વધી કે ઘટી શકે છે

#### મેમરી ટ્રીક

Dictionary = Key Value Treasure

### પ્રશ્ન 1(b) [4 ગુણ]

Explain Tuple Built-in functions and methods.

## જવાબ

Tuple immutable હોવાથી તેની પાસે limited built-in methods છે.

**Tuple મેથ્ડુસ:**

મેથડ	વર્ણન	ઉદાહરણ
<b>count()</b>	Element કેટલી વાર આવે છે તે આપે	t.count(5)
<b>index()</b>	Element નો પ્રથમ index આપે	t.index('a')
<b>len()</b>	Tuple ની length આપે	len(t)
<b>max()</b>	Maximum value આપે	max(t)
<b>min()</b>	Minimum value આપે	min(t)

```

1 # Tuple Methods ઉદાહરણ
2 numbers = (1, 2, 3, 2, 4, 2)
3 print(numbers.count(2))    # Output: 3
4 print(numbers.index(3))    # Output: 2
5 print(len(numbers))        # Output: 6
6

```

- Immutable Nature:** Methods tuple ને modify કરતા નથી
- Return Values:** બધી methods નવી values return કરે છે
- Type Conversion:** tuple() function નો ઉપયોગ list ને tuple માં ફેરબવા થાય છે

### મેમરી ટ્રીક

Count Index Length Max Min

## પ્રશ્ન 1(c) [7 ગુણા]

Write a python program to demonstrate set operations.

## જવાબ

Set operations mathematical set theory પર આધારિત છે.

**Set Operations:**

Operation	Symbol	મેથડ	વર્ણન
<b>Union</b>		union()	બંને sets ના elements
<b>Intersection</b>	&	intersection()	Common elements
<b>Difference</b>	-	difference()	પહેલા માંથી બીજા ના બાદ
<b>Symmetric Difference</b>	^	symmetric_difference()	માત્ર unique elements

```

1 # Set Operations પ્રોગ્રામ
2 set1 = {1, 2, 3, 4, 5}
3 set2 = {4, 5, 6, 7, 8}
4
5 print("Set 1:", set1)
6 print("Set 2:", set2)
7
8 # Union Operation
9 union_result = set1 | set2
10 print("Union:", union_result)
11

```

```

12 # Intersection Operation
13 intersection_result = set1 & set2
14 print("Intersection:", intersection_result)
15
16 # Difference Operation
17 difference_result = set1 - set2
18 print("Difference:", difference_result)
19
20 # Symmetric Difference
21 sym_diff_result = set1 ^ set2
22 print("Symmetric Difference:", sym_diff_result)
23
24 # Subset and Superset
25 set3 = {1, 2}
26 print("જું set3 એ set1 નો subset છે?", set3.issubset(set1))
27 print("જું set1 એ set3 નો superset છે?", set1.issuperset(set3))
28

```

- **Mathematical Operations:** Set theory ની operations implement કરે છે
- **Efficient Processing:** Duplicate elements automatically remove થાય છે
- **Boolean Results:** Subset/superset operations boolean return કરે છે

### મેમરી ટ્રીક

Union Intersection Difference Symmetric

## પ્રશ્ન 1(c) OR [7 ગુણા]

Write a python program to demonstrate the dictionaries functions and operations.

### જવાબ

Dictionary operations data manipulation માટે powerful tools પૂરા પાડે છે.

**Dictionary મેથ્ડ્સ:**

મેથ્ડ	વર્ણન	ઉદાહરણ
keys()	બધી keys આપે	dict.keys()
values()	બધી values આપે	dict.values()
items()	key-value pairs આપે	dict.items()
get()	Safe value retrieval	dict.get('key')
update()	Dictionary merge કરે	dict.update()

```

1 # Dictionary Operations પ્રોગ્રામ
2 student_data = {
3     "name": "Amit",
4     "age": 21,
5     "course": "IT",
6     "semester": 2
7 }
8
9 print("Original Dictionary:", student_data)
10
11 # Values access કરવી
12 print("Student Name:", student_data.get("name"))
13 print("Student Age:", student_data["age"])

```

```

14 # નવી key-value pair ઉમેરવી
15 student_data["city"] = "Ahmedabad"
16 print("City ઉમેર્યા પછી:", student_data)
17
18 # Existing value update કરવી
19 student_data.update({"age": 22, "semester": 3})
20 print("Update પછી:", student_data)
21
22 # Dictionary methods
23 print("Keys:", list(student_data.keys()))
24 print("Values:", list(student_data.values()))
25 print("Items:", list(student_data.items()))
26
27 # Elements remove કરવા
28 removed_value = student_data.pop("semester")
29 print("Removed value:", removed_value)
30 print("Final Dictionary:", student_data)
31
32

```

- **Dynamic Operations:** Keys અને values runtime એ add/remove કરી શકાય
- **Safe Access:** get() method KeyError થી બચાવે છે
- **Iteration Support:** keys(), values(), items() methods લૂપ્સ માટે ઉપયોગી છે

મેમરી ટ્રીક

Get Keys Values Items Update Pop

## Question 2

### પ્રશ્ન 2(a) [3 ગુણા]

Distinguish between Tuple and List in Python.

#### જવાબ

##### Tuple vs List તફાવત:

વિશેષતા	Tuple	List
<b>Mutability</b>	Immutable (બદલી ન શકાય)	Mutable (બદલી શકાય)
<b>Syntax</b>	Parentheses ()	Square brackets []
<b>Performance</b>	Faster (જડપી)	Slower (ધીમું)
<b>Memory</b>	ઓછી memory	વધુ memory
<b>Methods</b>	Limited (count, index)	ધારી methods ઉપલબ્ધ
<b>Use Case</b>	Fixed data	Dynamic data

- **Immutable Nature:** Tuple બન્યા પછી બદલી શકતું નથી
- **Performance:** Tuple operations list કરતા જડપી છે
- **Memory Efficient:** Tuple ઓછી memory રોક્કે છે

મેમરી ટ્રીક

Tuple Tight, List Light

## પ્રશ્ન 2(b) [4 ગુણ]

What is the dir() function in python? Explain with example.

### જવાબ

dir() એ એક built-in function છે જે object ના attributes અને methods નું list આપે છે.

#### dir() Function Features:

Feature	વર્ણન
Object Inspection	Object ના attributes બતાવે છે
Method Discovery	ઉપલબ્ધ methods નું list આપે છે
Namespace Exploration	Current namespace ના variables બતાવે છે
Module Analysis	Module ના contents explore કરે છે

```

1 # dir() Function નોંધરણ
2 # String object માટે
3 text = "Hello"
4 string_methods = dir(text)
5 print("String methods:", string_methods[:5])
6
7 # List object માટે
8 my_list = [1, 2, 3]
9 list_methods = dir(my_list)
10 print("List methods:", [m for m in list_methods if not m.startswith('_')][:5])
11
12 # Current namespace માટે
13 print("Current namespace:", dir()[:3])
14
15 # Built-in functions માટે
16 import math
17 print("Math module:", dir(math)[:5])
18

```

- **Interactive Development:** Objects ની capabilities જાણવા માટે ઉપયોગી
- **Debugging Tool:** ઉપલબ્ધ methods શોધવા માટે
- **Learning Aid:** નવી libraries explore કરવા માટે મદદગ્રાપ

### મેમરી ટ્રીક

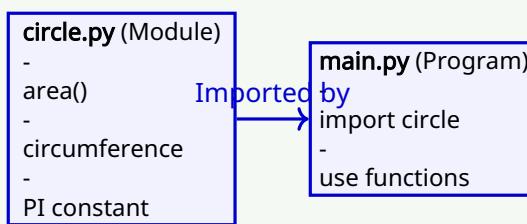
Dir = Directory of Methods

## પ્રશ્ન 2(c) [7 ગુણ]

Write a program to define a module to find the area and circumference of a circle. Import module to another program.

### જવાબ

Module approach code reusability અને organization સુધારે છે.



### File 1: circle.py (Module)

```

1 # circle.py - Circle calculation module
2 import math
3
4 # Constants
5 PI = math.pi
6
7 def area(radius):
8     """Circle ની area ગણો"""
9     if radius < 0:
10        return "Radius negative ન હોઈ શકે"
11    return PI * radius * radius
12
13 def circumference(radius):
14     """Circle ની circumference ગણો"""
15     if radius < 0:
16        return "Radius negative ન હોઈ શકે"
17     return 2 * PI * radius
18
19 def display_info():
20     """Module information display કરો"""
21     print("Circle Module - Version 1.0")
22     print("Functions: area(), circumference()")
23

```

### File 2: main.py (Main Program)

```

1 # main.py - Circle module નો ઉપયોગ કરતો main program
2 import circle
3
4 # User પાસેથી radius લો
5 radius = float(input("Radius નાખો: "))
6
7 # Module functions નો ઉપયોગ કરીને ગણતરી
8 circle_area = circle.area(radius)
9 circle_circumference = circle.circumference(radius)
10
11 # પરિણામ દર્શાવો
12 print(f"Radius {radius} માટે:")
13 print(f"Area: {circle_area:.2f}")
14 print(f"Circumference: {circle_circumference:.2f}")
15
16 # Module info દર્શાવો
17 circle.display_info()
18

```

- Modular Design:** Functions ને separate file માં organize કરે છે
- Reusability:** Module નો ઉપયોગ multiple programs માં થઈ શકે
- Namespace Management:** Functions module prefix દ્વારા access થાય છે

## મેમરી ટ્રીક

Import Calculate Display

## પ્રશ્ન 2(a) OR [3 ગુણ]

Explain Nested Tuple with example.

## જવાબ

Nested Tuple એ બીજા tuples ને તની અંદર સમાવે છે, જે hierarchical structure બનાવે છે.

Nested Tuple Features:

Feature	વર્ણન
Multi-dimensional	2D અથવા 3D data structure
Immutable	દરેક level પર immutable હોય છે
Indexing	Multiple square brackets દ્વારા access થાય
Heterogeneous	અલગ અલગ data types store કરી શકે

```

1 # Nested Tuple ઉદાહરણ
2 student_records = (
3     ("Raj", 20, ("IT", 2)),
4     ("Priya", 19, ("CS", 1)),
5     ("Amit", 21, ("IT", 3))
6 )
7
8 # Nested elements access કરવા
9 print("First student:", student_records[0])
10 print("First student name:", student_records[0][0])
11 print("First student course:", student_records[0][2][0])
12
13 # Nested tuple માં iterate કરવું
14 for student in student_records:
15     name, age, (course, semester) = student
16     print(f"{name} - Age: {age}, Course: {course}, Sem: {semester}")
17

```

- **Data Organization:** Related data ને group કરવા માટે ઉપયોગી
- **Immutable Structure:** Structure એકવાર બન્યા પછી બદલી શકતું નથી
- **Efficient Access:** Fast index-based access

## મેમરી ટ્રીક

Nested = Tuple Inside Tuple

## પ્રશ્ન 2(b) OR [4 ગુણ]

What is PIP? Write the syntax to install and uninstall python packages.

## જવાબ

PIP (Pip Installs Packages) એ Python package installer છે જે PyPI પરથી packages download અને install કરે છે.  
**PIP Commands:**

Command	Syntax	વર્ણન
<b>Install</b>	pip install package	Package install કરે
<b>Uninstall</b>	pip uninstall package	Package remove કરે
<b>List</b>	pip list	Installed packages બતાવે
<b>Show</b>	pip show package	Package info બતાવે
<b>Upgrade</b>	pip install --upgrade pkg	Package update કરે

```

1 # PIP Command ઉદાહરણો (Terminal/Command Prompt માં run કરો)
2
3 # Package install કરવા
4 # pip install requests
5
6 # Specific version install કરવા
7 # pip install Django==3.2.0
8
9 # Package uninstall કરવા
10 # pip uninstall numpy
11
12 # બધા installed packages જોવા
13 # pip list
14
15 # Package information જોવા
16 # pip show matplotlib
17
18 # Package upgrade કરવા
19 # pip install --upgrade pandas
20
21 # Requirements file માંથી install કરવા
22 # pip install -r requirements.txt
23

```

- Package Management:** Third-party libraries ને સરળતાથી manage કરે છે
- Version Control:** Specific versions install કરી શકાય છે
- Dependency Resolution:** જરૂરી dependencies automatically install થાય છે

### મેમરી ટ્રીક

PIP = Package Install Python

## પ્રશ્ન 2(c) OR [7 ગુણ]

Explain different ways of importing package. How are modules and packages connected to each other?

## જવાબ

Python માં અલગ અલગ રીતે import કરવાથી code organization અને namespace management માં મદદ મળે છે.  
**Import Methods:**

Method	Syntax	ઉપયોગ
<b>Basic Import</b>	import module	Full module name જરૂરી
<b>From Import</b>	from module import function	Direct function access
<b>Alias Import</b>	import module as alias	Module માટે short name
<b>Star Import</b>	from module import *	બધા functions import કરે
<b>Package Import</b>	from package import module	Package માંથી import

```

1 # અલગ અલગ Import Ways
2
3 # 1. Basic Import
4 import math
5 result = math.sqrt(16)
6
7 # 2. From Import
8 from math import sqrt, pi
9 result = sqrt(16)
10 area = pi * 5 * 5
11
12 # 3. Alias Import
13 import numpy as np
14 array = np.array([1, 2, 3])
15
16 # 4. Star Import (not recommended)
17 from math import *
18 result = cos(0)
19
20 # 5. Package Import
21 from mypackage import module1
22 from mypackage.subpackage import module3
23

```

#### Module-Package Connection:

- **Modules:** Single .py files જેમાં Python code હોય છે
- **Packages:** Directories જેમાં multiple modules અને \_\_init\_\_.py હોય છે
- **Namespace:** Packages hierarchical namespace structure બનાવે છે
- **\_\_init\_\_.py:** Directory ને package બનાવે છે અને imports control કરે છે

#### મેમરી ટ્રીક

Import From As Star Package

## Question 3

### પ્રશ્ન 3(a) [3 ગુણા]

Describe Runtime Error and Syntax Error. Explain with example.

#### જવાબ

Error Types તફાવત:

Error Type	ક્યારે થાય	Detection	ઉદાહરણ
Syntax Error	Code parsing વખતે	Execution પહેલાં	Colon ભૂલી જવું
Runtime Error	Execution દરમયાન	Run થતી વખતે	Zero division
Logic Error	હંમેશા	Execution પછી	ખોટું logic

```

1 # Syntax Error ઉદાહરણ
2 # print("Hello World" # Closing parenthesis નથી
3 # SyntaxError: unexpected EOF while parsing
4
5 # Runtime Error ઉદાહરણો
6 try:
7     # ZeroDivisionError
8     result = 10 / 0
9 except ZeroDivisionError:
10    print("Zero વડે ભાગી શકાય નહિં")
11
12 try:
13     # FileNotFoundError
14     file = open("nonexistent.txt", "r")
15 except FileNotFoundError:
16    print("File મળી નથી")
17

```

- **Syntax Errors:** Code run થાય તે પહેલાં પકડાઈ જાય છે
- **Runtime Errors:** Program execution દરમયાન આવે છે
- **Prevention:** Exception handling થી runtime errors handle કરી શકાય

### મેમરી ટ્રીક

Syntax Before, Runtime During

## પ્રશ્ન 3(b) [4 ગુણ]

What is Exception handling in Python? Explain with example.

### જવાબ

Exception handling એ runtime errors ને gracefully handle કરવા અને program crash થતો અટકાવવા માટે વપરાય છે.

#### Exception Handling Keywords:

Keyword	હેતુ	વર્ણન
try	જેમાં exception આવી શકે	Risk code block
except	Exception handle કરવા	Error handling block
finally	હંમેશા run થાય	Cleanup code
else	જો exception ન આવે	Success code block
raise	Manual exception raise કરવા	Custom error throwing

```

1 # Exception Handling ઉદાહરણ
2 def safe_division(a, b):
3     try:
4         # Code જેમાં exception આવી શકે
5         result = a / b
6         print(f"માંગાકાર સફળ: {result}")

```

```

7     except ZeroDivisionError:
8         # Specific exception handle કરો
9         print("Error: Zero વડે ભાગી શકાય નહોં")
10        result = None
11
12
13    except TypeError:
14        # Type errors handle કરો
15        print("Error: Invalid data types")
16        result = None
17
18    else:
19        # Exception ન આવે તો જ �run થાય
20        print("Division સફળતાપૂર્વક પૂરણ થયું")
21
22    finally:
23        # હુમેશા run થાય
24        print("Division operation સમાપ્ત")
25
26    return result
27
28 # Function test કરો
29 safe_division(10, 2) # Normal case
30 safe_division(10, 0) # Zero division
31 safe_division(10, "a") # Type error
32

```

- **Error Prevention:** Program ને crash થતો અટકાવે છે
- **Graceful Handling:** User-friendly error messages આપે છે
- **Resource Management:** Finally block માં cleanup operations થાય છે

### મેમરી ટ્રીક

Try Except Finally Else Raise

## પ્રશ્ન 3(c) [7 ગુણ]

Create a function for division of two numbers, if the value of any argument is non-integer then raise the error or if second argument is 0 then raise the error.

### જવાબ

Custom exception handling function validation અને error control માટે મહત્વપૂર્ણ છે.

```

1 def safe_integer_division(num1, num2):
2     """
3     એ સંખ્યાઓનો ભાગાકાર validation સાથે
4     જો arguments integer ન હોય તો TypeError raise કરો
5     જો બીજી argument 0 હોય તો ZeroDivisionError raise કરો
6     """
7
8     # બંને arguments integers છે કે કેમ તે તપાસો
9     if not isinstance(num1, int):
10        raise TypeError(f"પહેલી argument integer હોવી જોઈએ, {type(num1).__name__} મળીએ")
11
12     if not isinstance(num2, int):
13        raise TypeError(f"બીજી argument integer હોવી જોઈએ, {type(num2).__name__} મળીએ")
14

```

```

15 # Zero division તપાસો
16 if num2 == 0:
17     raise ZeroDivisionError("Zero વડે ભાગી શકાય નહીં")
18
19 # ભાગાકાર કરો
20 result = num1 / num2
21 return result
22
23 # અલગ અલગ cases સાથે function test કરો
24 def test_division():
25     test_cases = [
26         (10, 2),    # Valid case
27         (15, 3),    # Valid case
28         (10, 0),    # Zero division error
29         (10.5, 2),  # Non-integer first argument
30         (10, 2.5),  # Non-integer second argument
31         ("10", 2),  # String argument
32     ]
33
34     for num1, num2 in test_cases:
35         try:
36             result = safe_integer_division(num1, num2)
37             print(f"{num1} / {num2} = {result}")
38
39         except TypeError as e:
40             print(f"Type Error: {e}")
41
42         except ZeroDivisionError as e:
43             print(f"Zero Division Error: {e}")
44
45         except Exception as e:
46             print(f"Unexpected Error: {e}")
47
48     print("-" * 40)
49
50 # Tests run કરો
51 test_division()
52

```

- **Input Validation:** Arguments ના type અને value તપાસે છે
- **Custom Errors:** Specific exceptions raise કરે છે
- **Error Messages:** સ્પષ્ટ અને વર્ણનાત્મક error messages આપે છે

### મેમરી ટ્રીક

Validate Type, Check Zero, Divide Safe

## પ્રશ્ન 3(a) OR [3 ગુણ]

Describe any five built-in exceptions in Python.

### જવાબ

**Built-in Exceptions:**

Exception	કારણ	ઉદાહરણ
ValueError	Operation માટે invalid value અપા ત્યારે	int("abc")
TypeError	ખોટો data type હોય ત્યારે	"5" + 5
IndexError	Index range ની બહાર હોય ત્યારે	list[10]
KeyError	Dictionary key ન મળે ત્યારે	dict["missing"]
FileNotFoundException	File અસ્તિત્વમાં ન હોય ત્યારે	open("missing.txt")

- Automatic Detection: Python આ exceptions automatically raise કરે છે
- Specific Handling: દરેક exception નો specific હેતુ છે
- Inheritance: બધા exceptions BaseException class માંથી inherit થાય છે

### મેમરી ટ્રીક

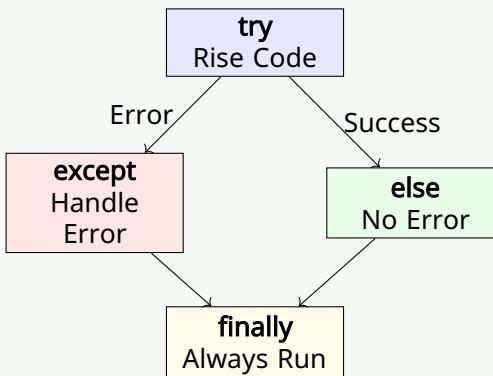
Value Type Index Key File

## પ્રશ્ન 3(b) OR [4 ગુણ]

Explain try...except...else...finally block with example.

### જવાબ

Exception Block Structure:



```

1 # Complete Exception Block ઉદાહરણ
2 def file_operation(filename):
3     try:
4         # File open કરવાનો પ્રયત્ન
5         f = open(filename, "r")
6         print("File સફળતાપૂર્વક open થઈ")
7
8     except FileNotFoundError:
9         # જો file ન મળે
10        print("Error: File મળી નથી")
11
12 else:
13     # જો કોઈ error ન આવે તો run થાય
14     print("File content વાચી રહ્યા છીએ...")
15     print(f.read())
16     f.close()
17
18 finally:
19     # હંમેશા run થાય
  
```

```

20 print("File operation process પૂરણ")
21
22 # Existing અને missing files સાથે test કરો
23 file_operation("existing.txt")
24 file_operation("missing.txt")
25

```

- **Else Block:** માત્ર ત્યારે જે execute થાય જો try block માં exception ન આવે
- **Finally Block:** Error આવે કે ન આવે, હમેશા execute થાય (cleanup માટે)
- **Complete Flow:** Execution ની બધી શક્યતાઓ cover કરે છે

### મેમરી ટ્રીક

Try Exception Else Finally

## પ્રશ્ન 3(c) OR [7 ગુણ]

Write a user defined exception that could be raised when the text entered by a user consists of less than 10 characters.

### જવાબ

User-defined exceptions custom validation logic implement કરવા દે છે.

```

1 class ShortTextError(Exception):
2     """Text validation માટે custom exception"""
3     def __init__(self, length):
4         self.length = length
5         self.message = f"Text ખૂબ નાનું છે ({length} chars). ઓછામાં ઓછા 10 જોઈએ."
6         super().__init__(self.message)
7
8     def validate_text():
9         while True:
10             try:
11                 # User input લો
12                 text = input("Text દાખલ કરો (min 10 chars): ")
13
14                 # Length તપાસો
15                 if len(text) < 10:
16                     raise ShortTextError(len(text))
17
18                 print(f"Valid text સ્વીકારણું: {text}")
19                 break
20
21             except ShortTextError as e:
22                 print(f"Error: {e}")
23                 print("ફરી પ્રયાસ કરો.\n")
24
25             except Exception as e:
26                 print(f"Unexpected error: {e}")
27
28     # Validation run કરો
29     print("--- Text Validation Program ---")
30     # validate_text() # Run કરવા માટે uncomment કરો
31

```

- **Inheritance:** Custom exception class Exception માંથી inherit થાય છે
- **Raising:** raise keyword exception trigger કરવા વપરાય છે

- **Usage:** Domain-specific rules implement કરવામાં મદદ કરે છે

**મેમરી ટ્રીક**

Class Inherit Raise Catch

## Question 4

### પ્રશ્ન 4(a) [3 ગુણ]

Write five points on difference between Text File and Binary File.

**જવાબ**

Text vs Binary File:

Feature	Text File	Binary File
<b>Content</b>	માનવ વાંચી શકે તેવા અક્ષરો	મશીન વાંચી શકે તેવો data (0s/1s)
<b>Encoding</b>	Encoding વપરાય (ASCII/UTF-8)	Encoding વપરતું નથી
<b>Extensions</b>	.txt, .py, .csv	.bin, .jpg, .exe
<b>EOL</b>	Newline translation handle કરે	Translation કરતું નથી
<b>Processing</b>	ધીમું processing	ડાપી processing

- **Readability:** Text files સાદા editors માં ખોલી શકાય છે
- **Portability:** Binary files bytes નો continuous stream છે
- **Usage:** Documents માટે Text, Images/Videos માટે Binary

**મેમરી ટ્રીક**

Text Human, Binary Machine

### પ્રશ્ન 4(b) [4 ગુણ]

Write a program to read the data from a file and separate the uppercase character and lowercase character into two separate files.

**જવાબ**

File processing માં વાંચવું, વિશ્લેષણ કરવું અને લખવું શામેલ છે.

```

1 def separate_case_chars(source_file):
2     try:
3         # Characters store કરવા માટે strings
4         upper_chars = ""
5         lower_chars = ""
6
7         # Source file વાંચો
8         with open(source_file, "r") as f:
9             content = f.read()
10
11         # દરેક character process કરો
12         for char in content:

```

```

13     if char.isupper():
14         upper_chars += char
15     elif char.islower():
16         lower_chars += char
17
18     # Uppercase characters લખો
19     with open("uppercase.txt", "w") as f:
20         f.write(upper_chars)
21
22     # Lowercase characters લખો
23     with open("lowercase.txt", "w") as f:
24         f.write(lower_chars)
25
26     print("Characters સફળતાપૂર્વક અલગ કરયા!")
27
28 except FileNotFoundError:
29     print("Source file મળી નથી")
30
31 # પહેલા sample file બનાવો
32 with open("input.txt", "w") as f:
33     f.write("Hello World! Python Programming.")
34
35 # Separation run કરો
36 separate_case_chars("input.txt")
37

```

- **String Methods:** isupper() અને islower() case તપાસે છે
- **File Handling:** with statement safe file operations માટે વપરાય છે
- **Data Separation:** Content અલગ અલગ streams માં filter થાય છે

### મેમરી ટ્રીક

Read Loop Check Write

## પ્રક્રિયા 4(c) [7 ગુણ]

Describe dump() and load() method. Explain with example.

### જવાબ

dump() અને load() એ **pickle** module ના ભાગ છે જે object serialization માટે વપરાય છે.  
**Pickle મેથ્ડ્સ:**

મેથ્ડ	હેતુ	Mode
dump(obj, file)	Object ને file માં serialize કરે	Write Binary ('wb')
load(file)	File માંથી object deserialize કરે	Read Binary ('rb')

```

1 import pickle
2
3 # Serialize કરવા માટે data
4 student = {
5     "name": "Rahul",
6     "roll": 101,
7     "marks": [85, 90, 88]
8 }
9

```

```

10 # DUMP ઉદાહરણ
11 try:
12     with open("data.pkl", "wb") as f:
13         pickle.dump(student, f)
14         print("Data સફળતાપૂર્વક dump થયો!")
15 except Exception as e:
16     print(f"Error: {e}")
17
18 # LOAD ઉદાહરણ
19 try:
20     with open("data.pkl", "rb") as f:
21         loaded_data = pickle.load(f)
22         print("Data સફળતાપૂર્વક load થયો:")
23         print(loaded_data)
24 except Exception as e:
25     print(f"Error: {e}")
26

```

- **Serialization:** Object ને byte stream માં ફેરવવું (Pickling)
- **Deserialization:** Byte stream ને પાછું object માં ફેરવવું (Unpickling)
- **Persistence:** Objects ની state ને disk પર save કરવા દે છે

### મેમરી ટ્રીક

Dump Store, Load Restore

## પ્રશ્ન 4(a) OR [3 ગુણ]

List different types of file modes provided by python for file operations and explain their uses.

### જવાબ

#### Python File Modes:

Mode	નામ	વર્ણન
'r'	Read	Default mode. વાંચવા માટે ખોલે.
'w'	Write	લખવા માટે ખોલે. File overwrite કરે.
'a'	Append	લખવા માટે ખોલે. અંતમાં ઉમેરે.
'x'	Create	નવી file બનાવે. જે હોય તો fail થાય.
'b'	Binary	Binary mode (દા.ત., 'rb', 'wb').
'+'	Update	Read અને Write (દા.ત., 'r+', 'w+').

- **Safety:** 'x' આકર્ષિક overwriting અટકાવે છે
- **Binary:** 'b' નો ઉપયોગ non-text files માટે થવો જોઈએ
- **Combination:** '+' ને અન્ય modes સાથે જોડી શકાય છે

### મેમરી ટ્રીક

Read Write Append Create

## પ્રશ્ન 4(b) OR [4 ગુણ]

Describe readline() and writeline() functions of the file.

## જવાબ

નોંધ: Python માં writelines() છે પરંપરા writeline() નથી.

### Line Operations:

Function	હેતુ	ઉદાહરણ
readline()	એક line વાંચે છે	line = f.readline()
readlines()	બધી lines list માં વાંચે છે	lines = f.readlines()
writelines()	Strings નું list લખે છે	f.writelines(list)

```

1 # Lines લખવી
2 lines = ["First Line\n", "Second Line\n"]
3 with open("demo.txt", "w") as f:
4     f.writelines(lines)
5
6 # Lines વાંચવી
7 with open("demo.txt", "r") as f:
8     line1 = f.readline()
9     print(f"Read 1: {line1.strip()}")
10
11 line2 = f.readline()
12 print(f"Read 2: {line2.strip()}")
13

```

- **Sequential:** readline pointer ને line by line ખસેડે છે
- **List Support:** writelines iterable (list/tuple) accept કરે છે
- **Memory:** readline મોટી files માટે efficient છે

### મેમરી ટ્રીક

Read One, Write Many

## પ્રશ્ન 4(c) OR [7 ગુણ]

Write a python program to demonstrate seek() and tell() methods.

## જવાબ

seek() file pointer ખસેડે છે, tell() current position આપે છે.

```

1 # Seek અને Tell Demonstration
2 filename = "seek_demo.txt"
3
4 # File બાબો
5 with open(filename, "w") as f:
6     f.write("Hello Python World")
7
8 # Read operations
9 with open(filename, "r") as f:
10    # શરૂઆતની position
11    print(f"Start Position: {f.tell()}") # 0
12
13    # પહેલા 5 chars વાંચો
14    print(f"Read: {f.read(5)}")      # Hello
15    print(f"Current Position: {f.tell()}") # 5
16

```

```

17 # શરૂઆતમાં seek કરો
18 f.seek(0)
19 print(f"After seek(0): {f.tell()}" ) # 0
20
21 # Specific position પર seek કરો
22 f.seek(6)
23 print(f"After seek(6): {f.read(6)}") # Python
24
25 # અંતે જાઓ (binary mode માં અથવા specific syntax સા�ે)
26 # f.seek(0, 2) અંતે લઈ જાય
27

```

Method	Syntax
tell()	pos = f.tell()
seek()	f.seek(offset, whence)

- Navigation:** Files માં random access ની સુવિધા આપે છે
- Whence:** 0=Start, 1=Current, 2=End
- Binary:** Relative seeking binary mode માં શ્રેષ્ઠ કામ કરે છે

### મેમરી ટ્રીક

Tell Position, Seek Location

## Question 5

### પ્રશ્ન 5(a) [3 ગુણ]

Draw the shape of circle and rectangle using turtle and fill with red color.

#### જવાબ

```

1 import turtle
2
3 t = turtle.Turtle()
4
5 # Fill color red set કરો
6 t.fillcolor("red")
7
8 # Circle દોરો
9 t.begin_fill()
10 t.circle(50) # 50 radius વાળું circle
11 t.end_fill()
12
13 # નવી position પર જાઓ
14 t.penup()
15 t.goto(100, 0)
16 t.pendown()
17
18 # Rectangle દોરો
19 t.begin_fill()
20 for _ in range(2):
21     t.forward(100) # લંબાઈ
22     t.right(90)

```

```

23     t.forward(50) # પહોળાઈ
24     t.right(90)
25     t.end_fill()
26
27     turtle.done()
28

```

- **begin\_fill()**: Shape ભરવાનું શરૂ કરે છે
- **end\_fill()**: Shape ભરવાનું પૂર્ણ કરે છે
- **Geometry**: Circle અને loop-આધારિત rectangle drawing

### મેમરી ટ્રીક

Begin Fill End

## પ્રશ્ન 5(b) [4 ગુણ]

Explain various inbuilt methods for changing the direction of turtle.

### જવાબ

#### Turtle Direction મેથ્ડ્સ:

મેથ્ડ	વર્ણન	ઉદાહરણ
<b>right(angle)</b>	જમણે વળો (angle)	t.right(90)
<b>left(angle)</b>	ડાબે વળો (angle)	t.left(45)
<b>setheading(angle)</b>	Absolute angle set કરો	t.setheading(0)
<b>towards(x,y)</b>	Coordinates તરફ point કરો	t.towards(0,0)

```

1 # Direction ઉદાહરણો
2 import turtle
3 t = turtle.Turtle()
4
5 t.forward(100)
6 t.right(90) # 90 degrees જમણે
7 t.forward(50)
8 t.left(45) # 45 degrees ડાબે
9 t.setheading(180) # પશ્ચયમિ તરફ (180 degrees)
10

```

- **Relative**: right/left વર્તમાન direction થી સપેક્ષ છે
- **Absolute**: setheading 0-360 degree system વાપરે છે
- **Navigation**: જાટિલ shapes દોરવા માટે જરૂરી

### મેમરી ટ્રીક

Right Left Heading Towards

## પ્રશ્ન 5(c) [7 ગુણ]

Write a python program to draw rainbow using turtle.

## જવાબ

Rainbow drawing માં VIBGYOR colors સાથે concentric semi-circles વપરાય છે.

```

1 import turtle
2
3 def draw_rainbow():
4     # Setup screen
5     wn = turtle.Screen()
6     wn.bgcolor("skyblue")
7
8     # Setup turtle
9     t = turtle.Turtle()
10    t.speed(5)
11    t.pensize(10)
12
13    # Rainbow colors (VIBGYOR reversed)
14    colors = ['violet', 'indigo', 'blue', 'green',
15              'yellow', 'orange', 'red']
16
17    # શરૂઆતના position parameters
18    radius = 180
19
20    # Arcs દોરો
21    for color in colors:
22        t.penup()
23        t.goto(0, -50)  # Center bottom
24        t.setheading(90)  # ઉપર તરફ
25        t.right(90)  # જમણી તરફ (0 deg)
26        t.forward(radius)  # Arc ની શરૂઆતમાં જાઓ
27        t.left(90)  # ફરી ઉપર તરફ face કરો
28        t.pendown()
29
30        t.pencolor(color)
31        t.circle(radius, 180) # Semi-circle દોરો
32
33        radius -= 20  # આગલા color માટે radius ઘટાડો
34
35    t.hideturtle()
36    turtle.done()
37
38 draw_rainbow()
39

```

- **Looping:** Colors ના list માં iterate કરે છે
- **Concentric:** દરેક અંદરના arc માટે radius ઘટે છે
- **Arc:** circle(radius, 180) અધ્ય વર્તૂળ દોરે છે

### મેમરી ટ્રીક

Colors Loop Radius Circle