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

```
# Dictionary ઉદાહરણ
student = {
    "name": "Raj",
    "age": 20,
    "course": "IT"
}
print(student["name"])  # Output: Raj

```

- **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 મેથડ્સ:

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

```
# Tuple Methods ઉદાહરણ
numbers = (1, 2, 3, 2, 4, 2)
print(numbers.count(2))    # Output: 3
print(numbers.index(3))    # Output: 2
print(len(numbers))        # Output: 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 ફેરવવા થાય છે

મેમરી ટ્રીક

Count Index Length Max Min

પ્રશ્ન 1(c) [7 ગુણ]

Write a python program to demonstrate set operations.

જવાબ

Set operations mathematical set theory પર આધારિત છે.
Set Operations:

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

```
# Set Operations પ્રોગ્રામ
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}

print("Set 1:", set1)
print("Set 2:", set2)

# Union Operation
union_result = set1 | set2
print("Union:", union_result)

```

```python
# Intersection Operation
intersection_result = set1 & set2
print("Intersection:", intersection_result)

# Difference Operation
difference_result = set1 - set2
print("Difference:", difference_result)

# Symmetric Difference
sym_diff_result = set1 ^ set2
print("Symmetric Difference:", sym_diff_result)

# Subset and Superset
set3 = {1, 2}
print("શું set3 એ set1 નો subset છે?", set3.issubset(set1))
print("શું set1 એ set3 નો superset છે?", set1.issuperset(set3))

```

- **Mathematical Operations**: Set theory ના 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 ગુણ]

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

જવાબ

Dictionary operations data manipulation માટે powerful tools પૂરા પાડે છે.
Dictionary મેથડ્સ:

મેથડ	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()

```python
# Dictionary Operations પ્રોગ્રામ
student_data = {
    "name": "Amit",
    "age": 21,
    "course": "IT",
    "semester": 2
}

print("Original Dictionary:", student_data)

# Values access કરવી
print("Student Name:", student_data.get("name"))
print("Student Age:", student_data["age"])
```

```python

# નવી key-value pair ઉમેરવી
student_data["city"] = "Ahmedabad"
print("City ઉમેર્યા પછી:", student_data)

# Existing value update કરવી
student_data.update({"age": 22, "semester": 3})
print("Update પછી:", student_data)

# Dictionary methods
print("Keys:", list(student_data.keys()))
print("Values:", list(student_data.values()))
print("Items:", list(student_data.items()))

# Elements remove કરવા
removed_value = student_data.pop("semester")
print("Removed value:", removed_value)
print("Final Dictionary:", student_data)

```

- **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 ગુણ]

Distinguish between Tuple and List in Python.

જવાબ

Tuple vs List તફાવત:

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

- **Immutable Nature**: Tuple બન્યા પછી બદલી શકાતું નથી
- **Performance**: Tuple operations list કરતા ઝડપી છે
- **Memory Efficient**: Tuple ઓછી memory રોકે 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

```
# dir() Function ઉદાહરણ
# String object માટે
text = "Hello"
string_methods = dir(text)
print("String methods:", string_methods[:5])

# List object માટે
my_list = [1, 2, 3]
list_methods = dir(my_list)
print("List methods:", [m for m in list_methods if not m.startswith('_')][:5])

# Current namespace માટે
print("Current namespace:", dir()[:3])

# Built-in functions માટે
import math
print("Math module:", dir(math)[:5])

```

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

મેમરી ટ્રીક

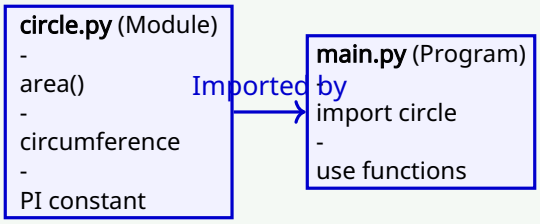
Dir = Directory of Methods

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)

```python
# circle.py - Circle calculation module
import math

# Constants
PI = math.pi

def area(radius):
    """Circle નો area ગણો"""
    if radius < 0:
        return "Radius negative ન હોઈ શકે"
    return PI * radius * radius

def circumference(radius):
    """Circle નો circumference ગણો"""
    if radius < 0:
        return "Radius negative ન હોઈ શકે"
    return 2 * PI * radius

def display_info():
    """Module information display કરો"""
    print("Circle Module - Version 1.0")
    print("Functions: area(), circumference()")

```

File 2: main.py (Main Program)

```python
# main.py - Circle module નો ઉપયોગ કરતો main program
import circle

# User પાસેથી radius લો
radius = float(input("Radius નાખો: "))

# Module functions નો ઉપયોગ કરીને ગણતરી
circle_area = circle.area(radius)
circle_circumference = circle.circumference(radius)

# પરિણામ દર્શાવો
print(f"Radius {radius} માટે:")
print(f"Area: {circle_area:.2f}")
print(f"Circumference: {circle_circumference:.2f}")

# Module info દર્શાવો
circle.display_info()

```

- **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

```python
# Nested Tuple ઉદાહરણ
student_records = (
    ("Raj", 20, ("IT", 2)),
    ("Priya", 19, ("CS", 1)),
    ("Amit", 21, ("IT", 3))
)

# Nested elements access કરવા
print("First student:", student_records[0])
print("First student name:", student_records[0][0])
print("First student course:", student_records[0][2][0])

# Nested tuple માં iterate કરવું
for student in student_records:
    name, age, (course, semester) = student
    print(f"{name} - Age: {age}, Course: {course}, Sem: {semester}")

```

- **Data Organization**: Related data ને group કરવા માટે ઉપયોગી
- **Immutable Structure**: Structure એકવાર બન્યા પછી બદલી શકાતું 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
Install	pip install package	Package install કરે
Uninstall	pip uninstall package	Package remove કરે
List	pip list	Installed packages બતાવે
Show	pip show package	Package info બતાવે
Upgrade	pip install --upgrade pkg	Package update કરે

```
# PIP Command ઉદાહરણો (Terminal/Command Prompt માં run કરો)

# Package install કરવા
# pip install requests

# Specific version install કરવા
# pip install Django==3.2.0

# Package uninstall કરવા
# pip uninstall numpy

# બધા installed packages જોવા
# pip list

# Package information જોવા
# pip show matplotlib

# Package upgrade કરવા
# pip install --upgrade pandas

# Requirements file માંથી install કરવા
# pip install -r requirements.txt

```

- **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
Basic Import	import module	Full module name જરૂરી
From Import	from module import function	Direct function access
Alias Import	import module as alias	Module માટે short name
Star Import	from module import *	બધા functions import કરે
Package Import	from package import module	Package માંથી import

```python
# અલગ અલગ Import Ways

# 1. Basic Import
import math
result = math.sqrt(16)

# 2. From Import
from math import sqrt, pi
result = sqrt(16)
area = pi * 5 * 5

# 3. Alias Import
import numpy as np
array = np.array([1, 2, 3])

# 4. Star Import (not recommended)
from math import *
result = cos(0)

# 5. Package Import
from mypackage import module1
from mypackage.subpackage import module3

```

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 ગુણ]

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 દરમ્યાન	Run થતી વખતે	Zero division
Logic Error	હંમેશા	Execution પછી	ખોટું logic

```
# Syntax Error ઉદાહરણ
# print("Hello World"  # Closing parenthesis નથી
# SyntaxError: unexpected EOF while parsing

# Runtime Error ઉદાહરણો
try:
    # ZeroDivisionError
    result = 10 / 0
except ZeroDivisionError:
    print("Zero વડે ભાગી શકાય નહિ")

try:
    # FileNotFoundError
    file = open("nonexistent.txt", "r")
except FileNotFoundError:
    print("File મળી નથી")

```

- **Syntax Errors**: Code run થાય તે પહેલાં પકડાઈ જાય છે
- **Runtime Errors**: Program execution દરમ્યાન 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

```
# Exception Handling ઉદાહરણ
def safe_division(a, b):
    try:
        # Code જેમાં exception આવી શકે
        result = a / b
        print(f"ભાગાકાર સફળ: {result}")
```

```
    except ZeroDivisionError:
        # Specific exception handle કરો
        print("Error: Zero વડે ભાગી શકાય નહિ")
        result = None

    except TypeError:
        # Type errors handle કરો
        print("Error: Invalid data types")
        result = None

    else:
        # Exception ન આવે તો જ run થાય
        print("Division સફળતાપૂર્વક પૂર્ણ થયું")

    finally:
        # હંમેશા run થાય
        print("Division operation સમાપ્ત")

    return result

# Function test કરો
safe_division(10, 2)   # Normal case
safe_division(10, 0)   # Zero division
safe_division(10, "a") # Type error
```

- **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.

```
def safe_integer_division(num1, num2):
    """
    બે સંખ્યાઓનો ભાગાકાર validation સાથે
    જો arguments integer ન હોય તો TypeError raise કરો
    જો બીજી argument 0 હોય તો ZeroDivisionError raise કરો
    """

    # બંને arguments integers છે કે કેમ તે તપાસો
    if not isinstance(num1, int):
        raise TypeError(f"પહેલી argument integer હોવી જોઈએ, {type(num1).__name__} મળી")

    if not isinstance(num2, int):
        raise TypeError(f"બીજી argument integer હોવી જોઈએ, {type(num2).__name__} મળી")

```

```
    # Zero division તપાસો
    if num2 == 0:
        raise ZeroDivisionError("Zero વડે ભાગી શકાય નહિ")

    # ભાગાકાર કરો
    result = num1 / num2
    return result

# અલગ અલગ cases સાથે function test કરો
def test_division():
    test_cases = [
        (10, 2),     # Valid case
        (15, 3),     # Valid case
        (10, 0),     # Zero division error
        (10.5, 2),   # Non-integer first argument
        (10, 2.5),   # Non-integer second argument
        ("10", 2),   # String argument
    ]

    for num1, num2 in test_cases:
        try:
            result = safe_integer_division(num1, num2)
            print(f"{num1} / {num2} = {result}")

        except TypeError as e:
            print(f"Type Error: {e}")

        except ZeroDivisionError as e:
            print(f"Zero Division Error: {e}")

        except Exception as e:
            print(f"Unexpected Error: {e}")

        print("-" * 40)

# Tests run કરો
test_division()
```

- **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"]
FileNotFoundError	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

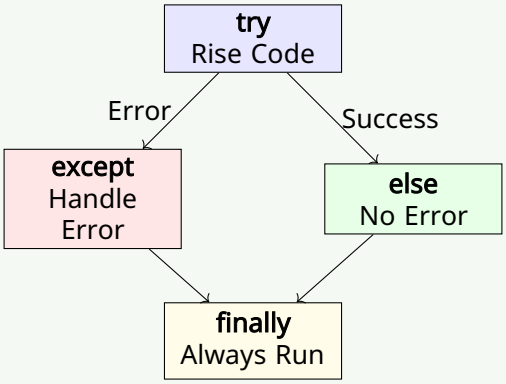
Value Type Index Key File

પ્રશ્ન 3(b) OR [4 ગુણ]

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

જવાબ

Exception Block Structure:




```
# Complete Exception Block ઉદાહરણ
def file_operation(filename):
    try:
        # File open કરવાનો પ્રયત્ન
        f = open(filename, "r")
        print("File સફળતાપૂર્વક open થઈ")

    except FileNotFoundError:
        # જો file ન મળે
        print("Error: File મળી નથી")

    else:
        # જો કોઈ error ન આવે તો run થાય
        print("File content વાંચી રહ્યા છીએ...")
        print(f.read())
        f.close()

    finally:
        # હંમેશા run થાય
```

```
    print("File operation process પૂર્ણ")

# Existing અને missing files સાથે test કરો
file_operation("existing.txt")
file_operation("missing.txt")

```

- **Else Block**: માત્ર ત્યારે જ execute થાય જો try block માં exception ન આવે
- **Finally Block**: Error આવે કે ન આવે, હંમેશા 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.

```
class ShortTextError(Exception):
    """Text validation માટે custom exception"""
    def __init__(self, length):
        self.length = length
        self.message = f"Text ખૂબ નાનું છે ({length} chars). ઓછામાં ઓછા 10 જોઈએ."
        super().__init__(self.message)

def validate_text():
    while True:
        try:
            # User input લો
            text = input("Text દાખલ કરો (min 10 chars): ")

            # Length તપાસો
            if len(text) < 10:
                raise ShortTextError(len(text))

            print(f"Valid text સ્વીકારાયું: {text}")
            break

        except ShortTextError as e:
            print(f"Error: {e}")
            print("ફરી પ્રયાસ કરો.\n")

        except Exception as e:
            print(f"Unexpected error: {e}")

# Validation run કરો
print("--- Text Validation Program ---")
# validate_text() # Run કરવા માટે uncomment કરો

```

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

```
def separate_case_chars(source_file):
    try:
        # Characters store કરવા માટે strings
        upper_chars = ""
        lower_chars = ""

        # Source file વાંચો
        with open(source_file, "r") as f:
            content = f.read()

            # દરેક character process કરો
            for char in content:
```

```
            if char.isupper():
                upper_chars += char
            elif char.islower():
                lower_chars += char

        # Uppercase characters લખો
        with open("uppercase.txt", "w") as f:
            f.write(upper_chars)

        # Lowercase characters લખો
        with open("lowercase.txt", "w") as f:
            f.write(lower_chars)

        print("Characters સફળતાપૂર્વક અલગ કર્યા!")

    except FileNotFoundError:
        print("Source file મળી નથી")

# પહેલા sample file બનાવો
with open("input.txt", "w") as f:
    f.write("Hello World! Python Programming.")

# Separation run કરો
separate_case_chars("input.txt")

```

- **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

પ્રશ્ન 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 મેથડ્સ:

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

```
import pickle

# Serialize કરવા માટે data
student = {
    "name": "Rahul",
    "roll": 101,
    "marks": [85, 90, 88]
}

```

```python
# DUMP ઉદાહરણ
try:
    with open("data.pkl", "wb") as f:
        pickle.dump(student, f)
    print("Data સફળતાપૂર્વક dump થયો")
except Exception as e:
    print(f"Error: {e}")

# LOAD ઉદાહરણ
try:
    with open("data.pkl", "rb") as f:
        loaded_data = pickle.load(f)
    print("Data સફળતાપૂર્વક load થયો:")
    print(loaded_data)
except Exception as e:
    print(f"Error: {e}")

```

- **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' આકસ્મિક 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 પણ 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)

```
# Lines લખવી
lines = ["First Line\n", "Second Line\n"]
with open("demo.txt", "w") as f:
    f.writelines(lines)

# Lines વાંચવી
with open("demo.txt", "r") as f:
    line1 = f.readline()
    print(f"Read 1: {line1.strip()}")

    line2 = f.readline()
    print(f"Read 2: {line2.strip()}")

```

- **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.

```
# Seek અને Tell Demonstration
filename = "seek_demo.txt"

# File બનાવો
with open(filename, "w") as f:
    f.write("Hello Python World")

# Read operations
with open(filename, "r") as f:
    # શરૂઆતની position
    print(f"Start Position: {f.tell()}")  # 0

    # પહેલા 5 chars વાંચો
    print(f"Read: {f.read(5)}")        # Hello
    print(f"Current Position: {f.tell()}") # 5

```

```
    # શરૂઆતમાં seek કરો
    f.seek(0)
    print(f"After seek(0): {f.tell()}")  # 0

    # Specific position પર seek કરો
    f.seek(6)
    print(f"After seek(6): {f.read(6)}")  # Python

    # અંતે જાઓ (binary mode માં અથવા specific syntax સાથે)
    # f.seek(0, 2) અંતે લઈ જાય

```

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

```
import turtle

t = turtle.Turtle()

# Fill color red set કરો
t.fillcolor("red")

# Circle દોરો
t.begin_fill()
t.circle(50)     # 50 radius વાળું circle
t.end_fill()

# નવી position પર જાઓ
t.penup()
t.goto(100, 0)
t.pendown()

# Rectangle દોરો
t.begin_fill()
for _ in range(2):
    t.forward(100) # લંબાઈ
    t.right(90)
```

```
    t.forward(50)  # પહોળાઈ
    t.right(90)
t.end_fill()

turtle.done()

```

- **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 મેથડ્સ:

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

```
# Direction ઉદાહરણો
import turtle
t = turtle.Turtle()

t.forward(100)
t.right(90)      # 90 degrees જમણે
t.forward(50)
t.left(45)       # 45 degrees ડાબે
t.setheading(180) # પશ્ચિમ તરફ (180 degrees)

```

- **Relative**: right/left વર્તમાન direction થી સાપેક્ષ છે
- **Absolute**: setheading 0-360 degree system વાપરે છે
- **Navigation**: જટિલ 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.

```
import turtle

def draw_rainbow():
    # Setup screen
    wn = turtle.Screen()
    wn.bgcolor("skyblue")

    # Setup turtle
    t = turtle.Turtle()
    t.speed(5)
    t.pensize(10)

    # Rainbow colors (VIBGYOR reversed)
    colors = ['violet', 'indigo', 'blue', 'green',
              'yellow', 'orange', 'red']

    # શરૂઆતના position parameters
    radius = 180

    # Arcs દોરો
    for color in colors:
        t.penup()
        t.goto(0, -50)    # Center bottom
        t.setheading(90)  # ઉપર તરફ
        t.right(90)       # જમણી તરફ (0 deg)
        t.forward(radius)  # Arc ની શરૂઆતમાં જાઓ
        t.left(90)        # ફરી ઉપર તરફ face કરો
        t.pendown()

        t.pencolor(color)
        t.circle(radius, 180) # Semi-circle દોરો

        radius -= 20      # આગલા color માટે radius ઘટાડો

    t.hideturtle()
    turtle.done()

draw_rainbow()

```

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

મેમરી ટ્રીક

Colors Loop Radius Circle