Subject Name (Gujarati)

4321602 -- Summer 2024

Semester 1 Study Material

Detailed Solutions and Explanations

પ્રશ્ન 1(અ) [3 ગુણ]

પાયથનમાં ટપલ અને લિસ્ટ વચ્ચેનો તફાવત લખો.

જવાબ

લક્ષણ	ટપલ	લિસ્ટ
મ્યુટેબિલિટી	ઇમ્યુટેબલ (બદલી શકાતું નથી)	મ્યુટેબલ (બદલી શકાય છે)
સિન્ટેક્સ	() સાથે બનાવાય છે	[] સાથે બનાવાય છે
પ્રદર્શન	ઝડપી	ધીમું
મેથડ્સ	મર્યાદિત મેથડ્સ (count, index)	ઘણી મેથડ્સ (append, remove, વગેરે)

- **મેમરી કાર્યક્ષમ**: ટપલ લિસ્ટ કરતાં ઓછી મેમરી વાપરે છે
- **ઉપયોગ**: સ્થિર ડેટા માટે ટપલ, ગતિશીલ ડેટા માટે લિસ્ટ

મેમરી ટ્રીક

``ટપલ ટાઇટ, લિસ્ટ લૂઝ''

પ્રશ્ન 1(બ) [4 ગુણ]

સેટ સમજાવો અને પાયથનમાં સેટ કેવી રીતે બનાવાય છે?

જવાબ

સેટ એ પાયથનમાં અનોખા તત્વોનો અક્રમાંકિત સંગ્રહ છે.
સેટ બનાવવાની રીતો:

```
\#
my\_set = set()

\#
fruits = \{"apple", "banana", "orange"\}

\#
numbers = set([1, 2, 3, 4])
```

- **અનોખા તત્વો**: ડુપ્લિકેટની મંજૂરી નથી
- **અક્રમાંકિત**: તત્વોનો કોઈ ચોક્કસ ક્રમ નથી
- **ઓપરેશન્સ**: યુનિયન, ઇન્ટરસેક્શન, ડિફરન્સ સપોર્ટેડ

મેમરી ટ્રીક

``સેટ સ્પેશિયલ - અનોખા અને અક્રમાંકિત''

પ્રશ્ન 1(ક) [7 ગુણ]

પાયથનમાં ડિક્શનરી એટલે શું? બે ડિક્શનરીને નવી ડિક્શનરીમાં જોડવા માટેનો પ્રોગ્રામ લખો.

જવાબ

ડિક્શનરી એ પાયથનમાં કી-વેલ્યુ પેર્સનો ક્રમાંકિત સંગ્રહ છે.
પ્રોગ્રામ:

```
\#
dict1 = \{1: 10, 2: 20\}
dict2 = \{3: 30, 4: 40\}

\#    1: update()
result1 = dict1.copy()
result1.update(dict2)

\#    2: **
result2 = \{**dict1, **dict2\}

print("   :", result2)
\#    : \{1: 10, 2: 20, 3: 30, 4: 40\}
```

- **કી-વેલ્યુ પેર્સ**: દરેક તત્વમાં કી અને વેલ્યુ હોય છે
- **મ્યુટેબલ**: બનાવ્યા પછી બદલી શકાય છે
- **ઝડપી એક્સેસ**: O(1) સરેરાશ સમય જટિલતા

મેમરી ટ્રીક

``ડિક્શનરી ડાયનેમિક કી-વેલ્યુ સ્ટોર છે''

પ્રશ્ન 1(ક) અથવા [7 ગુણ]

પાયથનમાં લિસ્ટ એટલે શું? એક પ્રોગ્રામ લખો જે સૂચિમાંથી મહત્તમ અને ન્યૂનતમ નંબરો શોધે.

જવાબ

લિસ્ટ એ પાયથનમાં તત્વોનો ક્રમાંકિત, મ્યુટેબલ સંગ્રહ છે.
પ્રોગ્રામ:

```
\#
numbers = [45, 12, 78, 23, 56, 89, 34]

\#
maximum = max(numbers)
minimum = min(numbers)

print(f"   : \{maximum\}")
print(f"   : \{minimum\}")

\#
max\_val = numbers[0]
min\_val = numbers[0]
for num in numbers:
    if num {} max\_val:
        max\_val = num
    if num {} min\_val:
        min\_val = num
```

- **ક્રમાંકિત**: તત્વો ઇન્સર્શન ઓર્ડર જાળવે છે
- **ઇન્ડેક્સિંગ**: ઇન્ડેક્સ [0, 1, 2...] વાપરીને એક્સેસ

- **બિલ્ટ-ઇન ફંક્શન્સ**: min(), max(), len() ઉપલબ્ધ

મેમરી ટ્રીક

``લિસ્ટ લિનિયર અને ઇન્ડેક્સ્ડ છે''

પ્રશ્ન 2(અ) [3 ગુણ]

નેસ્ટેડ ટપલને ઉદાહરણ સાથે સમજાવો.

જવાબ

નેસ્ટેડ ટપલ એ ટપલ છે જેમાં અન્ય ટપલ તત્વો તરીકે હોય છે.
ઉદાહરણ:

```
\#
student\_data = (
    ("John", 85, "A"),
    ("Alice", 92, "A+"),
    ("Bob", 78, "B")
)

\#
print(student\_data[0][1])  \#    : 85
print(student\_data[1][0])  \#    : Alice
```

- **બહુ-પરિમાણીય**: ટપલની અંદર ટપલ
- **ઇન્ડેક્સિંગ**: બહુવિધ ઇન્ડિસેસ [i][j] વાપરો
- **ઇમ્યુટેબલ**: નેસ્ટેડ તત્વો બદલી શકાતા નથી

મેમરી ટ્રીક

``નેસ્ટેડ મતલબ ટપલની અંદર ટપલ''

પ્રશ્ન 2(બ) [4 ગુણ]

રેન્ડમ મોડ્યુલ શું છે? ઉદાહરણ સાથે સમજાવો.

જવાબ

રેન્ડમ મોડ્યુલ રેન્ડમ નંબરો જનરેટ કરે છે અને રેન્ડમ ઓપરેશન્સ કરે છે.
ઉદાહરણ:

```
import random

\#
num = random.randint(1, 10)
print(f"     : \{num\}")

\#
colors = ["  ", "   ", "   "]
choice = random.choice(colors)
print(f"     : \{choice\}")

\#
decimal = random.random()
```

```
print(f"      : \{decimal\}")
```

- **ઇમ્પોર્ટ જરૂરી**: import random
- **વિવિધ ફંક્શન્સ**: randint(), choice(), random()
- **ઉપયોગી**: ગેમ્સ, સિમ્યુલેશન, ટેસ્ટિંગ માટે

મેમરી ટ્રીક

``રેન્ડમ વસ્તુઓને અણધારી બનાવે છે''

પ્રશ્ન 2(ક) [7 ગુણ]

પેકેજને ઇમ્પોર્ટ કરવાની વિવિધ રીતો સમજાવો. તેનું એક ઉદાહરણ આપો.

જવાબ

ઇમ્પોર્ટ મેથડ્સ:

મેથડ	સિન્ટેક્સ	ઉપયોગ
નોર્મલ ઇમ્પોર્ટ	`import package`	package.function()
ફ્રોમ ઇમ્પોર્ટ	`from package import function`	function()
બધું ઇમ્પોર્ટ	`from package import *`	function()
એલિયાસ ઇમ્પોર્ટ	`import package as alias`	alias.function()

ઉદાહરણ:

```
\#
import math
result1 = math.sqrt(16)

\#
from math import sqrt
result2 = sqrt(16)

\#
import math as m
result3 = m.sqrt(16)

\#       (     )
from math import *
result4 = sqrt(16)
```

- **નેમસ્પેસ**: નોર્મલ ઇમ્પોર્ટ અલગ નેમસ્પેસ રાખે છે
- **ડાયરેક્ટ એક્સેસ**: ફ્રોમ ઇમ્પોર્ટ ડાયરેક્ટ ફંક્શન કોલ કરવાની મંજૂરી આપે છે
- **એલિયાસ**: સુવિધા માટે ટૂંકા નામો

મેમરી ટ્રીક

``ઇમ્પોર્ટ મેથડ્સ: નોર્મલ, ફ્રોમ, બધું, એલિયાસ''

પ્રશ્ન 2(અ) અથવા [3 ગુણ]

પાયથનમાં ડિક્શનરીના ગુણધર્મો લખો.

જવાબ

ડિક્શનરીના ગુણધર્મો:

ગુણધર્મ	વર્ણન
ક્રમાંકિત	ઇન્સર્શન ઓર્ડર જાળવે છે (Python 3.7+)
મ્યુટેબલ	બનાવ્યા પછી બદલી શકાય છે
કી-અનોખી	ડુપ્લિકેટ કીઓની મંજૂરી નથી
હેટેરોજીનિયસ	કીઓ અને વેલ્યુઝ અલગ પ્રકારના હોઈ શકે

- **ઝડપી એક્સેસ**: O(1) સરેરાશ લુકઅપ ટાઇમ
- **ડાયનેમિક સાઇઝ**: વધી અથવા ઘટી શકે છે
- **કી પ્રતિબંધો**: કીઓ ઇમ્યુટેબલ હોવી જોઈએ

મેમરી ટ્રીક

``ડિક્શનરી ક્રમાંકિત, મ્યુટેબલ, અનોખી, હેટેરોજીનિયસ છે''

પ્રશ્ન 2(બ) અથવા [4 ગુણ]

પાયથનમાં dir() ફંક્શન શું છે. ઉદાહરણ સાથે સમજાવો.

જવાબ

dir() ફંક્શન ઓબ્જેક્ટના બધા એટ્રિબ્યુટ્સ અને મેથડ્સ રિટર્ન કરે છે.
ઉદાહરણ:

```
\#
text = "hello"
attributes = dir(text)
print(attributes[:5])  \#    5

\#
print("upper" in dir(text))  \# True

\#
import math
math\_methods = dir(math)
print("sqrt" in math\_methods)  \# True

\#
class MyClass:
    def my\_method(self):
        pass

obj = MyClass()
print(dir(obj))
```

- **ઇન્ટ્રોસ્પેક્શન**: ઓબ્જેક્ટ પ્રોપર્ટીઝ તપાસે છે
- **ડિબગિંગ**: ઉપલબ્ધ મેથડ્સ શોધવામાં મદદ કરે છે
- **બધા ઓબ્જેક્ટ્સ**: કોઈપણ Python ઓબ્જેક્ટ સાથે કામ કરે છે

મેમરી ટ્રીક

``dir() ઓબ્જેક્ટ એટ્રિબ્યુટ્સની ડિરેક્ટરી બતાવે છે''

પ્રશ્ન 2(ક) અથવા [7 ગુણ]

બે સંખ્યાઓનો સરવાળો શોધવા માટે મોડ્યુલને વ્યાખ્યાયિત કરવા માટે પ્રોગ્રામ લખો. બીજા પ્રોગ્રામમાં મોડ્યુલ ઇમ્પોર્ટ કરો.

જવાબ

મોડ્યુલ ફાઇલ (calculator.py):

```
\# calculator.py
def add\_numbers(a, b):
    """               """
    return a + b

def multiply\_numbers(a, b):
    """               """
    return a * b

def get\_sum(num1, num2):
    """         """
    result = num1 + num2
    return result
```

મુખ્ય પ્રોગ્રામ:

```
\# main.py
import calculator

\#
result1 = calculator.add\_numbers(10, 20)
print(f"    : \{result1\}")

\#
from calculator import get\_sum
result2 = get\_sum(15, 25)
print(f"               : \{result2\}")
```

- **મોડ્યુલ બનાવટ**: ફંક્શન્સને .py ફાઇલમાં સેવ કરો
- **ઇમ્પોર્ટ**: ઇમ્પોર્ટ સ્ટેટમેન્ટ વાપરીને એક્સેસ કરો
- **કોડ પુનઃઉપયોગ**: એક જ મોડ્યુલને અનેક પ્રોગ્રામમાં વાપરો

મેમરી ટ્રીક

``મોડ્યુલ કોડને પુનઃઉપયોગી અને વ્યવસ્થિત બનાવે છે''

પ્રશ્ન 3(અ) [3 ગુણ]

રનટાઇમ એરર અને લોજિકલ એરર શું છે. ઉદાહરણ સાથે સમજાવો.

જવાબ

એરર પ્રકાર	વ્યાખ્યા	ઉદાહરણ
રનટાઇમ એરર	પ્રોગ્રામ એક્ઝિક્યુશન દરમિયાન થાય છે	શૂન્ય વડે ભાગાકાર, ફાઇલ ન મળે
લોજિકલ એરર	પ્રોગ્રામ ચાલે છે પણ ખોટો આઉટપુટ આપે છે	ખોટું ફોર્મ્યુલા, ખોટી કન્ડિશન

ઉદાહરણો:

```
\#
x = 10
y = 0
result = x / y  \# ZeroDivisionError

\#
def calculate\_area(radius):
    return 3.14 * radius  \# radius * radius
```

- **રનટાઇમ**: પ્રોગ્રામ એક્ઝિક્યુશન ક્રેશ કરે છે
- **લોજિકલ**: પ્રોગ્રામ ચાલુ રહે છે પણ ખોટું પરિણામ

મેમરી ટ્રીક

``રનટાઇમ ક્રેશ કરે, લોજિકલ કન્ફ્યુઝ કરે''

પ્રશ્ન 3(બ) [4 ગુણ]

Except ક્લોઝના મુદ્દાઓ લખો અને તેને સમજાવો.

જવાબ

Except ક્લોઝ try-except બ્લોકમાં ચોક્કસ exceptions ને હેન્ડલ કરે છે.
મુખ્ય મુદ્દાઓ:

લક્ષણ	વર્ણન
સિન્ટેક્સ	`except ExceptionType:`
બહુવિધ	બહુવિધ except બ્લોક્સ હોઈ શકે
જનરિક	`except:` બધા exceptions પકડે છે
વેરિયેબલ	`except Exception as e:` એરર સ્ટોર કરે છે

```
try:
    number = int(input("        : "))
    result = 10 / number
except ValueError:
    print("      ")
except ZeroDivisionError:
    print("                  ")
except Exception as e:
    print(f"  : \{e\}")
```

- **સ્પેસિફિક હેન્ડલિંગ**: અલગ exceptions અલગ રીતે હેન્ડલ થાય
- **એરર રિકવરી**: હેન્ડલિંગ પછી પ્રોગ્રામ ચાલુ રહે

મેમરી ટ્રીક

``Except પકડે છે અને એરર હેન્ડલ કરે છે''

પ્રશ્ન 3(ક) [7 ગુણ]

Divide by zero Exception ને કેચ કરવા માટેનો પ્રોગ્રામ લખો. finally બ્લોકનો ઉપયોગ કરો.

જવાબ

```
def safe\_division():
    try:
        \#
        numerator = float(input("       : "))
        denominator = float(input("       : "))

        \#
        result = numerator / denominator
        print(f"    : \{numerator\} / \{denominator\} = \{result\}")

    except ZeroDivisionError:
        print("  :                    !")
        print("        {-          "})

    except ValueError:
        print("  :                        ")

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

    finally:
        print("             ")
        print("                  ")

\#
safe\_division()
```

- **Try બ્લોક**: જોખમી કોડ સમાવે છે
- **Except**: ZeroDivisionError ને સ્પેસિફિકલી હેન્ડલ કરે છે
- **Finally**: exception હોય કે ન હોય હંમેશા એક્ઝિક્યુટ થાય છે

મેમરી ટ્રીક

``Try જોખમી કોડ, Except એરર હેન્ડલ કરે, Finally હંમેશા ચાલે''

પ્રશ્ન 3(અ) અથવા [3 ગુણ]

બિલ્ટ-ઇન exceptions શું છે અને તેના પ્રકારો લખો.

જવાબ

બિલ્ટ-ઇન Exception પ્રકારો:

પ્રકાર	વર્ણન	ઉદાહરણ
ValueError	ઓપરેશન માટે અયોગ્ય વેલ્યુ	int(``abc'')
TypeError	ખોટો ડેટા પ્રકાર	``5'' + 5
IndexError	ઇન્ડેક્સ રેન્જની બહાર	list[10] for 5-element list
KeyError	ડિક્શનરીમાં કી ન મળે	dict[``missing_key'']
FileNotFoundError	ફાઇલ અસ્તિત્વમાં નથી	open(``missing.txt'')

```
\#
try:
    int("hello")  \# ValueError
    "5" + 5       \# TypeError
    [1,2,3][5]    \# IndexError
except (ValueError, TypeError, IndexError) as e:
    print(f"  : \{type(e).\_\_name\_\_\}")
```

મેમરી ટ્રીક

``Value, Type, Index, Key, File - સામાન્ય એરર પ્રકારો''

પ્રશ્ન 3(બ) અથવા [4 ગુણ]

સિન્ટેક્સ એરર સમજાવો અને આપણે તેને કેવી રીતે ઓળખી શકીએ? એક ઉદાહરણ આપો.

જવાબ

સિન્ટેક્સ એરર ત્યારે થાય છે જ્યારે Python ખોટા સિન્ટેક્સને કારણે કોડ parse કરી શકતું નથી.
ઓળખવાની રીતો:

મેથડ	વર્ણન
Python interpreter	લાઇન નંબર સાથે એરર મેસેજ બતાવે છે
IDE highlighting	કોડ એડિટર્સ સિન્ટેક્સ એરર હાઇલાઇટ કરે છે
Error message	એરરનું ચોક્કસ સ્થાન બતાવે છે

ઉદાહરણો:

```
\#
if x {} 5
    print("  ")  \# SyntaxError

\#
print("Hello"  \# SyntaxError

\#    indentation
def my\_function():
print("Hello")  \# IndentationError

\#
2variable = 10  \# SyntaxError
```

- **ડિટેક્શન**: પ્રોગ્રામ એક્ઝિક્યુશન પહેલાં
- **એરર મેસેજ**: લાઇન અને કેરેક્ટર પોઝિશન બતાવે છે
- **સામાન્ય કારણો**: ગુમ કોલન, બ્રેકેટ્સ, ખોટું indentation

મેમરી ટ્રીક

``સિન્ટેક્સ એરર કોડને શરૂ થવાથી રોકે છે''

પ્રશ્ન 3(ક) અથવા [7 ગુણ]

પાયથનમાં એક્સેપશન હેન્ડલિંગ શું છે? યોગ્ય ઉદાહરણ સાથે સમજાવો.

જવાબ

Exception Handling એ રનટાઇમ એરર્સને પ્રોગ્રામ ક્રેશ કર્યા વિના gracefully હેન્ડલ કરવાની પદ્ધતિ છે.
સ્ટ્રક્ચર:

```
try:
    \#
    pass
except SpecificException:
    \#
    pass
except Exception as e:
    \#
    pass
else:
    \# exception
    pass
finally:
    \#
    pass
```

સંપૂર્ણ ઉદાહરણ:

```
def file\_processor():
    filename = None
    try:
        filename = input("          : ")
        with open(filename, {r}) as file:
            content = file.read()
            numbers = [int(x) for x in content.split()]
            average = sum(numbers) / len(numbers)
            print(f"   : \{average\}")

    except FileNotFoundError:
        print(f"  :    {}\{filename\}{    "})

    except ValueError:
        print("  :       {-        "})

    except ZeroDivisionError:
        print("  :               ")

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

    else:
        print("             ")

    finally:
        print("               ")

\#
file\_processor()
```

- **Graceful handling**: એરર પછી પ્રોગ્રામ ચાલુ રહે છે
- **Multiple exceptions**: અલગ એરર પ્રકારો અલગ રીતે હેન્ડલ થાય છે
- **Else clause**: માત્ર exception ન હોય તો જ ચાલે છે
- **Finally clause**: cleanup માટે હંમેશા એક્ઝિક્યુટ થાય છે

મેમરી ટ્રીક

``Try-Except-Else-Finally: સંપૂર્ણ એરર હેન્ડલિંગ''

પ્રશ્ન 4(અ) [3 ગુણ]

ફાઇલમાં આપણે કેવા પ્રકારની વિવિધ ઓપરેશન કરી શકીએ છીએ?

જવાબ

ફાઇલ ઓપરેશન્સ:

ઓપરેશન	વર્ણન	મેથડ
Read	ફાઇલ કન્ટેન્ટ વાંચો	read(), readline(), readlines()
Write	ફાઇલમાં ડેટા લખો	write(), writelines()
Append	અંતમાં ડેટા ઉમેરો	`a' મોડ સાથે open
Create	નવી ફાઇલ બનાવો	`w' અથવા `x' મોડ સાથે open
Delete	ફાઇલ રીમૂવ કરો	os.remove()
Seek	ફાઇલ પોઇન્ટર ખસેડો	seek()

```
\#
with open({file.txt}, {w}) as f:
    f.write("Hello")  \# Write

with open({file.txt}, {r}) as f:
    content = f.read()  \# Read
```

મેમરી ટ્રીક

``Read, Write, Append, Create, Delete, Seek''

પ્રશ્ન 4(બ) [4 ગુણ]

ફાઇલ મોડ્સની યાદી આપો. કોઈપણ ચાર મોડનું વર્ણન લખો.

જવાબ

ફાઇલ મોડ્સ:

મોડ	વર્ણન	હેતુ
`r'	Read મોડ (default)	અસ્તિત્વમાં છે તે ફાઇલ વાંચો
`w'	Write મોડ	નવી બનાવો અથવા અસ્તિત્વમાં છે તેને overwrite કરો
`a'	Append મોડ	અસ્તિત્વમાં છે તે ફાઇલના અંતમાં ઉમેરો
`x'	Exclusive creation	નવી ફાઇલ બનાવો, અસ્તિત્વમાં હોય તો fail
`b'	Binary મોડ	binary ફાઇલ્સ હેન્ડલ કરો
`t'	Text મોડ (default)	text ફાઇલ્સ હેન્ડલ કરો
`+'	Read અને write	બંને ઓપરેશન્સની મંજૂરી

ચાર મોડનું વર્ણન:

1. **`r' (Read)**: માત્ર વાંચવા માટે ફાઇલ ખોલે છે, ફાઇલ પોઇન્ટર શરૂઆતમાં
2. **`w' (Write)**: લખવા માટે ખોલે છે, ફાઇલ truncate કરે છે અથવા નવી બનાવે છે
3. **`a' (Append)**: લખવા માટે ખોલે છે, ફાઇલ પોઇન્ટર ફાઇલના અંતમાં
4. **`r+' (Read/Write)**: વાંચવા અને લખવા બંને માટે ખોલે છે

મેમરી ટ્રીક

``Read, Write, Append, eXclusive - મુખ્ય ફાઇલ મોડ્સ''

પ્રશ્ન 4(ક) [7 ગુણ]

ફાઇલમાંના બધા શબ્દોને સૉર્ટ કરવા માટે એક પ્રોગ્રામ લખો અને તેને લિસ્ટમાં મૂકો.

જવાબ

```
def sort\_words\_from\_file():
    try:
        \#
        filename = input("          : ")

        \#
        with open(filename, {r}, encoding={utf{-}8}) as file:
            content = file.read()

        \#
        words = content.lower().split()

        \# Punctuation
        import string
        clean\_words = []
        for word in words:
            clean\_word = word.translate(str.maketrans({}, {}, string.punctuation))
            if clean\_word:  \#      {-         }
                clean\_words.append(clean\_word)

        \#
        sorted\_words = sorted(clean\_words)

        \#
        print("         :")
        print(sorted\_words)

        \#
        with open({sorted\_words.txt}, {w}, encoding={utf{-}8}) as output\_file:
            for word in sorted\_words:
                output\_file.write(word + {}{n}{})

        print(f"     : \{len(sorted\_words)\}")
        print("          {sorted\_words.txt       "})

    except FileNotFoundError:
        print("  :       ")
    except Exception as e:
        print(f"  : \{e\}")

\#
sort\_words\_from\_file()
```

- **ફાઇલ રીડિંગ**: સંપૂર્ણ ફાઇલ કન્ટેન્ટ વાંચો
- **શબ્દ પ્રોસેસિંગ**: શબ્દોને વિભાજિત, સાફ અને સૉર્ટ કરો
- **લિસ્ટ બનાવટ**: સૉર્ટ થયેલા શબ્દોને લિસ્ટમાં સ્ટોર કરો

મેમરી ટ્રીક

``વાંચો, વિભાજિત કરો, સાફ કરો, સૉર્ટ કરો, સેવ કરો''

પ્રશ્ન 4(અ) અથવા [3 ગુણ]

ફાઇલ હેન્ડલિંગ શું છે? ફાઇલ્સ હેન્ડલિંગ ઓપરેશનની યાદી બનાવો અને તેને સમજાવો.

જવાબ

ફાઇલ હેન્ડલિંગ એ ડેટાને કાયમી ધોરણે સ્ટોર અને retrieve કરવા માટે ફાઇલો સાથે કામ કરવાની પ્રક્રિયા છે.
ફાઇલ હેન્ડલિંગ ઓપરેશન્સ:

ઓપરેશન	ફંક્શન	વર્ણન
Open	open()	ચોક્કસ મોડમાં ફાઇલ ખોલે છે
Read	read(), readline()	ફાઇલમાંથી ડેટા વાંચે છે
Write	write(), writelines()	ફાઇલમાં ડેટા લખે છે
Close	close()	ફાઇલ બંધ કરે છે અને resources મુક્ત કરે છે
Seek	seek()	ફાઇલ પોઇન્ટર પોઝિશન ખસેડે છે
Tell	tell()	વર્તમાન ફાઇલ પોઇન્ટર પોઝિશન રિટર્ન કરે છે

```
\#
file = open({data.txt}, {w})  \# Open
file.write({Hello World})     \# Write
file.close()                  \# Close

file = open({data.txt}, {r})  \#
content = file.read()         \# Read
file.close()                  \# Close
```

મેમરી ટ્રીક

``Open, Read, Write, Close - બેસિક ફાઇલ સાઇકલ''

પ્રશ્ન 4(બ) અથવા [4 ગુણ]

ઉદાહરણ સાથે load() મેથડ સમજાવો.

જવાબ

load() મેથડ ફાઇલમાંથી ડેટાને deserialize કરવા માટે વપરાય છે (સામાન્ય રીતે pickle મોડ્યુલ સાથે).
Pickle load() ઉદાહરણ:

```
import pickle

\#    ,
data\_to\_save = \{
    {name}: {John},
    {age}: 25,
    {scores}: [85, 92, 78]
\}

\#
with open({data.pkl}, {wb}) as file:
    pickle.dump(data\_to\_save, file)

\#
with open({data.pkl}, {rb}) as file:
    loaded\_data = pickle.load(file)

print("       :", loaded\_data)
print("  :", loaded\_data[{name}])
print("   :", loaded\_data[{scores}])
```

JSON load() ઉદાહરણ:

```
import json

\# JSON
with open({config.json}, {r}) as file:
    config = json.load(file)

print("    :", config)
```

- **Deserialization**: ફાઇલ ડેટાને પાછું Python objects માં કન્વર્ટ કરે છે
- **Binary મોડ**: pickle ફાઇલ્સ માટે `rb' મોડ વાપરો
- **Error handling**: FileNotFoundError હેન્ડલ કરો

મેમરી ટ્રીક

``load() ફાઇલ ડેટાને પાછું Python objects માં લાવે છે''

પ્રશ્ન 4(ક) અથવા [7 ગુણ]

એક પ્રોગ્રામ લખો જે ટેક્સ્ટ ફાઇલને ઇનપુટ કરે. પ્રોગ્રામે ફાઇલમાંના તમામ યુનીક શબ્દોને મૂળાક્ષરોના ક્રમમાં છાપવા જોઈએ.

જવાબ

```
def find\_unique\_words():
    try:
        \#
        filename = input("            : ")

        \#
        with open(filename, {r}, encoding={utf{-}8}) as file:
            content = file.read().lower()

        \#
        import re
        import string

        \# Punctuation
        words = re.findall(r{}{b}[a{-zA{-}Z]}+{b}{}, content.lower())

        \#              set
        unique\_words = set(words)

        \#
        sorted\_unique\_words = sorted(list(unique\_words))

        \#
        print("{n}               :")
        print("{-"} * 40)

        for i, word in enumerate(sorted\_unique\_words, 1):
            print(f"\{i:3d\}. \{word\}")

        print(f"{n}       : \{len(sorted\_unique\_words)\}")

        \#
        with open({unique\_words\_output.txt}, {w}, encoding={utf{-}8}) as output\_file:
            output\_file.write("               {n}")
            output\_file.write("=" * 40 + "{nn}")
```

```
            for word in sorted\_unique\_words:
                output\_file.write(word + {}{n}{})

        print("     {unique\_words\_output.txt       "})

    except FileNotFoundError:
        print(f"  :     {}\{filename\}{     "})
    except PermissionError:
        print("  :                 ")
    except Exception as e:
        print(f"       : \{e\}")

\#
def create\_sample\_file():
    sample\_text = """
    Python                  .
    Python            Python       .
    Python                              .
    """

    with open({sample.txt}, {w}, encoding={utf{-}8}) as f:
        f.write(sample\_text)
    print("       {sample.txt        "})

\#
create\_sample\_file()
find\_unique\_words()
```

- **Regular expressions**: માત્ર અક્ષરવાળા શબ્દો એક્સ્ટ્રેક્ટ કરે છે
- **Set ડેટા સ્ટ્રક્ચર**: આપમેળે ડુપ્લિકેટ્સ રીમૂવ કરે છે
- **Sorted ફંક્શન**: શબ્દોને મૂળાક્ષરોના ક્રમમાં ગોઠવે છે
- **ફાઇલ આઉટપુટ**: ભાવિ સંદર્ભ માટે પરિણામો સેવ કરે છે

મેમરી ટ્રીક

``વાંચો, એક્સ્ટ્રેક્ટ કરો, યુનીક, સૉર્ટ, દર્શાવો''

પ્રશ્ન 5(અ) [3 ગુણ]

નીચેના ટર્ટલ ફંક્શનને યોગ્ય ઉદાહરણ સાથે સમજાવો. (a) turn() (b) move().

જવાબ

નોંધ: સ્ટાન્ડર્ડ ટર્ટલ મોડ્યુલ `turn()` ને બદલે `left()`, `right()` અને `move()` ને બદલે `forward()`, `backward()` વાપરે છે.
ટર્ટલ મૂવમેન્ટ ફંક્શન્સ:

ફંક્શન	હેતુ	ઉદાહરણ
left(angle)	ડિગ્રીમાં ડાબે ફેરવો	turtle.left(90)
right(angle)	ડિગ્રીમાં જમણે ફેરવો	turtle.right(45)
forward(distance)	આગળ ખસો	turtle.forward(100)
backward(distance)	પાછળ ખસો	turtle.backward(50)

```
import turtle

\#
t = turtle.Turtle()

\#
t.left(90)     \#    90
t.right(45)    \#    45

\#
t.forward(100)  \#    100
t.backward(50)  \#     50

\#
turtle.done()
```

મેમરી ટ્રીક

``ટર્ન દિશા બદલે છે, મૂવ પોઝિશન બદલે છે''

પ્રશ્ન 5(બ) [4 ગુણ]

ટર્ટલની દિશા બદલવાની વિવિધ ઇનબિલ્ટ પદ્ધતિઓ સમજાવો.

જવાબ

દિશા કન્ટ્રોલ મેથડ્સ:

મેથડ	વર્ણન	ઉદાહરણ
left(angle)	વામાવર્ત ફેરવો	turtle.left(90)
right(angle)	દક્ષિણાવર્ત ફેરવો	turtle.right(45)
setheading(angle)	ચોક્કસ દિશા સેટ કરો	turtle.setheading(0)
towards(x, y)	કોઓર્ડિનેટ્સ તરફ નિર્દેશ કરો	turtle.setheading(turtle.towards(100, 100))

```
import turtle

t = turtle.Turtle()

\#
t.left(90)        \#    90^
t.right(45)       \#    45^

\#
t.setheading(0)   \#            (0^)
t.setheading(90)  \#             (90^)

\#
angle = t.towards(100, 100)
t.setheading(angle)
```

- **સંબંધિત**: left() અને right() વર્તમાન દિશા બદલે છે
- **ચોક્કસ**: setheading() ચોક્કસ દિશા સેટ કરે છે
- **કોઓર્ડિનેટ-આધારિત**: towards() પોઇન્ટ તરફની દિશા ગણે છે

મેમરી ટ્રીક

``ડાબે-જમણે સંબંધિત, હેડિંગ ચોક્કસ, તરફ ગણતરી કરે''

પ્રશ્ન 5(ક) [7 ગુણ]

ટર્ટલનો ઉપયોગ કરીને ચોરસ, લંબચોરસ અને વર્તુળ દોરવા માટેનો પ્રોગ્રામ લખો.

જવાબ

```
import turtle

def draw\_shapes():
    \#
    screen = turtle.Screen()
    screen.title("            ")
    screen.bgcolor("white")
    screen.setup(800, 600)

    \#
    pen = turtle.Turtle()
    pen.speed(3)
    pen.color("blue")

    \#
    pen.penup()
    pen.goto({-}200, 100)
    pen.pendown()
    pen.write("  ", font=("Arial", 12, "bold"))
    pen.goto({-}200, 50)

    for i in range(4):
        pen.forward(80)
        pen.right(90)

    \#
    pen.penup()
    pen.goto(0, 100)
    pen.pendown()
    pen.color("red")
    pen.write("   ", font=("Arial", 12, "bold"))
    pen.goto(0, 50)

    for i in range(2):
        pen.forward(120)  \#
        pen.right(90)
        pen.forward(60)   \#
        pen.right(90)

    \#
    pen.penup()
    pen.goto(200, 100)
    pen.pendown()
    pen.color("green")
    pen.write("  ", font=("Arial", 12, "bold"))
    pen.goto(200, 50)

    pen.circle(40)  \# Radius = 40

    \#
```

```
    pen.hideturtle()
    screen.exitonclick()

\#
def draw\_square(turtle\_obj, size):
    """               """
    for \_ in range(4):
        turtle\_obj.forward(size)
        turtle\_obj.right(90)

def draw\_rectangle(turtle\_obj, width, height):
    """                  """
    for \_ in range(2):
        turtle\_obj.forward(width)
        turtle\_obj.right(90)
        turtle\_obj.forward(height)
        turtle\_obj.right(90)

def draw\_circle(turtle\_obj, radius):
    """   radius         """
    turtle\_obj.circle(radius)

\#
draw\_shapes()
```

- **ચોરસ**: $90° 4$
- **લંબચોરસ**: સમાન બાજુઓની 2 જોડી
- **વર્તુળ**: radius સાથે બિલ્ટ-ઇન circle() મેથડ

મેમરી ટ્રીક

``ચોરસ: 4 સમાન બાજુ, લંબચોરસ: 2 જોડી, વર્તુળ: radius મેથડ''

પ્રશ્ન 5(અ) અથવા [3 ગુણ]

ટર્ટલમાં પેન કમાન્ડના વિવિધ પ્રકારો કયા છે? તે બધાને સમજાવો.

જવાબ

પેન કન્ટ્રોલ કમાન્ડ્સ:

કમાન્ડ	હેતુ	ઉદાહરણ
penup()	પેન ઉઠાવો (દોરવું નહીં)	turtle.penup()
pendown()	પેન નીચે મૂકો (દોરવાનું શરૂ કરો)	turtle.pendown()
pensize(width)	પેનની જાડાઈ સેટ કરો	turtle.pensize(5)
pencolor(color)	પેનનો રંગ સેટ કરો	turtle.pencolor(``red'')
fillcolor(color)	ભરવાનો રંગ સેટ કરો	turtle.fillcolor(``blue'')
begin_fill()	આકાર ભરવાનું શરૂ કરો	turtle.begin_fill()
end_fill()	આકાર ભરવાનું બંધ કરો	turtle.end_fill()

```
import turtle

t = turtle.Turtle()

\#
t.penup()           \#
t.goto(50, 50)      \#
t.pendown()         \#
t.pensize(3)        \#
t.pencolor("red")   \#
```

મેમરી ટ્રીક

``Up-Down દોરવાનું કન્ટ્રોલ કરે, Size-Color દેખાવ કન્ટ્રોલ કરે''

પ્રશ્ન 5(બ) અથવા [4 ગુણ]

ટર્ટલનો ઉપયોગ કરીને વર્તુળ અને સ્ટારના આકાર દોરો અને તેમને લાલ રંગથી ભરો.

જવાબ

```
import turtle

def draw\_filled\_shapes():
    \#
    screen = turtle.Screen()
    screen.bgcolor("white")
    screen.title("          ")

    \#
    artist = turtle.Turtle()
    artist.speed(5)

    \#
    artist.penup()
    artist.goto({-}150, 0)
    artist.pendown()

    \#
    artist.color("red", "red")  \# pen color, fill color
    artist.begin\_fill()
    artist.circle(50)
    artist.end\_fill()

    \#
    artist.penup()
    artist.goto(100, 0)
    artist.pendown()

    \#
    artist.color("red", "red")
    artist.begin\_fill()

    \# 5{-         }
    for i in range(5):
        artist.forward(100)
        artist.right(144)

    artist.end\_fill()
```

```
    \#
    artist.penup()
    artist.goto({-}180, {-}80)
    artist.color("black")
    artist.write("     ", font=("Arial", 12, "bold"))

    artist.goto(70, {-}80)
    artist.write("     ", font=("Arial", 12, "bold"))

    \#
    artist.hideturtle()
    screen.exitonclick()

\#
draw\_filled\_shapes()
```

મુખ્ય મુદ્દાઓ:
- **begin_fill()**: આકાર ભરવાનું શરૂ કરો
- **end_fill()**: ભરવાનું પૂર્ણ કરો
- **color()**: pen અને fill બંને રંગો સેટ કરો
- **સ્ટાર angle**: 5-પોઇન્ટેડ સ્ટાર માટે 144°

મેમરી ટ્રીક

``Begin fill, આકાર દોરો, End fill = ભરેલા આકાર''

પ્રશ્ન 5(ક) અથવા [7 ગુણ]

ટર્ટલનો ઉપયોગ કરીને ભારતનો ઝંડો દોરવા માટેનો પ્રોગ્રામ લખો.

જવાબ

```
import turtle

def draw\_indian\_flag():
    \#
    screen = turtle.Screen()
    screen.bgcolor("white")
    screen.title("      ")
    screen.setup(800, 600)

    \#
    flag = turtle.Turtle()
    flag.speed(5)
    flag.pensize(2)

    \#
    flag\_width = 300
    flag\_height = 200

    \#
    start\_x = {-}150
    start\_y = 100

    \#
    flag.penup()
    flag.goto(start\_x {-} 20, start\_y + 50)
    flag.pendown()
```

```
flag.color("brown")
flag.pensize(8)
flag.setheading(270)  \#
flag.forward(400)

\#
flag.pensize(2)
flag.color("black")

\#        (  )
flag.penup()
flag.goto(start\_x, start\_y)
flag.pendown()
flag.color("orange", "orange")
flag.begin\_fill()
flag.setheading(0)

for \_ in range(2):
    flag.forward(flag\_width)
    flag.right(90)
    flag.forward(flag\_height // 3)
    flag.right(90)
flag.end\_fill()

\#        (  )
flag.penup()
flag.goto(start\_x, start\_y {-} flag\_height // 3)
flag.pendown()
flag.color("black", "white")
flag.begin\_fill()

for \_ in range(2):
    flag.forward(flag\_width)
    flag.right(90)
    flag.forward(flag\_height // 3)
    flag.right(90)
flag.end\_fill()

\#        (  )
flag.penup()
flag.goto(start\_x, start\_y {-} 2 * flag\_height // 3)
flag.pendown()
flag.color("green", "green")
flag.begin\_fill()

for \_ in range(2):
    flag.forward(flag\_width)
    flag.right(90)
    flag.forward(flag\_height // 3)
    flag.right(90)
flag.end\_fill()

\#       (  )
chakra\_center\_x = start\_x + flag\_width // 2
chakra\_center\_y = start\_y {-} flag\_height // 2

flag.penup()
flag.goto(chakra\_center\_x, chakra\_center\_y {-} 30)
flag.pendown()
flag.color("navy")
flag.pensize(3)
```

```
    \#
    flag.circle(30)

    \#
    flag.penup()
    flag.goto(chakra\_center\_x, chakra\_center\_y)
    flag.pendown()

    for i in range(24):  \#        24
        flag.setheading(i * 15)  \# 360/24 = 15
        flag.forward(30)
        flag.backward(30)

    \#
    flag.penup()
    flag.goto(chakra\_center\_x, chakra\_center\_y {-} 5)
    flag.pendown()
    flag.circle(5)

    \#
    flag.penup()
    flag.goto({-}100, 200)
    flag.color("black")
    flag.write("      ", font=("Arial", 16, "bold"))

    \#
    flag.hideturtle()
    screen.exitonclick()

\#
draw\_indian\_flag()
```

ઝંડાના ઘટકો:

- **કેસરી**: બહાદુરી અને બલિદાન (ઉપર)
- **સફેદ**: સત્ય અને શાંતિ (મધ્ય)
- **લીલો**: શ્રદ્ધા અને વીરતા (નીચે)
- **અશોક ચક્ર**: ઘેરા વાદળી રંગમાં 24-તીલીવાળું ચક્ર

મેમરી ટ્રીક

``કેસરી-સફેદ-લીલી પટ્ટીઓ 24-તીલીવાળા ચક્ર સાથે''