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:

```
# ખાલી સેટ
my_set = set()

# તત્વો સાથે સેટ
fruits = {"apple", "banana", "orange"}

# લિસ્ટમાંથી સેટ
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:

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

# મેથડ 1: update() નો ઉપયોગ
result1 = dict1.copy()
result1.update(dict2)

# મેથડ 2: ** ઓપરેટરનો ઉપયોગ
result2 = {**dict1, **dict2}

print("પરિણામ:", result2)
# આઉટપુટ: {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:

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

# મહત્તમ અને ન્યૂનતમ શોધો
maximum = max(numbers)
minimum = min(numbers)

print(f"મહત્તમ: {maximum}")
print(f"ન્યૂનતમ: {minimum}")

# મેન્યુઅલ મેથડ
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:

```
# નેસ્ટેડ ટપલ
student_data = (
    ("John", 85, "A"),
    ("Alice", 92, "A+"),
    ("Bob", 78, "B")
)

# તત્વોને એક્સેસ કરવું
print(student_data[0][1])  # આઉટપુટ: 85
print(student_data[1][0])  # આઉટપુટ: 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:

```python
import random

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

# લિસ્ટમાંથી રેન્ડમ પસંદગી
colors = ["લાલ", "નીલો", "લીલો"]
choice = random.choice(colors)
print(f"રેન્ડમ રંગ: {choice}")

# રેન્ડમ ફ્લોટ
decimal = random.random()
print(f"રેન્ડમ દશાંશ: {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:

```python
# નોર્મલ ઇમ્પોર્ટ
import math
result1 = math.sqrt(16)

# ફ્રોમ ઇમ્પોર્ટ
from math import sqrt
result2 = sqrt(16)

# એલિયાસ સાથે ઇમ્પોર્ટ
```

```
import math as m
result3 = m.sqrt(16)

# બધું ઇમ્પોર્ટ (ભલામણ નથી)
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:

```
# સ્ટ્રિંગના બધા એટ્રિબ્યુટ્સ
text = "hello"
```

```
attributes = dir(text)
print(attributes[:5])  # પ્રથમ 5 એટ્રિબ્યુટ્સ

# ઉપલબ્ધ મેથડ્સ ચેક કરો
print("upper" in dir(text))  # True

# મોડ્યુલ્સ માટે
import math
math_methods = dir(math)
print("sqrt" in math_methods)  # True

# કસ્ટમ ઓબ્જેક્ટ્સ માટે
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):
    """બે સંખ્યાઓ ઉમેરવા માટેનું ફંક્શન"""
    return a + b

def multiply_numbers(a, b):
    """બે સંખ્યાઓ ગુણવા માટેનું ફંક્શન"""
    return a * b

def get_sum(num1, num2):
    """વૈકલ્પિક સમ ફંક્શન"""
    result = num1 + num2
    return result
```

મુખ્ય પ્રોગ્રામ:

```python
# main.py
import calculator

# મોડ્યુલનો ઉપયોગ
result1 = calculator.add_numbers(10, 20)
print(f"સરવાળો: {result1}")

# ફ્રોમ ઇમ્પોર્ટ
from calculator import get_sum
result2 = get_sum(15, 25)
print(f"ફ્રોમ ઇમ્પોર્ટ વાપરીને સરવાળો: {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:

```python
# રનટાઇમ એરર
x = 10
y = 0
result = x / y  # ZeroDivisionError

# લોજિકલ એરર
def calculate_area(radius):
    return 3.14 * radius  # radius * radius હોવું જોઈએ
```

- **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("નંબર દાખલ કરો: "))
    result = 10 / number
except ValueError:
    print("અયોગ્ય ઇનપુટ")
except ZeroDivisionError:
    print("શૂન્ય વડે ભાગાકાર કરી શકાતો નથી")
except Exception as e:
    print(f"એરર: {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:
        # યુઝરથી ઇનપુટ લો
        numerator = float(input("અંશ દાખલ કરો: "))
        denominator = float(input("હર દાખલ કરો: "))

        # ભાગાકાર કરો
        result = numerator / denominator
        print(f"પરિણામ: {numerator} / {denominator} = {result}")
```

```
    except ZeroDivisionError:
        print("એરર: શૂન્ય વડે ભાગાકાર કરી શકાતો નથી!")
        print("કૃપા કરીને બિન-શૂન્ય હર દાખલ કરો")

    except ValueError:
        print("એરર: કૃપા કરીને માત્ર માન્ય નંબરો જ દાખલ કરો")

    except Exception as e:
        print(f"અનપેક્ષિત એરર આવી: {e}")

    finally:
        print("ભાગાકાર ઓપરેશન પૂર્ણ થયું")
        print("કેલ્ક્યુલેટર ઉપયોગ કરવા બદલ આભાર")

# ફંક્શનને કોલ કરો
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")

```
# ઉદાહરણો
try:
    int("hello")  # ValueError
    "5" + 5       # TypeError
    [1,2,3][5]    # IndexError
except (ValueError, TypeError, IndexError) as e:
    print(f"એરર: {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:

```
# ગુમ થયેલો કોલન
if x > 5
    print("વધારે")  # SyntaxError

# અમેળ કૌંસ
print("Hello"  # SyntaxError

# ખોટું indentation
def my_function():
print("Hello")  # IndentationError

# અયોગ્ય વેરિયેબલ નામ
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:
    # જોખમી કોડ
    pass
except SpecificException:
    # સ્પેસિફિક એરર હેન્ડલ કરો
    pass
except Exception as e:
    # અન્ય કોઈ એરર હેન્ડલ કરો
    pass
else:
    # exception ન હોય તો ચાલે
    pass
finally:
    # હંમેશા ચાલે
    pass
```

સંપૂર્ણ ઉદાહરણ:

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

    except FileNotFoundError:
        print(f"એરર: ફાઇલ '{filename}' ન મળી")

    except ValueError:
        print("એરર: ફાઇલમાં બિન-આંકડાકીય ડેટા છે")

    except ZeroDivisionError:
        print("એરર: ફાઇલમાં કોઈ નંબરો ન મળ્યા")

    except Exception as e:
        print(f"અનપેક્ષિત એરર: {e}")
```

```
    else:
        print("ફાઇલ સફળતાપૂર્વક પ્રોસેસ થઈ")

    finally:
        print("ફાઇલ પ્રોસેસિંગ ઓપરેશન પૂર્ણ થયું")

# ફંક્શન ચલાવો
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()

```
# ઉદાહરણ ઓપરેશન્સ
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:
        # ફાઇલનામ ઇનપુટ
        filename = input("ફાઇલનામ દાખલ કરો: ")

        # ફાઇલ કન્ટેન્ટ વાંચો
        with open(filename, 'r', encoding='utf-8') as file:
            content = file.read()

        # શબ્દોમાં વિભાજિત કરો અને સાફ કરો
        words = content.lower().split()
```

```
        # Punctuation રીમૂવ કરો અને ખાલી સ્ટ્રિંગ્સ
        import string
        clean_words = []
        for word in words:
            clean_word = word.translate(str.maketrans('', '', string.punctuation))
            if clean_word:  # માત્ર બિન-ખાલી શબ્દો ઉમેરો
                clean_words.append(clean_word)

        # શબ્દોને સૉર્ટ કરો
        sorted_words = sorted(clean_words)

        # પરિણામો દર્શાવો
        print("સૉર્ટ થયેલા શબ્દો:")
        print(sorted_words)

        # નવી ફાઇલમાં સેવ કરો
        with open('sorted_words.txt', 'w', encoding='utf-8') as output_file:
            for word in sorted_words:
                output_file.write(word + '\n')

        print(f"કુલ શબ્દો: {len(sorted_words)}")
        print("સૉર્ટ થયેલા શબ્દો 'sorted_words.txt' માં સેવ થયા")

    except FileNotFoundError:
        print("એરર: ફાઇલ ન મળી")
    except Exception as e:
        print(f"એરર: {e}")

# પ્રોગ્રામ ચલાવો
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

```
# બેસિક ફાઇલ ઓપરેશન્સ
file = open('data.txt', 'w')  # Open
file.write('Hello World')     # Write
file.close()                  # Close

file = open('data.txt', 'r')  # વાંચવા માટે ખોલો
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

# પ્રથમ, કંઈક ડેટા સેવ કરો
data_to_save = {
    'name': 'John',
    'age': 25,
    'scores': [85, 92, 78]
}

# ડેટાને ફાઇલમાં સેવ કરો
with open('data.pkl', 'wb') as file:
    pickle.dump(data_to_save, file)

# ફાઇલમાંથી ડેટા લોડ કરો
with open('data.pkl', 'rb') as file:
    loaded_data = pickle.load(file)
```

```
print("લોડ થયેલો ડેટા:", loaded_data)
print("નામ:", loaded_data['name'])
print("સ્કોર્સ:", loaded_data['scores'])
```

JSON load() ઉદાહરણ:

```
import json

# JSON ડેટા લોડ કરો
with open('config.json', 'r') as file:
    config = json.load(file)

print("કન્ફિગરેશન:", 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:
        # યુઝરથી ફાઇલનામ લો
        filename = input("ટેક્સ્ટ ફાઇલનામ દાખલ કરો: ")

        # ફાઇલ કન્ટેન્ટ વાંચો
        with open(filename, 'r', encoding='utf-8') as file:
            content = file.read().lower()

        # શબ્દો સાફ કરો અને એક્સ્ટ્રેક્ટ કરો
        import re
        import string

        # Punctuation રીમૂવ કરો અને શબ્દોમાં વિભાજિત કરો
        words = re.findall(r'\b[a-zA-Z]+\b', content.lower())

        # યુનીક શબ્દો મેળવવા માટે set બનાવો
        unique_words = set(words)

        # સૉર્ટ થયેલી લિસ્ટમાં કન્વર્ટ કરો
        sorted_unique_words = sorted(list(unique_words))
```

```
        # પરિણામો દર્શાવો
        print("\nમૂળાક્ષરોના ક્રમમાં યુનીક શબ્દો:")
        print("-" * 40)

        for i, word in enumerate(sorted_unique_words, 1):
            print(f"{i:3d}. {word}")

        print(f"\nકુલ યુનીક શબ્દો: {len(sorted_unique_words)}")

        # પરિણામો ફાઇલમાં સેવ કરો
        with open('unique_words_output.txt', 'w', encoding='utf-8') as output_file:
            output_file.write("મૂળાક્ષરોના ક્રમમાં યુનીક શબ્દો\n")
            output_file.write("=" * 40 + "\n\n")
            for word in sorted_unique_words:
                output_file.write(word + '\n')

        print("પરિણામો 'unique_words_output.txt' માં સેવ થયા")

    except FileNotFoundError:
        print(f"એરર: ફાઇલ '{filename}' ન મળી")
    except PermissionError:
        print("એરર: ફાઇલ વાંચવાની પરમિશન નકારાઈ")
    except Exception as e:
        print(f"અનપેક્ષિત એરર: {e}")

# ઉદાહરણ ઉપયોગ
def create_sample_file():
    sample_text = """
    Python એક શક્તિશાળી પ્રોગ્રામિંગ લેંગ્વેજ છે.
    Python શીખવામાં સરળ છે અને Python વર્સેટાઇલ છે.
    Python સાથે પ્રોગ્રામિંગ મજાદાર છે અને પ્રોગ્રામિંગ લાભદાયક છે.
    """

    with open('sample.txt', 'w', encoding='utf-8') as f:
        f.write(sample_text)
    print("નમૂનો ફાઇલ 'sample.txt' બનાવવામાં આવી")

# નમૂનો બનાવો અને પ્રોગ્રામ ચલાવો
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

# ટર્ટલ બનાવો
t = turtle.Turtle()

# ટર્ન ફંક્શન્સ
t.left(90)     # ડાબે 90 ડિગ્રી ફેરવો
t.right(45)    # જમણે 45 ડિગ્રી ફેરવો

# મૂવ ફંક્શન્સ
t.forward(100)  # આગળ 100 યુનિટ ખસો
t.backward(50)  # પાછળ 50 યુનિટ ખસો

# વિન્ડો ખુલ્લી રાખો
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()

# સંબંધિત ફેરવણું
t.left(90)         # ડાબે 90° ફેરવો
t.right(45)        # જમણે 45° ફેરવો

# ચોક્કસ દિશા
t.setheading(0)    # પૂર્વ તરફ નિર્દેશ કરો (0°)
t.setheading(90)   # ઉત્તર તરફ નિર્દેશ કરો (90°)

# ચોક્કસ પોઇન્ટ તરફ નિર્દેશ કરો
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():
    # ટર્ટલ અને સ્ક્રીન બનાવો
    screen = turtle.Screen()
    screen.title("ટર્ટલ સાથે આકારો દોરવા")
    screen.bgcolor("white")
    screen.setup(800, 600)

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

    # ચોરસ દોરો
    pen.penup()
    pen.goto(-200, 100)
    pen.pendown()
    pen.write("ચોરસ", font=("Arial", 12, "bold"))
    pen.goto(-200, 50)
```

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

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

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

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

    pen.circle(40)  # Radius = 40

    # ટર્ટલ છુપાવો અને વિન્ડો ખુલ્લી રાખો
    pen.hideturtle()
    screen.exitonclick()

# દરેક આકાર માટે વૈકલ્પિક ફંક્શન
def draw_square(turtle_obj, size):
    """આપેલા સાઇઝ સાથે ચોરસ દોરો"""
    for _ in range(4):
        turtle_obj.forward(size)
        turtle_obj.right(90)

def draw_rectangle(turtle_obj, width, height):
    """આપેલા પરિમાણો સાથે લંબચોરસ દોરો"""
    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):
    """આપેલા radius સાથે વર્તુળ દોરો"""
    turtle_obj.circle(radius)

# મુખ્ય પ્રોગ્રામ ચલાવો
```

```
draw_shapes()
```

- **ચોરસ**: 90° ફેરવણું સાથે 4 સમાન બાજુઓ
- **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()

# પેન કન્ટ્રોલ
t.penup()           # પેન ઉઠાવો
t.goto(50, 50)      # દોર્યા વિના ખસો
t.pendown()         # પેન નીચે મૂકો
t.pensize(3)        # જાડાઈ સેટ કરો
t.pencolor("red")   # રંગ સેટ કરો
```

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():
    # સ્ક્રીન સેટઅપ
    screen = turtle.Screen()
    screen.bgcolor("white")
    screen.title("ભરેલા વર્તુળ અને સ્ટાર")

    # ટર્ટલ બનાવો
    artist = turtle.Turtle()
    artist.speed(5)

    # ભરેલા વર્તુળ દોરો
    artist.penup()
    artist.goto(-150, 0)
    artist.pendown()

    # વર્તુળ માટે રંગો સેટ કરો
    artist.color("red", "red")  # pen color, fill color
    artist.begin_fill()
    artist.circle(50)
    artist.end_fill()

    # ભરેલા સ્ટાર દોરો
    artist.penup()
    artist.goto(100, 0)
    artist.pendown()

    # સ્ટાર માટે રંગો સેટ કરો
    artist.color("red", "red")
    artist.begin_fill()

    # 5-પોઇન્ટેડ સ્ટાર દોરો
    for i in range(5):
        artist.forward(100)
        artist.right(144)

    artist.end_fill()

    # લેબલ્સ ઉમેરો
    artist.penup()
    artist.goto(-180, -80)
    artist.color("black")
    artist.write("ભરેલા વર્તુળ", font=("Arial", 12, "bold"))

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

    # ટર્ટલ છુપાવો
```

```
    artist.hideturtle()
    screen.exitonclick()

# પ્રોગ્રામ ચલાવો
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():
    # સ્ક્રીન બનાવો
    screen = turtle.Screen()
    screen.bgcolor("white")
    screen.title("ભારતનો ઝંડો")
    screen.setup(800, 600)

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

    # ઝંડાના પરિમાણો
    flag_width = 300
    flag_height = 200

    # શરૂઆતની પોઝિશન
    start_x = -150
    start_y = 100

    # ઝંડાનો દંડ દોરો
    flag.penup()
    flag.goto(start_x - 20, start_y + 50)
    flag.pendown()
    flag.color("brown")
    flag.pensize(8)
```

```
    flag.setheading(270)  # નીચે તરફ નિર્દેશ કરો
    flag.forward(400)

    # પેન રીસેટ કરો
    flag.pensize(2)
    flag.color("black")

    # કેસરી લંબચોરસ (ઉપર)
    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()

    # સફેદ લંબચોરસ (મધ્ય)
    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()

    # લીલો લંબચોરસ (નીચે)
    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()

    # અશોક ચક્ર (ચક્ર) દોરો
    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)

    # બાહ્ય વર્તુળ દોરો
    flag.circle(30)

    # તીલીઓ દોરો
    flag.penup()
    flag.goto(chakra_center_x, chakra_center_y)
    flag.pendown()

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

    # અંદરનું વર્તુળ દોરો
    flag.penup()
    flag.goto(chakra_center_x, chakra_center_y - 5)
    flag.pendown()
    flag.circle(5)

    # શીર્ષક ઉમેરો
    flag.penup()
    flag.goto(-100, 200)
    flag.color("black")
    flag.write("ભારતનો ઝંડો", font=("Arial", 16, "bold"))

    # ટર્ટલ છુપાવો
    flag.hideturtle()
    screen.exitonclick()

# પ્રોગ્રામ ચલાવો
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"