Advanced Python Programming (4321602) - Winter 2024 Solution (Gujarati)

Milav Dabgar

January 18, 2025

પ્રશ્ન 1(અ) [3 ગુણ]

પાયથનમાં સેટ અને ડિક્શનરી વચ્ચેનો તફાવત લખો.

જવાબ

કોષ્ટક 1. સેટ વિરુદ્ધ ડિક્શનરી તુલના

લક્ષણ	સેટ	ડિક્શનરી
ડેટા સ્ટોરેજ	ફક્ત યુનિક એલિમેન્ટ્સ સ્ટોર કરે	કી-વેલ્યુ પેર સ્ટોર કરે
ક્રમ	અનઓર્ડર્ડ કલેક્શન	ઓર્ડર્ડ (Python 3.7+)
ડુપ્લિકેટ્સ	ડુપ્લિકેટ્સની મંજૂરી નથી	કીઝ યુનિક હોવી જોઈએ
એક્સેસ	ઈન્ડેક્સ દ્વારા એક્સેસ કરી શકાતું નથી	કીઝ દ્વારા વેલ્યુઝ એક્સેસ કરવા
સિન્ટેક્સ	{1, 2, 3}	{'key': 'value'}

- **સેટ**: યુનિક, અનઓર્ડર્ડ એલિમેન્ટ્સનો કલેક્શન
- **ડિક્શનરી**: યુનિક કીઝ સાથે કી-વેલ્યુ પેરનો કલેક્શન

મેમરી ટ્રીક

``Sets are Unique, Dicts have Keys''

પ્રશ્ન 1(બ) [4 ગુણ]

પાયથોનમાં લિસ્ટ ઉદાહરણ સાથે સમજાવો.

જવાબ

લિસ્ટ એક ઓર્ડર્ડ, મ્યુટેબલ કલેક્શન છે જે વિવિધ ડેટા ટાઈપ્સ સ્ટોર કરી શકે છે.

કોષ્ટક 2. લિસ્ટ ઓપરેશન્સ

ઓપરેશન	સિન્ટેક્સ	ઉદાહરણ
બનાવવું	list_name = []	fruits = ['apple', 'banana']
એક્સેસ	list[index]	fruits[0] રિટર્ન 'apple'
ઉમેરવું	append()	fruits.append('orange')
હટાવવું	remove()	fruits.remove('apple')

Listing 1. લિસ્ટ ઉદાહરણ

```python
# ઉદાહરણ
numbers = [1, 2, 3, 4, 5]
numbers.append(6)  # [1, 2, 3, 4, 5, 6]
print(numbers[0])  # આઉટપુટ: 1
```

- **ઓર્ડર્ડ**: એલિમેન્ટ્સ તેમની પોઝિશન જાળવે છે
- **મ્યુટેબલ**: બનાવ્યા પછી મોડિફાઈ કરી શકાય છે
- **ફ્લેક્સિબલ**: કોઈપણ ડેટા ટાઈપ સ્ટોર કરે છે

મેમરી ટ્રીક

``Lists are Ordered and Modifiable''

પ્રશ્ન 1(ક) [7 ગુણ]

પાયથોનમાં ટપલ શું છે? બે ટપલ વેલ્યુને અદલાબદલી કરવાનો પાયથન પ્રોગ્રામ લખો.

જવાબ

ટપલ એક ઓર્ડર્ડ, ઈમ્યુટેબલ કલેક્શન છે જે મલ્ટિપલ આઈટમ્સ સ્ટોર કરે છે.

કોષ્ટક 3. ટપલના લક્ષણો

પ્રોપર્ટી	વર્ણન	ઉદાહરણ
ઈમ્યુટેબલ	બનાવ્યા પછી બદલી શકાતું નથી	t = (1, 2, 3)
ઓર્ડર્ડ	એલિમેન્ટ્સનો નિર્ધારિત ક્રમ	ઈન્ડેક્સ દ્વારા એક્સેસ
ડુપ્લિકેટ્સ	ડુપ્લિકેટ વેલ્યુઝની મંજૂરી	(1, 1, 2)
ઈન્ડેક્સિંગ	પોઝિશન દ્વારા એલિમેન્ટ્સ એક્સેસ	t[0]

Listing 2. બે ટપલ વેલ્યુઝને સ્વેપ કરવાનો પ્રોગ્રામ

```python
# બે ટપલ વેલ્યુઝને સ્વેપ કરવાનો પ્રોગ્રામ
def swap_tuple_values(tup, pos1, pos2):
    # સ્વેપિંગ માટે ટપલને લિસ્ટમાં કન્વર્ટ કરો
    temp_list = list(tup)

    # વેલ્યુઝ સ્વેપ કરો
    temp_list[pos1], temp_list[pos2] = temp_list[pos2], temp_list[pos1]

    # પાછું ટપલમાં કન્વર્ટ કરો
    return tuple(temp_list)

# ઉદાહરણ ઉપયોગ
original_tuple = (10, 20, 30, 40, 50)
print("મૂળ ટપલ:", original_tuple)

# પોઝિશન 1 અને 3 પર વેલ્યુઝ સ્વેપ કરો
swapped_tuple = swap_tuple_values(original_tuple, 1, 3)
print("સ્વેપિંગ પછી:", swapped_tuple)
```

- **ઈમ્યુટેબલ**: એકવાર બનાવ્યા પછી મોડિફાઈ કરી શકાતું નથી
- **ઓર્ડર્ડ**: એલિમેન્ટ સિક્વન્સ જાળવે છે
- **હેટેરોજીનિયસ**: વિવિધ ડેટા ટાઈપ્સ સ્ટોર કરી શકે છે

મેમરી ટ્રીક

``Tuples are Immutable and Ordered''

પ્રશ્ન 1(ક OR) [7 ગુણ]

પાયથોનમાં ડિક્શનરી શું છે? લૂપની મદદથી ડિક્શનરીને ટ્રાવર્સ કરવાનો પાયથન પ્રોગ્રામ લખો.

જવાબ

ડિક્શનરી એક યુનિક કીઝ સાથે કી-વેલ્યુ પેરનો અનઓર્ડર્ડ કલેક્શન છે.

કોષ્ટક 4. ડિક્શનરી મેથડ્સ

મેથડ	હેતુ	ઉદાહરણ
keys()	બધી કીઝ મેળવો	dict.keys()
values()	બધી વેલ્યુઝ મેળવો	dict.values()
items()	કી-વેલ્યુ પેર મેળવો	dict.items()
get()	સેફ કી એક્સેસ	dict.get('key')

Listing 3. લૂપ વાપરીને ડિક્શનરી ટ્રાવર્સ કરવાનો પ્રોગ્રામ

```
# લૂપ વાપરીને ડક્ષ્ણિનરી ટ્રાવર્સ કરવાનો પ્રોગ્રામ
student_marks = {
    'Alice': 85,
    'Bob': 92,
    'Charlie': 78,
    'Diana': 96,
    'Eve': 89
}

print("ડક્ષ્ણિનરી ટ્રાવર્સલ મેથડ્સ:")
print("-" * 30)

# મેથડ 1: ફક્ત કીઝ ટ્રાવર્સ કરો
print("1. ફક્ત કીઝ:")
for key in student_marks:
    print(f"  {key}")

# મેથડ 2: ફક્ત વેલ્યુઝ ટ્રાવર્સ કરો
print("\n2. ફક્ત વેલ્યુઝ:")
for value in student_marks.values():
    print(f"  {value}")

# મેથડ 3: કીવેલ્યુ- પેર ટ્રાવર્સ કરો
print("\n3. કીવેલ્યુ- પેર:")
for key, value in student_marks.items():
    print(f"  {key}: {value}")

# મેથડ 4: keys() મેથડ વાપરીને
print("\n4. keys() મેથડ વાપરીને:")
for key in student_marks.keys():
    print(f"  {key} ને {student_marks[key]} માર્ક્સ મળ્યા")
```

- **કી-વેલ્યુ સ્ટોરેજ**: દરેક કી એક વેલ્યુ સાથે મેપ થાય છે
- **યુનિક કીઝ**: ડુપ્લિકેટ કીઝની મંજૂરી નથી
- **ફાસ્ટ લુકઅપ**: O(1) એવરેજ ટાઈમ કોમ્પ્લેક્સિટી

મેમરી ટ્રીક

``Dicts map Keys to Values''

પ્રશ્ન 2(અ) [3 ગુણ]

પેકેજ શું છે? પેકેજનો ઉપયોગ કરવાના ફાયદાઓની યાદી આપો.

જવાબ

પેકેજ એક ડિરેક્ટરી છે જેમાં મલ્ટિપલ મોડ્યુલ્સ એકસાથે ઓર્ગેનાઈઝ કરવામાં આવે છે.

કોષ્ટક 5. પેકેજના ફાયદાઓ

ફાયદો	વર્ણન
ઓર્ગેનાઈઝેશન	સંબંધિત મોડ્યુલ્સને એકસાથે ગ્રુપ કરે
નેમસ્પેસ	નામિંગ કોન્ફ્લિક્ટ્સ ટાળે
રીયુઝેબિલિટી	કોડ પ્રોજેક્ટ્સમાં ફરીથી વાપરી શકાય
મેઈન્ટેનેબિલિટી	મોટા કોડબેસ મેનેજ કરવું સરળ
ડિસ્ટ્રિબ્યુશન	શેર કરવું અને ઈન્સ્ટોલ કરવું સરળ

- **મોડ્યુલર સ્ટ્રક્ચર**: વધુ સારું કોડ ઓર્ગેનાઈઝેશન
- **હાયરાર્કિકલ નેમસ્પેસ**: નેમ કોન્ફ્લિક્ટ્સ અટકાવે
- **કોડ રીયુઝ**: સોફ્ટવેર રીયુઝેબિલિટીને પ્રમોટ કરે

મેમરી ટ્રીક

``Packages Organize Related Modules''

પ્રશ્ન 2(બ) [4 ગુણ]

કોઈપણ બે પેકેજ આયાત પદ્ધતિઓ ઉદાહરણો સાથે સમજાવો.

જવાબ

કોષ્ટક 6. આયાત મેથડ્સ

મેથડ	સિન્ટેક્સ	ઉપયોગ
નોર્મલ આયાત	import package.module	ફુલ પાથ સાથે એક્સેસ
ફ્રમ આયાત	from package import module	ડાયરેક્ટ મોડ્યુલ એક્સેસ
સ્પેસિફિક આયાત	from package.module import function	સ્પેસિફિક આઈટમ્સ આયાત
વાઈલ્ડકાર્ડ આયાત	from package import *	બધા મોડ્યુલ્સ આયાત

Listing 4. પેકેજ આયાત ઉદાહરણો

```
# મેથડ 1: નોર્મલ આયાત
import mypackage.calculator
result = mypackage.calculator.add(5, 3)
print(f"નોર્મલ આયાત પરિણામ: {result}")

# મેથડ 2: ફ્રમ આયાત
from mypackage import calculator
```

```
result = calculator.multiply(4, 6)
print(f"ફ્રમ આયાત પરિણા: {result}")
```

- **નોર્મલ આયાત**: ફુલ પેકેજ પાથ જરૂરી
- **ફ્રમ આયાત**: ડાયરેક્ટ મોડ્યુલ એક્સેસની મંજૂરી
- **સ્પેસિફિક ફંક્શન આયાત**: ફક્ત જરૂરી ફંક્શન્સ આયાત

મેમરી ટ્રીક

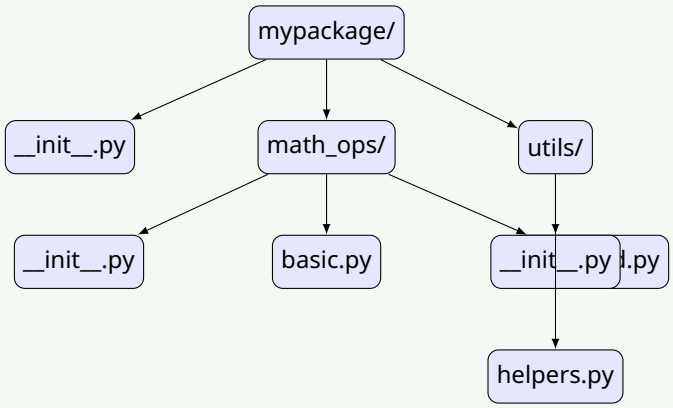
``Import Normally or From Package''

પ્રશ્ન 2(ક) [7 ગુણ]

ઈન્ટ્રા-પેકેજ સંદર્ભ વિશે ઉદાહરણ સાથે સમજાવો.

જવાબ

ઈન્ટ્રા-પેકેજ રેફરન્સ પેકેજની અંદરના મોડ્યુલ્સને એકબીજાથી આયાત કરવાની મંજૂરી આપે છે.
પેકેજ સ્ટ્રક્ચર ડાયાગ્રામ:



આકૃતિ 1. પેકેજ ડિરેક્ટરી સ્ટ્રક્ચર

કોષ્ટક 7. રેફરન્સ ટાઈપ્સ

ટાઈપ	સિન્ટેક્સ	ઉપયોગ
એબ્સોલ્યુટ	from mypackage.math_ops import basic	પેકેજ રૂટથી ફુલ પાથ
રિલેટિવ	from . import basic	વર્તમાન પેકેજ
પેરન્ટ	from .. import utils	પેરન્ટ પેકેજ
સિબલિંગ	from ..utils import helpers	સિબલિંગ પેકેજ

Listing 5. ઈન્ટ્રા-પેકેજ રેફરન્સ ઉદાહરણ

```
# પેકેજ સ્ટ્રક્ચર ઉદાહરણ
# mypackage/math_ops/advanced.py
from . import basic  # સેમ પેકેજથી રલિટવિ આયાત
from ..utils import helpers  # સબિલગિ પેકેજથી આયાત

def power_operation(base, exp):
    # બેસકિ મોડ્યુલથી ફંક્શન વાપરીને
    if basic.is_valid_number(base) and basic.is_valid_number(exp):
        result = base ** exp
        # હેલ્પર ફંક્શન વાપરીને
```

```
        return helpers.format_result(result)
    return None

# mypackage/math_ops/basic.py
def is_valid_number(num):
    return isinstance(num, (int, float))

def add(a, b):
    return a + b

# mypackage/utils/helpers.py
def format_result(value):
    return f"પરિણામ: {value:.2f}"
```

- **રિલેટિવ આયાત્સ**: વર્તમાન પેકેજ માટે ડોટ્સ (.) વાપરો
- **એબ્સોલ્યુટ આયાત્સ**: ફુલ પેકેજ પાથ
- **પેકેજ હાયરાર્કી**: ડોટ નોટેશન વાપરીને નેવિગેટ કરો

મેમરી ટ્રીક

``Dots Navigate Package Levels''

પ્રશ્ન 2(અ OR) [3 ગુણ]

મોડ્યુલ શું છે? મોડ્યુલનો ઉપયોગ કરવાના ફાયદાઓની યાદી આપો.

જવાબ

મોડ્યુલ એક Python ફાઈલ છે જેમાં ડેફિનિશન્સ, સ્ટેટમેન્ટ્સ અને ફંક્શન્સ હોય છે.

કોષ્ટક 8. મોડ્યુલના ફાયદાઓ

ફાયદો	વર્ણન
કોડ રીયુઝેબિલિટી	એકવાર લખો, અનેક વાર વાપરો
નેમસ્પેસ	ફંક્શન્સ માટે અલગ નેમસ્પેસ
ઓર્ગેનાઈઝેશન	વધુ સારું કોડ સ્ટ્રક્ચર
મેઈન્ટેનેબિલિટી	ડિબગ અને અપડેટ કરવું સરળ
કોલેબોરેશન	મલ્ટિપલ ડેવલપર્સ કામ કરી શકે

- **રીયુઝેબલ કોડ**: ફંક્શન્સ ગમે ત્યાં આયાત કરી શકાય
- **મોડ્યુલર ડિઝાઈન**: મોટા પ્રોગ્રામ્સને નાના ભાગોમાં વહેંચો
- **સરળ મેઈન્ટેનન્સ**: એક જગ્યાએ ફેરફાર બધી આયાત્સને અસર કરે

મેમરી ટ્રીક

``Modules Make Code Reusable''

પ્રશ્ન 2(બ OR) [4 ગુણ]

કોઈપણ બે મોડ્યુલ આયાત પદ્ધતિ ઉદાહરણ સાથે સમજાવો.

જવાબ

કોષ્ટક 9. મોડ્યુલ આયાત મેથડ્સ

મેથડ	સિન્ટેક્સ	એક્સેસ પેટર્ન
ડાયરેક્ટ આયાત	import module_name	module_name.function()
ફ્રમ આયાત	from module_name import function	function()
એલિયાસ આયાત	import module_name as alias	alias.function()
વાઈલ્ડકાર્ડ આયાત	from module_name import *	function()

Listing 6. મોડ્યુલ આયાત ઉદાહરણો

```
# મેથડ 1: ડાયરેક્ટ આયાત
import math
result1 = math.sqrt(16)
print(f"ડાયરેક્ટ આયાત: {result1}")

# મેથડ 2: ફ્રમ આયાત
from math import pi, sin
result2 = sin(pi/2)
print(f"ફ્રમ આયાત: {result2}")
```

- **ડાયરેક્ટ આયાત**: મોડ્યુલ નામ પ્રીફિક્સ સાથે એક્સેસ
- **ફ્રમ આયાત**: પ્રીફિક્સ વગર ડાયરેક્ટ ફંક્શન એક્સેસ
- **નેમસ્પેસ કંટ્રોલ**: યોગ્ય આયાત મેથડ પસંદ કરો

મેમરી ટ્રીક

``Import Directly or From Module''

પ્રશ્ન 2(ક OR) [7 ગુણ]

વતુળનું ક્ષેત્રફળ અને પરિઘ શોધવા માટેના મોડ્યુલનો પ્રોગ્રામ લખો.

જવાબ

Listing 7. વતુળ ઓપરેશન્સ મોડ્યુલ

```
# circle_operations.py મોડ્યુલ( ફાઈલ)
import math

def area(radius):
    """વતુળનું ક્ષેત્રફળ ગણો"""
    if radius <= 0:
        return 0
    return math.pi * radius * radius

def circumference(radius):
    """વતુળનો પરિઘ ગણો"""
    if radius <= 0:
        return 0
    return 2 * math.pi * radius

def display_info(radius):
    """વતુળની માહિતી દર્શાવો"""
    print(f"ત્રિજ્યા {radius} વાળું વતુળ:")
    print(f"ક્ષેત્રફળ: {area(radius):.2f}")
```

```
    print(f"પરિઘ: {circumference(radius):.2f}")

# કોન્સ્ટન્ટ્સ
PI = math.pi

# અ) મોડ્યુલને બીજા પ્રોગ્રામમાં આયાત કરો
# main_program.py
import circle_operations

radius = 5
print("મેથડ 1: સંપૂર્ણ મોડ્યુલ આયાત")
area_result = circle_operations.area(radius)
circumference_result = circle_operations.circumference(radius)

print(f"ક્ષેત્રફળ: {area_result:.2f}")
print(f"પરિઘ: {circumference_result:.2f}")

# બ) મોડ્યુલમાંથી ચોક્કસ ફંક્શન આયાત કરો
# specific_import.py
from circle_operations import area, circumference

radius = 7
print("\nમેથડn 2: ચોક્કસ ફંક્શન્સ આયાત")
area_result = area(radius)
circumference_result = circumference(radius)

print(f"ક્ષેત્રફળ: {area_result:.2f}")
print(f"પરિઘ: {circumference_result:.2f}")
```

કોષ્ટક 10. મોડ્યુલ ફીચર્સ

ફીચર	ઇમ્પ્લિમેન્ટેશન
ફંક્શન્સ	area(), circumference()
એરર હેન્ડલિંગ	નેગેટિવ ત્રિજ્યા માટે ચેક
કોન્સ્ટન્ટ્સ	PI વેલ્યુ
ડોક્યુમેન્ટેશન	ફંક્શન ડોકસ્ટ્રિંગ્સ

- **મોડ્યુલ બનાવવું**: ફંક્શન્સને .py ફાઈલમાં સેવ કરો
- **આયાત લવચીકતા**: સંપૂર્ણ મોડ્યુલ અથવા ચોક્કસ ફંક્શન્સ
- **કોડ રીયુઝ**: એક જ ફંક્શન્સ મલ્ટિપલ પ્રોગ્રામ્સમાં વાપરો

મેમરી ટ્રીક

``Modules Contain Reusable Functions''

પ્રશ્ન 3(અ) [3 ગુણ]

પાયથોનમાં ભૂલના પ્રકારો સમજાવો.

જવાબ

કોષ્ટક 11. Python એરર ટાઈપ્સ

એરર ટાઈપ	વર્ણન	ઉદાહરણ
સિન્ટેક્સ એરર	ખોટું Python સિન્ટેક્સ	કોલન : ભૂલી જવું
રનટાઈમ એરર	એક્ઝિક્યુશન દરમિયાન થાય	શૂન્યથી ભાગાકાર
લોજિકલ એરર	ખોટું પ્રોગ્રામ લોજિક	ખોટું અલ્ગોરિધમ
નેમ એરર	અંડિફાઈન્ડ વેરિએબલ	અઘોષિત વેરિએબલ વાપરવું
ટાઈપ એરર	ખોટું ડેટા ટાઈપ ઓપરેશન	સ્ટ્રિંગ + ઇન્ટિજર

- **સિન્ટેક્સ એરર**: પ્રોગ્રામ રન થાય તે પહેલાં શોધાય
- **રનટાઈમ એરર**: પ્રોગ્રામ એક્ઝિક્યુશન દરમિયાન થાય
- **લોજિકલ એરર**: પ્રોગ્રામ રન થાય પણ ખોટા પરિણામ આપે

મેમરી ટ્રીક

``Syntax, Runtime, Logic Errors''

પ્રશ્ન 3(બ) [4 ગુણ]

યુઝર-ડિફાઈન્ડ એક્સેપ્શન રેઈઝ સ્ટેટમેન્ટ સાથે સમજાવો.

જવાબ

યુઝર-ડિફાઈન્ડ એક્સેપ્શન્સ પ્રોગ્રામર્સ દ્વારા બનાવવામાં આવેલા કસ્ટમ એરર ક્લાસીસ છે.

કોષ્ટક 12. એક્સેપ્શન કોમ્પોનન્ટ્સ

કોમ્પોનન્ટ	હેતુ	ઉદાહરણ
ક્લાસ ડેફિનિશન	કસ્ટમ એક્સેપ્શન બનાવો	class CustomError(Exception):
રેઈઝ સ્ટેટમેન્ટ	એક્સેપ્શન ટ્રિગર કરો	raise CustomError("message")
એરર મેસેજ	સમસ્યાનું વર્ણન	માહિતીપ્રદ ટેક્સ્ટ
એક્સેપ્શન હેન્ડલિંગ	કસ્ટમ એક્સેપ્શન પકડો	except CustomError:

Listing 8. યુઝર-ડિફાઈન્ડ એક્સેપ્શન ઉદાહરણ

```
# કસ્ટમ એક્સેપ્શન ડેફિનિશન કરો
class AgeValidationError(Exception):
    def __init__(self, age, message="અયોગ્ય વય આપેલ છે"):
        self.age = age
        self.message = message
        super().__init__(self.message)

def validate_age(age):
    if age < 0:
        raise AgeValidationError(age, "વય નેગેટિવ હોઈ શકે નહીં")
    elif age > 150:
        raise AgeValidationError(age, "વય 150 કરતાં વધુ હોઈ શકે નહીં")
    else:
        print(f"યોગ્ય વય: {age}")

# કસ્ટમ એક્સેપ્શનનો ઉપયોગ
try:
    validate_age(-5)
except AgeValidationError as e:
    print(f"એરર: {e.message}, વય: {e.age}")
```

- **કસ્ટમ એક્સેપ્શન ક્લાસ**: Exception થી ઇન્હેરિટ કરે
- **રેઈઝ સ્ટેટમેન્ટ**: મેન્યુઅલી એક્સેપ્શન્સ ટ્રિગર કરે
- **અર્થપૂર્ણ મેસેજિસ**: ડિબગિંગમાં મદદ કરે

મેમરી ટ્રીક

``Raise Custom Exceptions for Validation''

પ્રશ્ન 3(ક) [7 ગુણ]

ટ્રાય-એક્સેપ્ટ-ફાઈનલી ક્લોઝ ઉદાહરણ સાથે સમજાવો.

જવાબ

ટ્રાય-એક્સેપ્ટ-ફાઈનલી સંપૂર્ણ એક્સેપ્શન હેન્ડલિંગ મિકેનિઝમ પૂરું પાડે છે.

કોષ્ટક 13. એક્સેપ્શન હેન્ડલિંગ બ્લોક્સ

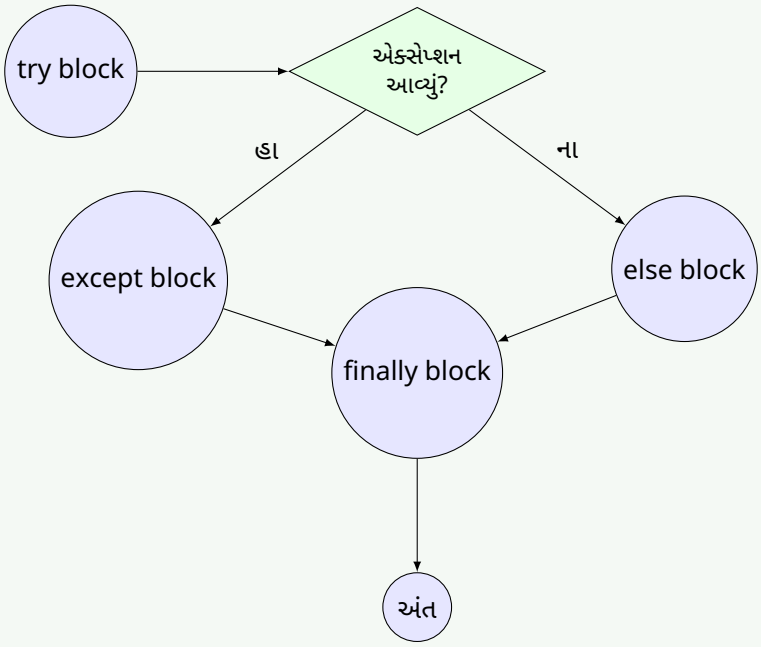
બ્લોક	હેતુ	એક્ઝિક્યુશન
try	એક્સેપ્શન ઉઠાવી શકે તેવો કોડ	હંમેશા પહેલા એક્ઝિક્યુટ
except	ચોક્કસ એક્સેપ્શન્સ હેન્ડલ કરે	ફક્ત એક્સેપ્શન આવે તો
else	કોઈ એક્સેપ્શન નહીં આવે ત્યારે	ફક્ત કોઈ એક્સેપ્શન નહીં આવે તો
finally	ક્લીનઅપ કોડ	હંમેશા એક્ઝિક્યુટ થાય

Listing 9. સંપૂર્ણ એક્સેપ્શન હેન્ડલિંગ ઉદાહરણ

```
# સંપૂર્ણ એક્સેપ્શન હેન્ડલિંગ ઉદાહરણ
def divide_numbers():
    try:
        print("ભાગાકાર ઓપરેશન શરૂ કરી રહ્યા છીએ...")

        # યુઝરથી ઇનપુટ મેળવો
        num1 = float(input("પહેલો નંબર દાખલ કરો: "))
        num2 = float(input("બીજો નંબર દાખલ કરો: "))

        # ભાગાકાર કરો
        result = num1 / num2

    except ValueError:
        print("એરર: કૃપા કરીને ફક્ત યોગ્ય નંબર્સ દાખલ કરો")
        return None

    except ZeroDivisionError:
        print("એરર: શૂન્યથી ભાગાકાર કરી શકાતો નથી")
        return None

    except Exception as e:
        print(f"અનપેક્ષિત એરર આવ્યું: {e}")
        return None

    else:
        print(f"ભાગાકાર સફળ: {num1} / {num2} = {result}")
        return result

    finally:
        print("ભાગાકાર ઓપરેશન પૂર્ણ")
```

```
        print("રિસોર્સિસ ક્લીન કરી રહ્યા છીએ...")

# ઉદાહરણ ઉપયોગ
result = divide_numbers()
if result:
    print(f"અંતિમ પરિણામ: {result}")
```

એક્સેપ્શન હેન્ડલિંગ ફ્લો:



આકૃતિ 2. Try-Except-Finally ફ્લો

- **try**: જોખમી કોડ હોય છે
- **except**: ચોક્કસ એરર્સ હેન્ડલ કરે
- **finally**: ક્લીનઅપ માટે હંમેશા એક્ઝિક્યુટ થાય

મેમરી ટ્રીક

``Try-Except-Finally Always Cleans''

પ્રશ્ન 3(અ OR) [3 ગુણ]

બિલ્ટ-ઇન એક્સેપ્શન શું છે? કોઇ પણ બેની તેમના અર્થ સાથે યાદી બનાવો.

જવાબ

બિલ્ટ-ઇન એક્સેપ્શન્સ Python માં પૂર્વ-નિર્ધારિત એરર ટાઈપ્સ છે.

કોષ્ટક 14. બિલ્ટ-ઇન એક્સેપ્શન્સ

એક્સેપ્શન	અર્થ	ઉદાહરણ
ValueError	યોગ્ય ટાઈપ પણ અમાન્ય વેલ્યુ	int("abc")
TypeError	ખોટું ડેટા ટાઈપ ઓપરેશન	"5" + 5
IndexError	લિસ્ટ ઇન્ડેક્સ રેન્જની બહાર	5-આઇટમ લિસ્ટ માટે list[10]
KeyError	ડિક્શનરી કી મળી નહીં	dict["missing_key"]
ZeroDivisionError	શૂન્યથી ભાગાકાર	10 / 0

બે મુખ્ય બિલ્ટ-ઇન એક્સેપ્શન્સ:
- **ValueError**: જ્યારે ફંક્શનને યોગ્ય ટાઈપ પણ અમાન્ય વેલ્યુ મળે
- **TypeError**: જ્યારે અયોગ્ય ડેટા ટાઈપ પર ઓપરેશન કરવામાં આવે

મેમરી ટ્રીક

``Built-in Exceptions Handle Common Errors''

પ્રશ્ન 3(બ OR) [4 ગુણ]

ટ્રાય-એક્સેપ્ટ ક્લોઝ ઉદાહરણ સાથે સમજાવો.

જવાબ

ટ્રાય-એક્સેપ્ટ પ્રોગ્રામ એક્ઝિક્યુશન દરમિયાન આવી શકે તેવા એક્સેપ્શન્સ હેન્ડલ કરે છે.

કોષ્ટક 15. એક્સેપ્શન હેન્ડલિંગ કોમ્પોનન્ટ્સ

કોમ્પોનન્ટ	હેતુ	સિન્ટેક્સ
try	નિષ્ફળ થઈ શકે તેવો કોડ	try:
except	ચોક્કસ એક્સેપ્શન હેન્ડલ કરે	except ErrorType:
મલ્ટિપલ except	વિવિધ એરર્સ હેન્ડલ કરે	મલ્ટિપલ except બ્લોક્સ
જનરલ except	કોઈપણ એક્સેપ્શન પકડે	except:

Listing 10. ટ્રાય-એક્સેપ્ટ ઉદાહરણ

```
# ટ્રાયએક્સેપ્ટ- ક્લોઝનું ઉદાહરણ
def safe_division():
    try:
        # એક્સેપ્શન્સ ઉઠાવી શકે તેવો કોડ
        dividend = int(input("ભાજ્ય દાખલ કરો: "))
        divisor = int(input("ભાજક દાખલ કરો: "))

        result = dividend / divisor
        print(f"પરિણામ: {dividend} / {divisor} = {result}")

    except ValueError:
        print("એરર: કૃપા કરીને ફક્ત યોગ્ય ઇન્ટજિર્સ દાખલ કરો")

    except ZeroDivisionError:
        print("એરર: શૂન્યથી ભાગાકાર કરી શકાતો નથી")

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

    print("એક્સેપ્શન હેન્ડલગિ પછી પ્રોગ્રામ ચાલુ રહે છે")

```

```
# ઉદાહરણ ઉપયોગ
safe_division()
```

- **try બ્લોક**: સંભવિત જોખમી કોડ હોય છે
- **except બ્લોક**: ચોક્કસ એક્સેપ્શન ટાઈપ્સ હેન્ડલ કરે
- **મલ્ટિપલ હેન્ડલર્સ**: વિવિધ એક્સેપ્શન્સ અલગ અલગ હેન્ડલ

મેમરી ટ્રીક

``Try Risky Code, Except Handles Errors''

પ્રશ્ન 3(ક OR) [7 ગુણ]

ડિવાઇડ બાય ઝીરો એક્સેપ્શન ફાઈનલી ક્લોઝ સાથે કેચ કરવાનો પ્રોગ્રામ લખો.

જવાબ

Listing 11. ફાઈનલી ક્લોઝ સાથે ડિવાઇડ બાય ઝીરો

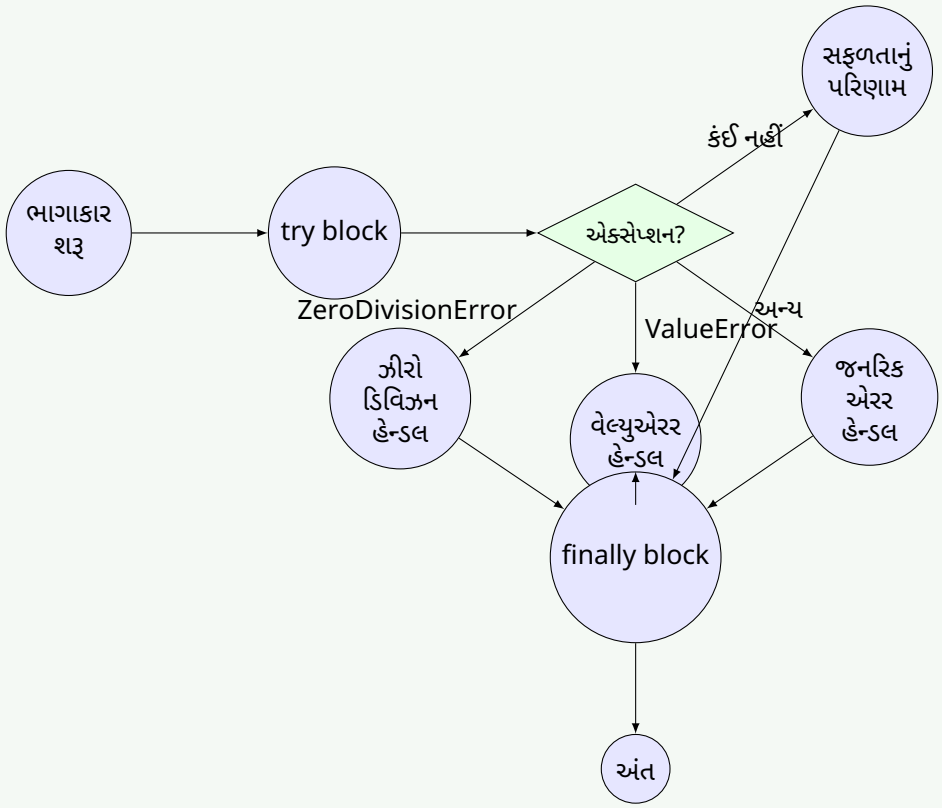
```
# ફાઈનલી ક્લોઝ સાથે ડવિાઇડ બાય ઝીરો હેન્ડલ કરવાનો પ્રોગ્રામ
def advanced_calculator():
    """વ્યાપક એક્સેપ્શન હેન્ડલગિ સાથે કેલ્ક્યુલેટર"""

    try:
        print("=== એડવાન્સ કેલ્ક્યુલેટર ===")
        print("ભાગાકાર માટે બે નંબર દાખલ કરો")

        # ઇનપુટ સેક્શન
        numerator = float(input("અંશ દાખલ કરો: "))
        denominator = float(input("છેદ દાખલ કરો: "))

        print(f"\n{numerator} ને {denominator} થી ભાગવાનો પ્રયાસ કરી રહ્યા છીએ...")

        # ક્રટિકિલ ઓપરેશન જે નષ્ફિળ થઈ શકે
        if denominator == 0:
            raise ZeroDivisionError("શૂન્યથી ભાગાકારની મંજૂરી નથી")

        result = numerator / denominator

        # સફળતાનો સંદેશ
        print(f"✅ ભાગાકાર સફળ!")
        print(f"📊 પરણિામ: {numerator} / {denominator} = {result:.4f}")

        return result

    except ZeroDivisionError as zde:
        print(f"❌ ઝીરો ડવિઝિન એરર: {zde}")
        print("💡 કૃપા કરીને શૂન્ય સવિાયનો છેદ વાપરો")
        return None

    except ValueError as ve:
        print(f"❌ વેલ્યુ એરર: અમાન્ય ઇનપુટ આપ્યું")
        print("💡 કૃપા કરીને ફક્ત નંબરકિ વેલ્યુઝ દાખલ કરો")
        return None

    except Exception as e:
        print(f"❌ અનપેક્ષતિ એરર: {e}")
        return None
```

```
    finally:
        print("\n" + "="*40)
        print("ક્લીનઅપ ઓપરેશન્સ:")
        print("- કેલ્ક્યુલેટર સેશન બંધ કરી રહ્યા છીએ")
        print("- ઓપરેશન લોગ સેવ કરી રહ્યા છીએ")
        print("- મેમરી રિસોર્સિસ છોડી રહ્યા છીએ")
        print("- કેલ્ક્યુલેટર શટડાઉન પૂર્ણ")
        print("="*40)

# કેલ્ક્યુલેટર ચલાવો
if __name__ == "__main__":
    result = advanced_calculator()

    if result is not None:
        print(f"\અંતમિn ગણેલું પરણિામ: {result}")
    else:
        print("\એરર્સનાn કારણે ગણતરી નષ્ફિળ")
```

કોષ્ટક 16. એક્સેપ્શન હેન્ડલિંગ ફીચર્સ

ફીચર	ઇમ્પ્લિમેન્ટેશન
ZeroDivisionError	શૂન્યથી ભાગાકાર માટે ચોક્કસ હેન્ડલિંગ
ValueError	અમાન્ય ઇનપુટ ટાઈપ્સ હેન્ડલ કરે
જનરિક એક્સેપ્શન	અનપેક્ષિત એરર્સ પકડે
ફાઈનલી બ્લોક	હંમેશા ક્લીનઅપ કોડ એક્ઝિક્યુટ

એક્સેપ્શન હેન્ડલિંગ ફ્લો:



આકૃતિ 3. ડિવાઇડ બાય ઝીરો એક્સેપ્શન હેન્ડલિંગ ફ્લો

- **ચોક્કસ એક્સેપ્શન હેન્ડલિંગ**: ZeroDivisionError અલગથી પકડાય

- **ફાઈનલી ક્લોઝ**: ક્લીનઅપ માટે હંમેશા એક્ઝિક્યુટ થાય
- **રિસોર્સ મેનેજમેન્ટ**: એરર્સ છતાં યોગ્ય ક્લીનઅપ

મેમરી ટ્રીક

``Finally Always Cleans Up Resources''

પ્રશ્ન 4(અ) [3 ગુણ]

વ્યાખ્યાયિત કરો: ફાઇલ, બાઇનરી ફાઇલ, ટેક્સ્ટ ફાઇલ

જવાબ

કોષ્ટક 17. ફાઈલ વ્યાખ્યાઓ

શબ્દ	વ્યાખ્યા	ઉદાહરણ
ફાઈલ	ડિસ્ક પર નામવાળું સ્ટોરેજ સ્થાન	document.txt, image.jpg
બાઈનરી ફાઈલ	બાઈનરી ફોર્મેટમાં નોન-ટેક્સ્ટ ડેટા સમાવે	.exe, .jpg, .mp3, .pdf
ટેક્સ્ટ ફાઈલ	માનવ-વાંચી શકાય તેવા ટેક્સ્ટ કેરેક્ટર્સ સમાવે	.txt, .py, .html, .csv

વિગતવાર વ્યાખ્યાઓ:
- **ફાઈલ**: સ્ટોરેજ ડિવાઈસ પર યુનિક નામ સાથે સ્ટોર કરેલો ડેટાનો કલેક્શન
- **બાઈનરી ફાઈલ**: બાઈનરી ફોર્મેટ (0s અને 1s) માં ડેટા સ્ટોર કરે, માનવ-વાંચી ન શકાય
- **ટેક્સ્ટ ફાઈલ**: ASCII અથવા Unicode કેરેક્ટર્સ સમાવે, માનવ-વાંચી શકાય તેવું ફોર્મેટ

મેમરી ટ્રીક

``Files store data, Binary=Machine, Text=Human''

પ્રશ્ન 4(બ) [4 ગુણ]

write() અને writelines() ફંક્શન ઉદાહરણ સાથે સમજાવો.

જવાબ

કોષ્ટક 18. રાઈટ ફંક્શન્સ

ફંક્શન	હેતુ	પેરામીટર	ઉપયોગ
write()	સિંગલ સ્ટ્રિંગ લખે	સ્ટ્રિંગ	file.write("Hello")
writelines()	સ્ટ્રિંગ્સની લિસ્ટ લખે	લિસ્ટ/સિક્વન્સ	file.writelines(["line1", "line2"])

Listing 12. રાઈટ ફંક્શન્સનું ઉદાહરણ

```
# write() અને writelines() ફંક્શન્સનું ઉદાહરણ
def demonstrate_write_functions():

    # write() ફંક્શન વાપરીને
    with open("write_demo.txt", "w") as file:
        file.write("નમસ્તે વિશ્વ!\n")
        file.write("આ બીજી લાઈન છે\n")
        file.write("આ ત્રીજી લાઈન છે\n")

```

```
    # writelines() ફંક્શન વાપરીને
    lines = [
        "writelines વાપરીને પહેલી લાઈન\n",
        "writelines વાપરીને બીજી લાઈન\n",
        "writelines વાપરીને ત્રીજી લાઈન\n"
    ]

    with open("writelines_demo.txt", "w") as file:
        file.writelines(lines)

    print("ફાઈલો સફળતાપૂર્વક બનાવી!")

# ડેમોન્સ્ટ્રેશન ચલાવો
demonstrate_write_functions()
```

મુખ્ય તફાવતો:
- **write()**: એક વખતે એક સ્ટ્રિંગ લખે
- **writelines()**: સિક્વન્સમાંથી મલ્ટિપલ સ્ટ્રિંગ્સ લખે
- **ન્યૂલાઈન્સ**: \n સાથે મેન્યુઅલી ઉમેરવી જોઈએ
- **રિટર્ન વેલ્યુ**: બંને લખેલા કેરેક્ટર્સની સંખ્યા રિટર્ન કરે

મેમરી ટ્રીક

``write() Single, writelines() Multiple''

પ્રશ્ન 4(ક) [7 ગુણ]

tell() અને seek() ફંક્શન ઉદાહરણ સાથે સમજાવો.

જવાબ

ફાઈલ પોઈન્ટર ફંક્શન્સ ફાઈલની અંદર રીડિંગ/રાઈટિંગ માટે પોઝિશન કંટ્રોલ કરે છે.

કોષ્ટક 19. પોઝિશન ફંક્શન્સ

ફંક્શન	હેતુ	રિટર્ન/પેરામીટર	ઉપયોગ
tell()	વર્તમાન પોઝિશન મેળવો	વર્તમાન બાઈટ પોઝિશન રિટર્ન	pos = file.tell()
seek(offset, whence)	ચોક્કસ પોઝિશન પર જાઓ	offset: બાઈટ્સ, whence: રેફરન્સ	file.seek(10, 0)

કોષ્ટક 20. Seek Whence વેલ્યુઝ

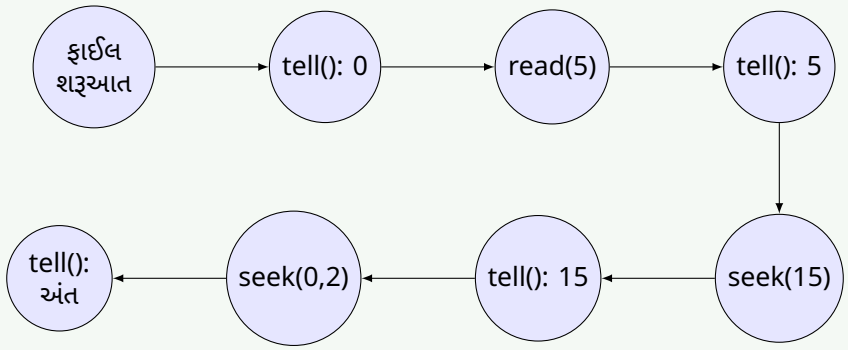
વેલ્યુ	રેફરન્સ પોઈન્ટ	વર્ણન
0	ફાઈલની શરૂઆત	એબ્સોલ્યુટ પોઝિશનિંગ
1	વર્તમાન પોઝિશન	વર્તમાનના સંબંધમાં
2	ફાઈલનો અંત	અંતના સંબંધમાં

Listing 13. tell() અને seek() ઉદાહરણ

```
# tell() અને seek() ફંક્શન્સનું સંપૂર્ણ ઉદાહરણ
def demonstrate_file_positioning():

    # સેમ્પલ ફાઈલ બનાવો
    sample_text = "Hello World! This is a sample file for demonstrating tell() and seek() functions."

```

```
    with open("position_demo.txt", "w") as file:
        file.write(sample_text)

    # tell() અને seek() દર્શાવો
    with open("position_demo.txt", "r") as file:

        # શરૂઆતની પોઝિશન
        print(f"1. શરૂઆતની પોઝિશન: {file.tell()}")

        # પહેલા 5 કેરેક્ટર્સ વાંચો
        data1 = file.read(5)
        print(f"2. '{data1}' વાંચ્યું, વર્તમાન પોઝિશન: {file.tell()}")

        # પોઝિશન 15 પર જાઓ
        file.seek(15)
        print(f"3. seek(15) પછી, પોઝિશન: {file.tell()}")

        # આગળના 10 કેરેક્ટર્સ વાંચો
        data2 = file.read(10)
        print(f"4. '{data2}' વાંચ્યું, વર્તમાન પોઝિશન: {file.tell()}")

        # seek(0, 0) વાપરીને શરૂઆતમાં જાઓ
        file.seek(0, 0)
        print(f"5. seek(0,0) પછી, પોઝિશન: {file.tell()}")

        # seek(0, 2) વાપરીને અંતમાં જાઓ
        file.seek(0, 2)
        print(f"6. seek(0,2) પછી, પોઝિશન: {file.tell()}")

        # વર્તમાન પોઝિશનથી પાછળ જાઓ
        file.seek(-10, 1)
        print(f"7. seek(-10,1) પછી, પોઝિશન: {file.tell()}")

        # બાકીનું કન્ટેન્ટ વાંચો
        remaining = file.read()
        print(f"8. બાકીનું કન્ટેન્ટ: '{remaining}'")

# ડેમોન્સ્ટ્રેશન ચલાવો
demonstrate_file_positioning()
```

પોઝિશન કંટ્રોલ ફ્લો:



આકૃતિ 4. ફાઈલ પોઝિશન કંટ્રોલ ફ્લો

- **tell()**: ફાઈલમાં વર્તમાન બાઈટ પોઝિશન રિટર્ન કરે
- **seek()**: ફાઈલ પોઈન્ટરને સ્પેસિફાઈડ પોઝિશન પર મૂવ કરે
- **પોઝિશનિંગ**: રેન્ડમ ફાઈલ એક્સેસ માટે જરૂરી
- **બાઈનરી મોડ**: બાઈટ પોઝિશન્સ સાથે કામ કરે

મેમરી ટ્રીક

``tell() Position, seek() Movement''

પ્રશ્ન 4(અ OR) [3 ગુણ]

એબ્સોલ્યુટ અને રિલેટિવ ફાઈલ પાથ શું છે?

જવાબ

કોષ્ટક 21. પાથ ટાઈપ્સ

પાથ ટાઈપ	વર્ણન	ઉદાહરણ
એબ્સોલ્યુટ પાથ	રૂટ ડિરેક્ટરીથી સંપૂર્ણ પાથ	/home/user/documents/file.txt
રિલેટિવ પાથ	વર્તમાન ડિરેક્ટરીના સંબંધમાં પાથ	../documents/file.txt

કોષ્ટક 22. પાથ સિમ્બોલ્સ

સિમ્બોલ	અર્થ	ઉદાહરણ
/	રૂટ ડિરેક્ટરી (Linux/Mac)	/home/user/
C:\	ડ્રાઈવ લેટર (Windows)	C:\Users\Documents\
.	વર્તમાન ડિરેક્ટરી	./file.txt
..	પેરન્ટ ડિરેક્ટરી	../folder/file.txt

- **એબ્સોલ્યુટ**: સિસ્ટમ રૂટથી સંપૂર્ણ પાથ
- **રિલેટિવ**: વર્તમાન વર્કિંગ ડિરેક્ટરીથી પાથ

મેમરી ટ્રીક

``Absolute from Root, Relative from Current''

પ્રશ્ન 4(બ OR) [4 ગુણ]

બાયનરી અને ટેક્સ્ટ ફાઈલ ખોલવાના વિવિધ મોડ સમજાવો.

જવાબ

કોષ્ટક 23. ફાઈલ ઓપનિંગ મોડ્સ

મોડ	ટાઈપ	હેતુ	ફાઈલ પોઈન્ટર
'r'	ટેક્સ્ટ	ફક્ત વાંચવા	શરૂઆત
'w'	ટેક્સ્ટ	લખવા (ઓવરરાઈટ)	શરૂઆત
'a'	ટેક્સ્ટ	ઉમેરવા	અંત
'rb'	બાઈનરી	બાઈનરી વાંચવા	શરૂઆત
'wb'	બાઈનરી	બાઈનરી લખવા	શરૂઆત
'ab'	બાઈનરી	બાઈનરી ઉમેરવા	અંત
'r+'	ટેક્સ્ટ	વાંચવા અને લખવા	શરૂઆત
'w+'	ટેક્સ્ટ	લખવા અને વાંચવા	શરૂઆત

Listing 14. ફાઈલ મોડ્સનું ઉદાહરણ

```python
# વિવિધ ફાઈલ મોડ્સના ઉદાહરણો
def demonstrate_file_modes():

    # ટેક્સ્ટ ફાઈલ મોડ્સ
    with open("text_file.txt", "w") as f:  # રાઈટ મોડ
        f.write("નમસ્તે વિશ્વ")

    with open("text_file.txt", "r") as f:  # રીડ મોડ
        content = f.read()
        print(f"ટેક્સ્ટ કન્ટેન્ટ: {content}")

    # બાઈનરી ફાઈલ મોડ્સ
    data = b"Binary data example"
    with open("binary_file.bin", "wb") as f:  # બાઈનરી રાઈટ
        f.write(data)

    with open("binary_file.bin", "rb") as f:  # બાઈનરી રીડ
        binary_content = f.read()
        print(f"બાઈનરી કન્ટેન્ટ: {binary_content}")

demonstrate_file_modes()
```

- **ટેક્સ્ટ મોડ્સ**: એન્કોડિંગ સાથે સ્ટ્રિંગ ડેટા હેન્ડલ કરે
- **બાઈનરી મોડ્સ**: એન્કોડિંગ વગર રો બાઈટ્સ હેન્ડલ કરે
- **પ્લસ મોડ્સ**: રીડિંગ અને રાઈટિંગ બંનેની મંજૂરી

મેમરી ટ્રીક

``Text for Strings, Binary for Bytes''

પ્રશ્ન 4(ક OR) [7 ગુણ]

વિદ્યાર્થીના વિષયનો રેકોર્ડ જેમ કે બ્રાન્ચ નામ, સેમેસ્ટર, વિષય કોડ અને વિષય નામ બાઈનરી ફાઇલમાં લખવાનો પાયથન પ્રોગ્રામ લખો.

જવાબ

Listing 15. બાઈનરી ફાઈલ વિદ્યાર્થી રેકોર્ડ્સ

```python
import pickle
import os

class StudentSubjectRecord:
    """વિદ્યાર્થી વિષય રેકોર્ડ્સ હેન્ડલ કરવા માટેનો ક્લાસ"""

    def __init__(self, branch_name, semester, subject_code, subject_name):
        self.branch_name = branch_name
        self.semester = semester
        self.subject_code = subject_code
        self.subject_name = subject_name

    def __str__(self):
        return f"Branch: {self.branch_name}, Semester: {self.semester}, Code: {self.subject_code}, Subject: {self.subject_name}"

def write_student_records():
    """બાઈનરી ફાઈલમાં વિદ્યાર્થી રેકોર્ડ્સ લખો"""

```

```
    # સેમ્પલ વદ્યિાર્થી રેકોર્ડ્સ
    records = [
        StudentSubjectRecord("Information Technology", 2, "4321602", "Advanced Python Programming"),
        StudentSubjectRecord("Information Technology", 2, "4321601", "Database Management System"),
        StudentSubjectRecord("Computer Engineering", 3, "4330701", "Data Structure"),
        StudentSubjectRecord("Information Technology", 2, "4321603", "Web Development"),
        StudentSubjectRecord("Computer Engineering", 3, "4330702", "Computer Networks")
    ]

    # pickle વાપરીને બાઈનરી ફાઈલમાં રેકોર્ડ્સ લખો
    try:
        with open("student_records.bin", "wb") as binary_file:
            pickle.dump(records, binary_file)

        print("✅ વદ્યિાર્થી રેકોર્ડ્સ બાઈનરી ફાઈલમાં સફળતાપૂર્વક લખાયા!")
        print(f"📊 કુલ લખાયેલા રેકોર્ડ્સ: {len(records)}")

    except Exception as e:
        print(f"❌ બાઈનરી ફાઈલમાં લખતી વખતે એરર: {e}")

def read_student_records():
    """બાઈનરી ફાઈલમાંથી વદ્યિાર્થી રેકોર્ડ્સ વાંચો"""

    try:
        if not os.path.exists("student_records.bin"):
            print("❌ બાઈનરી ફાઈલ મળી નહી!")
            return

        with open("student_records.bin", "rb") as binary_file:
            records = pickle.load(binary_file)

        print("\n" + "="*60)
        print("બાઈનરી ફાઈલમાંથી વદ્યિાર્થી વષિય રેકોર્ડ્સ")
        print("="*60)

        for i, record in enumerate(records, 1):
            print(f"{i}. {record}")

        print("="*60)
        print(f"કુલ વાંચેલા રેકોર્ડ્સ: {len(records)}")

    except Exception as e:
        print(f"❌ બાઈનરી ફાઈલમાંથી વાંચતી વખતે એરર: {e}")

# મુખ્ય પ્રોગ્રામ એક્ઝક્યિુશન
def main():
    """બાઈનરી ફાઈલ ઓપરેશન્સ દર્શાવવા માટેનો મુખ્ય ફંક્શન"""

    print("=== વદ્યિાર્થી વષિય રેકોર્ડ મેનેજમેન્ટ ===\n")

    # શરૂઆતના રેકોર્ડ્સ લખો
    print("1. બાઈનરી ફાઈલમાં વદ્યિાર્થી રેકોર્ડ્સ લખી રહ્યા છીએ...")
    write_student_records()

    # રેકોર્ડ્સ વાંચો અને દર્શાવો
    print("\n2. બાઈનરી ફાઈલમાંથી રેકોર્ડ્સ વાંચી રહ્યા છીએ...")
    read_student_records()

# પ્રોગ્રામ એક્ઝક્યિુટ કરો
if __name__ == "__main__":
    main()
```

કોષ્ટક 24. બાઈનરી ફાઈલ ઓપરેશન્સ

ઓપરેશન	મેથડ	હેતુ
લખવું	pickle.dump()	બાઈનરીમાં ઓબ્જેક્ટ્સ સીરિયલાઈઝ કરો
વાંચવું	pickle.load()	બાઈનરીમાંથી ઓબ્જેક્ટ્સ ડીસીરિયલાઈઝ કરો
ઉમેરવું	વાંચો + ઉમેરો + લખો	નવા રેકોર્ડ્સ ઉમેરો
શોધવું	લોડ કરેલા ડેટા ફિલ્ટર કરો	ચોક્કસ રેકોર્ડ્સ શોધો

બાઈનરી ફાઈલ ડેટા ફ્લો:

આકૃતિ 5. બાઈનરી ફાઈલ સીરિયલાઈઝેશન ફ્લો

- **બાઈનરી સ્ટોરેજ**: ઓબ્જેક્ટ સીરિયલાઈઝેશન માટે pickle વાપરે
- **કાર્યક્ષમ સ્ટોરેજ**: કોમ્પેક્ટ બાઈનરી ફોર્મેટ
- **ઓબ્જેક્ટ પ્રિઝર્વેશન**: ડેટા સ્ટ્રક્ચર ઇન્ટિગ્રિટી જાળવે
- **ક્રોસ-પ્લેટફોર્મ**: વિવિધ ઓપરેટિંગ સિસ્ટમ્સ પર કામ કરે

મેમરી ટ્રીક

``Pickle Preserves Python Objects''

પ્રશ્ન 5(અ) [3 ગુણ]

વ્યાખ્યાયિત કરો: GUI, CLI

જવાબ

કોષ્ટક 25. ઈન્ટરફેસ વ્યાખ્યાઓ

શબ્દ	ફુલ ફોર્મ	વર્ણન	ઉદાહરણ
GUI	Graphical User Interface	વિન્ડોઝ, બટનો, આઇકોન્સ સાથે વિઝ્યુઅલ ઈન્ટરફેસ	Windows, Mac desktop
CLI	Command Line Interface	કમાન્ડ્સ વાપરતું ટેક્સ્ટ-બેઝ્ડ ઈન્ટરફેસ	Terminal, Command Prompt

મુખ્ય તફાવતો:
- **GUI**: યુઝર-ફ્રેન્ડલી, માઉસ-ડ્રિવન, વિઝ્યુઅલ એલિમેન્ટ્સ
- **CLI**: ટેક્સ્ટ-બેઝ્ડ, કીબોર્ડ-ડ્રિવન, કમાન્ડ સિન્ટેક્સ
- **ઈન્ટરએક્શન**: GUI ક્લિક્સ વાપરે, CLI ટાઈપ કરેલા કમાન્ડ્સ વાપરે

મેમરી ટ્રીક

``GUI Graphics, CLI Commands''

પ્રશ્ન 5(બ) [4 ગુણ]

Turtle વાપરીને for અને while લૂપ વડે ચોરસ આકાર દોરવાનો પાયથન પ્રોગ્રામ લખો.

જવાબ

Listing 16. લૂપ્સ સાથે ચોરસ દોરવું

```
import turtle

def draw_square_with_for_loop():
    """for લૂપ વાપરીને ચોરસ દોરો"""

    # turtle સેટઅપ
    screen = turtle.Screen()
    screen.bgcolor("white")
    square_turtle = turtle.Turtle()
    square_turtle.color("blue")
    square_turtle.pensize(3)

    # for લૂપ વાપરીને ચોરસ દોરો
    print("for લૂપ વડે ચોરસ દોરી રહ્યા છીએ...")
    side_length = 100

    for i in range(4):
        square_turtle.forward(side_length)
        square_turtle.right(90)

    square_turtle.penup()
    square_turtle.goto(150, 0)
    square_turtle.pendown()

    return square_turtle

def draw_square_with_while_loop(turtle_obj):
    """while લૂપ વાપરીને ચોરસ દોરો"""

    # બીજા ચોરસ માટે રંગ બદલો
    turtle_obj.color("red")

    # while લૂપ વાપરીને ચોરસ દોરો
    print("while લૂપ વડે ચોરસ દોરી રહ્યા છીએ...")
    side_length = 100
    sides_drawn = 0

    while sides_drawn < 4:
        turtle_obj.forward(side_length)
        turtle_obj.right(90)
        sides_drawn += 1

    # ટેક્સ્ટ માટે turtle ને સેન્ટરમાં મૂવ કરો
    turtle_obj.penup()
    turtle_obj.goto(-50, -150)
    turtle_obj.write("Blue: for loop, Red: while loop",
                     font=("Arial", 12, "normal"))

# મુખ્ય એક્ઝિક્યુશન
def main():
    # ચોરસ દોરો
    turtle_obj = draw_square_with_for_loop()
    draw_square_with_while_loop(turtle_obj)

    # વિન્ડો ખુલ્લી રાખો
    turtle.Screen().exitonclick()

# પ્રોગ્રામ ચલાવો
main()
```

કોષ્ટક 26. લૂપ તુલના

લૂપ ટાઈપ	સ્ટ્રક્ચર	ઉપયોગ	કંટ્રોલ
for લૂપ	for i in range(4):	જાણીતા આઈટરેશન્સ	કાઉન્ટર-બેઝ્ડ
while લૂપ	while condition:	કન્ડિશનલ આઈટરેશન્સ	કન્ડિશન-બે ઝ્ડ

- **for લૂપ**: જાણીતી સંખ્યાના આઈટરેશન્સ માટે શ્રેષ્ઠ
- **while લૂપ**: કન્ડિશન-બેઝ્ડ રિપીટીશન માટે શ્રેષ્ઠ
- **બંને પ્રાપ્ત કરે**: સમાન ચોરસ દોરવાનું પરિણામ

મેમરી ટ્રીક

``For Count, While Condition''

પ્રશ્ન 5(ક) [7 ગુણ]

Turtle વાપરીને ચેસબોર્ડ દોરવાનો પાયથન પ્રોગ્રામ લખો.

જવાબ

Listing 17. ચેસબોર્ડ દોરવાનો પ્રોગ્રામ

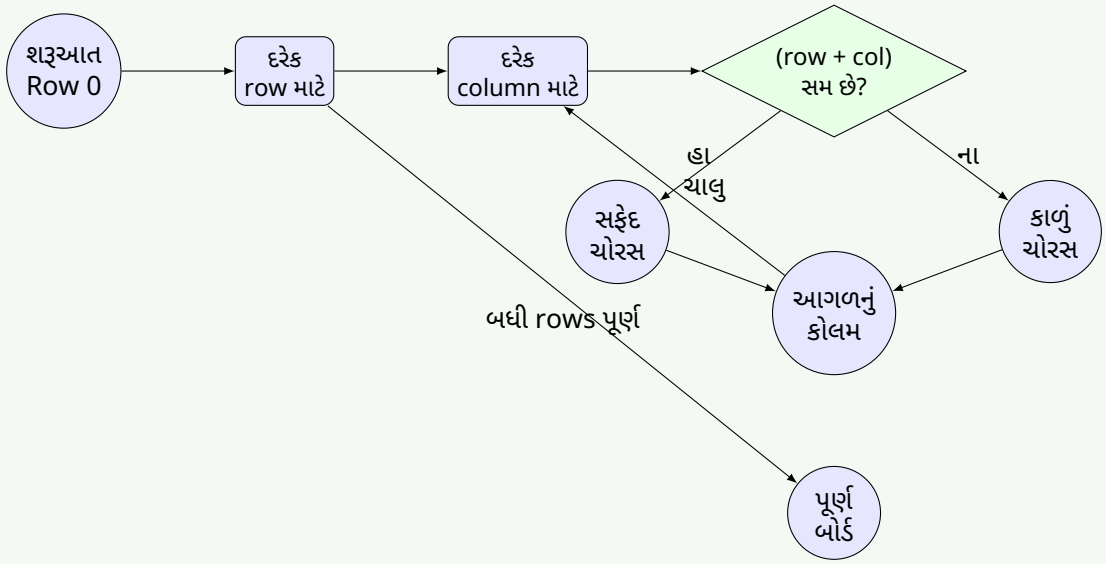
```
import turtle

def setup_chessboard():
    """ચેસબોર્ડ માટે turtle સ્ક્રીન અને પ્રોપર્ટીઝ સેટઅપ કરો"""

    screen = turtle.Screen()
    screen.bgcolor("white")
    screen.title("Python Turtle વાપરીને ચેસબોર્ડ")
    screen.setup(width=600, height=600)

    # દોરવા માટે turtle બનાવો
    chess_turtle = turtle.Turtle()
    chess_turtle.speed(0)  # સૌથી ઝડપી સ્પીડ
    chess_turtle.penup()

    return screen, chess_turtle

def draw_square(turtle_obj, size, fill_color):
    """આપેલા રંગ સાથે સિંગલ ચોરસ દોરો"""

    turtle_obj.pendown()
    turtle_obj.fillcolor(fill_color)
    turtle_obj.begin_fill()

    # ચોરસ દોરો
    for _ in range(4):
        turtle_obj.forward(size)
        turtle_obj.right(90)

    turtle_obj.end_fill()
    turtle_obj.penup()

def draw_chessboard():
```

```
    """સંપૂર્ણ 8x8 ચેસબોર્ડ દોરો"""

    screen, chess_turtle = setup_chessboard()

    # ચેસબોર્ડ પેરામીટર્સ
    square_size = 40
    board_size = 8
    start_x = -160
    start_y = 160

    print("ચેસબોર્ડ દોરી રહ્યા છીએ...")

    # બોર્ડ દોરો
    for row in range(board_size):
        for col in range(board_size):

            # પોઝિશન ગણો
            x = start_x + (col * square_size)
            y = start_y - (row * square_size)

            # turtle ને પોઝિશન પર મૂવ કરો
            chess_turtle.goto(x, y)

            # ચોરસનો રંગ નક્કી કરો અલ્ટરનેટિંગ( પેટર્ન)
            if (row + col) % 2 == 0:
                color = "white"
            else:
                color = "black"

            # ચોરસ દોરો
            draw_square(chess_turtle, square_size, color)

    # શીર્ષક ઉમેરો
    chess_turtle.goto(0, start_y + 30)
    chess_turtle.write("Python Turtle ચેસબોર્ડ", align="center",
                       font=("Arial", 16, "bold"))

    print("ચેસબોર્ડ સફળતાપૂર્વક બન્યું!")
    return screen

# મુખ્ય એક્ઝિક્યુશન
def main():
    """ચેસબોર્ડ બનાવવા માટેનો મુખ્ય ફંક્શન"""

    screen = draw_chessboard()
    print("વિન્ડો બંધ કરવા માટે સ્ક્રીન પર ક્લિક કરો.")
    screen.exitonclick()

# પ્રોગ્રામ ચલાવો
if __name__ == "__main__":
    main()
```

કોષ્ટક 27. ચેસબોર્ડ કોમ્પોનન્ટ્સ

કોમ્પોનન્ટ	ઇમ્પ્લિમેન્ટેશન	હેતુ
ચોરસ	8x8 ગ્રિડ અલ્ટરનેટિંગ રંગો	મુખ્ય બોર્ડ પેટર્ન
રંગો	કાળા અને સફેદ અલ્ટ રનેટિંગ	પરંપરાગત ચેસ પેટર્ન
પેટર્ન લોજિક	(row + col) % 2	ચોરસનો રંગ નક્કી કરે
લૂપ સ્ટ્રક્ચર	નેસ્ટેડ for લૂપ્સ	ગ્રિડમાં ઇટરેટ કરે

ચેસબોર્ડ પેટર્ન લોજિક:



આકૃતિ 6. ચેસબોર્ડ પેટર્ન અલ્ગોરિધમ

- **અલ્ટરનેટિંગ પેટર્ન**: (row + col) % 2 રંગ નક્કી કરે
- **ગ્રિડ સિસ્ટમ**: ચોક્કસ પોઝિશનિંગ સાથે 8x8 ચોરસ
- **સ્કેલેબલ ડિઝાઈન**: ચોરસ સાઈઝ મોડિફાઈ કરવું સરળ
- **નેસ્ટેડ લૂપ્સ**: રો અને કોલમ ઇટરેશન

મેમરી ટ્રીક

``Alternate Colors in Grid Pattern''

પ્રશ્ન 5(અ OR) [3 ગુણ]

turtle માં કેટલા પ્રકારના આકાર છે? યોગ્ય ઉદાહરણ સાથે કોઈપણ એક આકાર સમજાવો.

જવાબ

કોષ્ટક 28. Turtle આકારો

આકાર ટાઈપ	ઉદાહરણો	મેથડ
મૂળભૂત આકારો	વતુળ, ચોરસ, ત્રિકોણ	બિલ્ટ-ઇન ફંક્શન્સ
લાઈન પેટર્ન્સ	સીધી લાઈનો, વક્ર	forward(), backward()
બહુકોણ	પંચકોણ, ષટ્કોણ, અષ્ટકોણ	કોણ સાથે લૂપ
જટિલ આકારો	તારાઓ, સર્પાકાર, ફ્રેક્ટલ્સ	ગાણિતિક પેટર્ન્સ
કસ્ટમ આકારો	યુઝર-ડિફાઈન્ડ પેટર્ન્સ	મૂવ્ઝનું સંયોજન

વતુળ આકારનું ઉદાહરણ:

Listing 18. વતુળ ઉદાહરણ

```
import turtle

def draw_circle_example():
    screen = turtle.Screen()
    circle_turtle = turtle.Turtle()

    # ત્રિજ્યા 50 સાથે વતુળ દોરો
```

```
    circle_turtle.circle(50)

    screen.exitonclick()

draw_circle_example()
```

- **બિલ્ટ-ઇન આકારો**: વતુળ, ચોરસ, ત્રિકોણ સહેલાઈથી ઉપલબ્ધ
- **કસ્ટમ આકારો**: મૂવમેન્ટ સંયોજન વાપરીને બનાવાય
- **ગાણિતિક આકારો**: ચોક્કસ દોરવા માટે ભૂમિતિ વાપરે

મેમરી ટ્રીક

``Turtle Draws Many Shape Types''

પ્રશ્ન 5(બ OR) [4 ગુણ]

Turtle મોડ્યુલની ચાર મૂળભૂત મેથડ્સ વિશે સમજાવો.

જવાબ

કોષ્ટક 29. મૂળભૂત Turtle મેથડ્સ

મેથડ	હેતુ	પેરામીટર્સ	ઉદાહરણ
forward(distance)	turtle ને આગળ મૂવ કરો	પિક્સલ્સમાં અંતર	turtle.forward(100)
backward(distance)	turtle ને પાછળ મૂવ કરો	પિક્સલ્સમાં અંતર	turtle.backward(50)
right(angle)	turtle ને જમણે ફેરવો	ડિગ્રીમાં કોણ	turtle.right(90)
left(angle)	turtle ને ડાબે ફેરવો	ડિગ્રીમાં કોણ	turtle.left(45)

Listing 19. મૂળભૂત મેથડ્સ ઉદાહરણ

```
import turtle

def demonstrate_basic_methods():
    # turtle બનાવો
    demo_turtle = turtle.Turtle()

    # 1. આગળની મૂવમેન્ટ
    demo_turtle.forward(100)  # 100 પિક્સલ્સ આગળ મૂવ કરો

    # 2. જમણે વળવું
    demo_turtle.right(90)    # 90 ડિગ્રી જમણે ફેરવો

    # 3. પાછળની મૂવમેન્ટ
    demo_turtle.backward(50)  # 50 પિક્સલ્સ પાછળ મૂવ કરો

    # 4. ડાબે વળવું
    demo_turtle.left(135)    # 135 ડિગ્રી ડાબે ફેરવો

    turtle.done()

demonstrate_basic_methods()
```

- **મૂવમેન્ટ મેથડ્સ**: અંતર માટે forward() અને backward()
- **રોટેશન મેથડ્સ**: દિશા બદલવા માટે right() અને left()
- **કોઓર્ડિનેટ સિસ્ટમ**: વર્તમાન turtle પોઝિશન અને હેડિંગ પર આધારિત
- **કોણ માપન**: ડિગ્રી (0-360)

મેમરી ટ્રીક

``Forward, Backward, Right, Left Basics''

પ્રશ્ન 5(ક OR) [7 ગુણ]

Turtle વાપરીને ચોરસ, લંબચોરસ અને વતુળ દોરવાનો પાયથન પ્રોગ્રામ લખો.

જવાબ

Listing 20. મલ્ટિપલ આકારો દોરવા

```
import turtle
import math

def setup_drawing_environment():
    """turtle સ્ક્રીન અને દોરવાનું વાતાવરણ સેટઅપ કરો"""

    screen = turtle.Screen()
    screen.bgcolor("lightblue")
    screen.title("આકારો દોરવા: ચોરસ, લંબચોરસ, વતુળ")
    screen.setup(width=800, height=600)

    # મુખ્ય દોરવાનું turtle બનાવો
    shape_turtle = turtle.Turtle()
    shape_turtle.speed(3)
    shape_turtle.pensize(2)

    return screen, shape_turtle

def draw_square(turtle_obj, size, color, position):
    """આપેલ સાઈઝ અને રંગ સાથે ચોરસ દોરો"""

    x, y = position
    turtle_obj.penup()
    turtle_obj.goto(x, y)
    turtle_obj.pendown()

    turtle_obj.color(color)
    turtle_obj.fillcolor(color)
    turtle_obj.begin_fill()

    # 4 સમાન બાજુઓ વાપરીને ચોરસ દોરો
    for _ in range(4):
        turtle_obj.forward(size)
        turtle_obj.right(90)

    turtle_obj.end_fill()

    # લેબલ ઉમેરો
    turtle_obj.penup()
    turtle_obj.goto(x + size//2, y - 30)
    turtle_obj.color("black")
    turtle_obj.write(f"ચોરસ ({size}x{size})", align="center",
                     font=("Arial", 10, "bold"))

def draw_rectangle(turtle_obj, width, height, color, position):
    """આપેલ પરિમાણો અને રંગ સાથે લંબચોરસ દોરો"""

    x, y = position
```

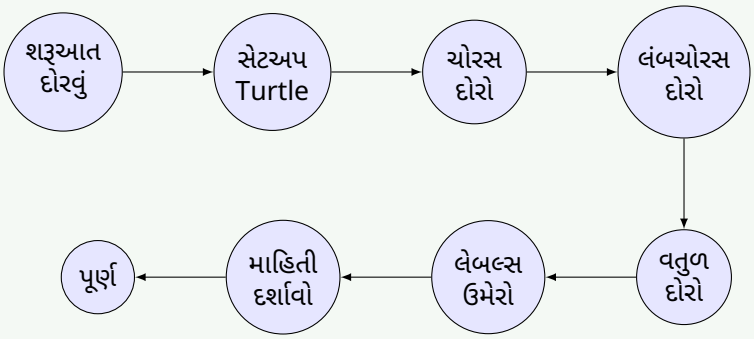
```
    turtle_obj.penup()
    turtle_obj.goto(x, y)
    turtle_obj.pendown()

    turtle_obj.color(color)
    turtle_obj.fillcolor(color)
    turtle_obj.begin_fill()

    # અલ્ટરનેટિંગ પહોળાઈ અને ઊંચાઈ સાથે લંબચોરસ દોરો
    for _ in range(2):
        turtle_obj.forward(width)
        turtle_obj.right(90)
        turtle_obj.forward(height)
        turtle_obj.right(90)

    turtle_obj.end_fill()

    # લેબલ ઉમેરો
    turtle_obj.penup()
    turtle_obj.goto(x + width//2, y - height - 20)
    turtle_obj.color("black")
    turtle_obj.write(f"લંબચોરસ ({width}x{height})", align="center",
                     font=("Arial", 10, "bold"))

def draw_circle(turtle_obj, radius, color, position):
    """આપેલ ત્રિજ્યા અને રંગ સાથે વર્તુળ દોરો"""

    x, y = position
    turtle_obj.penup()
    turtle_obj.goto(x, y - radius)  # વર્તુળના તળિયે પોઝિશન કરો
    turtle_obj.pendown()

    turtle_obj.color(color)
    turtle_obj.fillcolor(color)
    turtle_obj.begin_fill()

    # વર્તુળ દોરો
    turtle_obj.circle(radius)

    turtle_obj.end_fill()

    # ક્ષેત્રફળ ગણતરી સાથે લેબલ ઉમેરો
    area = math.pi * radius * radius
    turtle_obj.penup()
    turtle_obj.goto(x, y - radius - 30)
    turtle_obj.color("black")
    turtle_obj.write(f"વર્તુળ (r={radius}, ક્ષેત્રફળ={area:.1f})", align="center",
                     font=("Arial", 10, "bold"))

def draw_all_shapes():
    """બધા ત્રણ આકારો દોરવા માટેનો મુખ્ય ફંક્શન"""

    screen, shape_turtle = setup_drawing_environment()

    print("ભૌમિતિક આકારો દોરી રહ્યા છીએ...")

    # ચોરસ દોરો
    print("1. ચોરસ દોરી રહ્યા છીએ...")
    draw_square(shape_turtle, 80, "red", (-300, 100))

    # લંબચોરસ દોરો
    print("2. લંબચોરસ દોરી રહ્યા છીએ...")
```

```
    draw_rectangle(shape_turtle, 120, 80, "green", (-50, 100))

    # વતુળ દોરો
    print("3. વતુળ દોરી રહ્યા છીએ...")
    draw_circle(shape_turtle, 60, "blue", (200, 100))

    # શીર્ષક ઉમેરો
    shape_turtle.penup()
    shape_turtle.goto(0, 200)
    shape_turtle.color("purple")
    shape_turtle.write("Python Turtle આકારો", align="center",
                       font=("Arial", 18, "bold"))

    print("બધા આકારો સફળતાપૂર્વક દોરાયા!")
    return screen

# મુખ્ય એક્ઝક્યુશન
def main():
    screen = draw_all_shapes()
    print("\nવિન્ડો બંધ કરવા માટે સ્ક્રીન પર ક્લિક કરો.")
    screen.exitonclick()

# પ્રોગ્રામ ચલાવો
if __name__ == "__main__":
    main()
```

કોષ્ટક 30. આકાર લક્ષણો

આકાર	બાજુઓ	પ્રોપર્ટીઝ	ક્ષેત્રફળ સૂત્ર
ચોરસ	4 સમાન	બધા કોણ 90°	બાજુ
લંબચોરસ	4 (2 જોડ)	વિરુદ્ધ બાજુઓ સમાન	લંબાઈ × પહોળાઈ
વતુળ	0 (વક્ર)	બધા બિંદુઓ સમદૂરસ્થ	× ત્રિજ્યા

આકાર દોરવાની પ્રક્રિયા:



આકૃતિ 7. આકાર દોરવાની પ્રક્રિયા ફ્લો

- **ભૌમિતિક ચોકસાઈ**: ચોક્કસ કોણ અને અંતર માપો
- **વિઝ્યુઅલ આકર્ષણ**: વિવિધ રંગો અને ભરેલા આકારો
- **શૈક્ષણિક મૂલ્ય**: સૂત્રો દર્શાવે
- **ગાણિતિક ગણતરીઓ**: ક્ષેત્રફળ સૂત્રો સામેલ

મેમરી ટ્રીક

``Square Equal, Rectangle Opposite, Circle Round''