

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

Milav Dabgar

January 18, 2025

પ્રશ્ન 1(અ) [3 ગુણ]

પાયથનમાં સેટ અને ડિક્ષનારી વચ્ચેનો તકાવત લખો.

જવાબ

કોષ્ટક 1. સેટ વિરાષ્ટ ડિક્ષનારી તુલના

લક્ષણ	સેટ	ડિક્ષનારી
ડેટા સ્ટોરેજ	કક્ત યુનિક એલિમેન્ટ્સ સ્ટોર કરે	કી-વેલ્યુ પેર સ્ટોર કરે
કમ	અનાર્ડેર્ડ કલેક્શન	ઓર્ડર્ડ (Python 3.7+)
ડુલિક્ટ્સ	ડુલિક્ટ્સની મંજૂરી નથી	કીજ યુનિક હોવી જોઈએ
એક્સેસ	ઇન્ડ્ક્સ દ્વારા એક્સેસ કરી શકતું નથી	કીજ દ્વારા વેલ્યુઝ એક્સેસ કરવા
સિન્ક્સ	{1, 2, 3}	{"key": "value"}

- સેટ: યુનિક, અનાર્ડેર્ડ એલિમેન્ટ્સનો કલેક્શન
- ડિક્ષનારી: યુનિક કીજ 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 ટાઇપ્સ સ્ટોર કરી શકે છે.

કોષ્ટક 2. લિસ્ટ ઓપરેશન્સ

ઓપરેશન	સિન્ક્સ	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

```

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

```

- ઓર્ડર્ડ: એલિમેન્ટ્સ તેમની પોર્જિશન જાળવે છે
- મ્યુટેબલ: બનાવ્યા પછી મોડિફાઈ કરી શકાય છે
- ફ્લૌક્સબલ: કોઈપણ ડેટા ટાઈપ સ્ટોર કરે છે

મેમરી ટ્રીક

“Lists are Ordered and Modifiable”

પ્રશ્ન 1(ક) [૭ ગુણા]

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

જવાબ

ટપલ એક ઓર્ડર્ડ, ઇમ્પ્યુટેબલ કલેક્શન છે જે મલ્ટિપલ આઈમ્સ સ્ટોર કરે છે.

કોષ્ટક 3. ટપલના લક્ષણો

પ્રોપર્ટી	વર્ણન	ઉદાહરણ
ઇમ્પ્યુટેબલ	બનાવ્યા પછી બદલી શકતું નથી	t = (1, 2, 3)
ઓર્ડર્ડ	એલિમેન્ટ્સનો નિર્ધારિત ક્રમ	ઇન્ડેક્સ દ્વારા એક્સેસ
કુલ્લિક્ટ્સ	કુલ્લિક્ટ વેલ્યુઝની મંજૂરી	(1, 1, 2)
ઇન્ડેક્સિંગ	પોર્જિશન દ્વારા એલિમેન્ટ્સ એક્સેસ	t[0]

Listing 2. બે ટપલ વેલ્યુઝને સ્વેપ કરવાનો પ્રોગ્રામ

```

1 # બે ટપલ વેલ્યુઝને સ્વેપ કરવાનો પ્રોગ્રામ
2 def swap_tuple_values(tup, pos1, pos2):
3     # સ્વેપગિ માટે ટપલને લસ્ટિંગ કન્વર્ટ કરો
4     temp_list = list(tup)
5
6     # વેલ્યુઝ સ્વેપ કરો
7     temp_list[pos1], temp_list[pos2] = temp_list[pos2], temp_list[pos1]
8
9     # પાછું ટપલમાં કન્વર્ટ કરો
10    return tuple(temp_list)
11
12 # ઉદાહરણ ઉપયોગ
13 original_tuple = (10, 20, 30, 40, 50)
14 print("મૂળ ટપલ:", original_tuple)
15
16 # પોર્જિશન 1 અને 3 પર વેલ્યુઝ સ્વેપ કરો
17 swapped_tuple = swap_tuple_values(original_tuple, 1, 3)
18 print("સ્વેપગિ પછી:", swapped_tuple)

```

- ઇમ્પ્યુટેબલ: એકવાર બનાવ્યા પછી મોડિફાઈ કરી શકતું નથી
- ઓર્ડર્ડ: એલિમેન્ટ્સનો નિર્ધારિત ક્રમ
- હેટેરોજુનિયસ: વિવિધ ડેટા ટાઈપ્સ સ્ટોર કરી શકે છે

મેમરી ટ્રીક

"Tuples are Immutable and Ordered"

પ્રશ્ન 1(ક) OR [7 ગુણ]

પાયથોનમાં ડિક્ષનરી શું છે? લૂપની મદદથી ડિક્ષનરીને ટ્રાવર્સ કરવાનો પાયથન પ્રોગ્રામ લખો.

જવાબ

ડિક્ષનરી એક યુનિક કીજ સાથે કી-વેલ્યુ પેરનો અનારોડર્ડ કલેક્શન છે.

કોષ્ટક 4. ડિક્ષનરી મેથડ્સ

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

Listing 3. લૂપ વાપરીને ડિક્ષનરી ટ્રાવર્સ કરવાનો પ્રોગ્રામ

```

1 # લૂપ વાપરીને ડિક્ષનરી ટ્રાવર્સ કરવાનો પ્રોગ્રામ
2 student_marks = {
3     'Alice': 85,
4     'Bob': 92,
5     'Charlie': 78,
6     'Diana': 96,
7     'Eve': 89
8 }
9
10 print("ડિક્ષનરી ટ્રાવર્સથી મેથડ્સ:")
11 print("-" * 30)
12
13 # મેથડ 1: ફક્ત કીજ ટ્રાવર્સ કરો
14 print("1. ફક્ત કીજ:")
15 for key in student_marks:
16     print(f" {key}")
17
18 # મેથડ 2: ફક્ત વેલ્યુઝ ટ્રાવર્સ કરો
19 print("\n2. ફક્ત વેલ્યુઝ:")
20 for value in student_marks.values():
21     print(f" {value}")
22
23 # મેથડ 3: કીવેલ્યુ- પર ટ્રાવર્સ કરો
24 print("\n3. કીવેલ્યુ- પર:")
25 for key, value in student_marks.items():
26     print(f" {key}: {value}")
27
28 # મેથડ 4: keys() મેથડ વાપરીને
29 print("\n4. keys() મેથડ વાપરીને:")
30 for key in student_marks.keys():
31     print(f" {key} ને {student_marks[key]} માર્ક્સ મળ્યા")

```

- કી-વેલ્યુ સ્ટોરેજ: દરેક કી એક વેલ્યુ સાથે મેપ થાય છે
- યુનિક કીજ: યુલિકેટ કીજની મંજૂરી નથી
- ફાસ્ટ લુકઅપ: O(1) એવરેજ ટાઈમ કોમ્પ્લેક્શન્ટી

મેમરી ટ્રીક

"Dicts map Keys to Values"

પ્રક્રિયા 2(અ) [3 ગુણ]

પેકેજ શું છે? પેકેજનો ઉપયોગ કરવાના ફાયદાઓની યાદી આપો.

જવાબ

પેકેજ એક ડિરેક્ટરી છે જેમાં મલ્ટિપલ મોડ્યુલ્સ એક્સાથે ઓર્ગનાઇઝ કરવામાં આવે છે.

કોષ્ટક 5. પેકેજના ફાયદાઓ

ફાયદો	વર્ણન
ઓર્ગનાઇઝેશન	સંબંધિત મોડ્યુલ્સને એક્સાથે ગુપ કરે
નેમસ્પેસ	નામિંગ કોન્ફિલિક્ટ્સ ટાળે
રીયુઝનિભિલિટી	કોડ પ્રોજેક્ટ્સમાં ફરીથી વાપરી શકાય
મેઈન્ટેનન્ચિલિટી	મોટા કોડબેસ મેનેજ કરવું સરળ
ડિસ્ટ્રિબ્યુશન	શેર કરવું અને ઈન્સ્ટોલ કરવું સરળ

- મોડ્યુલર સ્ટ્રક્ચર: વધુ સારાં કોડ ઓર્ગનાઇઝેશન
- હાયરાર્કિલ નેમસ્પેસ: નેમ કોન્ફિલિક્ટ્સ અટકાવે
- કોડ રીયુઝ: સોફ્ટવેર રીયુઝનિભિલિટીને પ્રમોટ કરે

મેમરી ટ્રીક

"Packages Organize Related Modules"

પ્રક્રિયા 2(બ) [4 ગુણ]

કોઈપણ બે પેકેજ આચાત પદ્ધતિઓ ઉદાહરણો સાથે સમજાવો.

જવાબ

કોષ્ટક 6. આચાત મેથડ્સ

મેથડ	સિન્ટેક્સ	ઉપયોગ
નોર્મલ આચાત	import package.module	કુલ પાથ સાથે એક્સેસ
ફુમ આચાત	from package import module	ડાયરેક્ટ મોડ્યુલ એક્સેસ
સ્પેસિફિક આચાત	from package.module import function	સ્પેસિફિક આઈમ્સ આચાત
વાઈફકાર્ડ આચાત	from package import *	બધા મોડ્યુલ્સ આચાત

Listing 4. પેકેજ આચાત ઉદાહરણો

```

1 # મેથડ 1: નોર્મલ આચાત
2 import mypackage.calculator
3 result = mypackage.calculator.add(5, 3)
4 print(f"નોર્મલ આચાત પરિણામ: {result}")
5
6 # મેથડ 2: ફુમ આચાત
7 from mypackage import calculator

```

```

8 result = calculator.multiply(4, 6)
9 print(f"જુરૂમ આયાત પરણાં: {result}")

```

- નોર્મલ આયાત: કુલ પેકેજ પાથ જરૂરી
- હમ આયાત: ડાયરેક્ટ મોડયુલ એક્સેસની મંજૂરી
- સ્પેસિફિક ફંક્શન આયાત: ફક્ત જરૂરી ફંક્શન્સ આયાત

મેમરી ટ્રીક

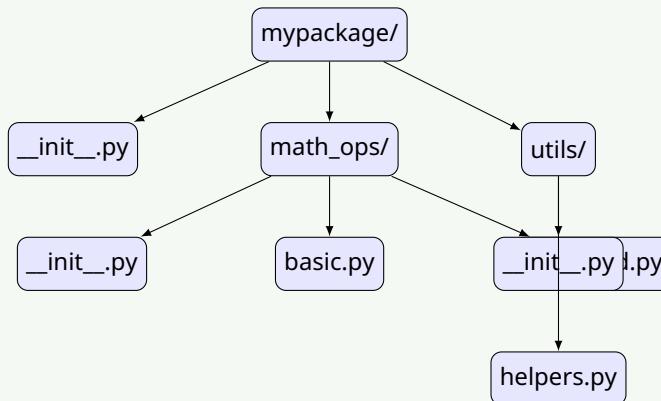
“Import Normally or From Package”

પ્રશ્ન 2(ક) [૭ ગુણ]

ઇન્ટ્રો-પેકેજ સંદર્ભ વિશે ઉદાહરણ સાથે સમજાવો.

જવાબ

ઇન્ટ્રો-પેકેજ રેફરન્સ પેકેજની અંદરના મોડયુલ્સને 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 કુલ પાથ
રિલેટિવ	from . import basic	વર્તમાન પેકેજ
પેરન્ટ	from .. import utils	પેરન્ટ પેકેજ
સિબલિંગ	from ..utils import helpers	સિબલિંગ પેકેજ

Listing 5. ઇન્ટ્રો-પેકેજ રેફરન્સ ઉદાહરણ

```

1 # પેકેજ સ્ટ્રક્ચર ઉદાહરણ
2 # mypackage/math_ops/advanced.py
3 from . import basic # સેમ પેકેજથી રલિટિવ આયાત
4 from ..utils import helpers # સિબલિંગ પેકેજથી આયાત
5
6 def power_operation(base, exp):
7     # બેસકી મોડયુલથી ફંક્શન વાપરીને
8     if basic.is_valid_number(base) and basic.is_valid_number(exp):
9         result = base ** exp
10        # હેલ્પર ફંક્શન વાપરીને

```

```

11     return helpers.format_result(result)
12     return None
13
14 # mypackage/math_ops/basic.py
15 def is_valid_number(num):
16     return isinstance(num, (int, float))
17
18 def add(a, b):
19     return a + b
20
21 # mypackage/utils/helpers.py
22 def format_result(value):
23     return f"પરિણામ: {value:.2f}"

```

- રિલેટિવ આયાત્સ: વર્તમાન પેકેજ માટે ડોટ્સ (.) વાપરો
- એબ્સોલ્યુટ આયાત્સ: કુલ પેકેજ પાથ
- પેકેજ હાયરર્કોઝ: 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 રીયુઝનિભિલિટી	એકવાર લખો, અનેક વાર વાપરો
નેમસ્પેસ	ફંક્શન્સ માટે અલગ નેમસ્પેસ
ઓર્ગનાઇઝેશન	વધુ સારું કોડ સ્ટ્રક્ચર
મેઈન્ટેનન્સ	ડિબગ અને અપડેટ કરવું સરળ
કોલેબોરેશન	મલિટિપલ ડેવલપર્સ કામ કરી શકે

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

મેમરી ટ્રીક

“Modules Make Code Reusable”

પ્રશ્ન 2(અ OR) [4 ગુણ]

કોઈપણ બે મોડ્યુલ આયાત પદ્ધતિ ઉદાહરણ સાથે સમજાવો.

જવાબ

કોષ્ટક 9. મોડ્યુલ આયાત મેથ્ડ્સ

મેથ્ડ	સિન્ક્રેટ્સ	એક્સેસ પેર્ટ્સ
ડાયરેક્ટ આયાત	import module_name	module_name.function()
ફુમ આયાત	from module_name import function	function()
એલિયાસ આયાત	import module_name as alias	alias.function()
વાર્ડિંગ આયાત	from module_name import *	function()

Listing 6. મોડ્યુલ આયાત ઉદાહરણો

```

1 # મેથ્ડ 1: ડાયરેક્ટ આયાત
2 import math
3 result1 = math.sqrt(16)
4 print(f"ડાયરેક્ટ આયાત: {result1}")
5
6 # મેથ્ડ 2: ફુરમ આયાત
7 from math import pi, sin
8 result2 = sin(pi/2)
9 print(f"ફુરમ આયાત: {result2}")

```

- ડાયરેક્ટ આયાત: મોડ્યુલ નામ પ્રીફિક્સ સાથે એક્સેસ
- ફુમ આયાત: પ્રીફિક્સ વગર ડાયરેક્ટ ફંક્શન એક્સેસ
- નેમસ્પેસ કંટ્રોલ: યોગ્ય આયાત મેથ્ડ પરંદ કરો

મેમરી ટ્રીક

``Import Directly or From Module''

પ્રશ્ન 2(ક) OR [7 ગુણ]

વતુળનું ક્ષેત્રફળ અને પરિધિ શોધવા માટેના મોડ્યુલનો પ્રોગ્રામ લખો.

જવાબ

Listing 7. વતુળ ઓપરેશન્સ મોડ્યુલ

```

1 # circle_operations.py મોડ્યુલ( ફાઈલ)
2 import math
3
4 def area(radius):
5     """વતુળનું ક્ષેત્રફળ ગણો"""
6     if radius <= 0:
7         return 0
8     return math.pi * radius * radius
9
10 def circumference(radius):
11     """વતુળનો પરિધિ ગણો"""
12     if radius <= 0:
13         return 0
14     return 2 * math.pi * radius
15
16 def display_info(radius):
17     """વતુળની માહિતી દર્શાવો"""
18     print(f"ત્રાણીયા {radius} વાળું વતુળ:")
19     print(f"ક્ષેત્રફળ: {area(radius):.2f}")

```

```

20 print(f"પરિધિ: {circumference(radius):.2f}")
21
22 # કોન્સ્ટન્ટ્સ
23 PI = math.pi
24
25 # અ) મોડ્યુલને બીજા પ્રોગ્રામમાં આયાત કરો
26 # main_program.py
27 import circle_operations
28
29 radius = 5
30 print("મેથદ 1: સંપૂર્ણ મોડ્યુલ આયાત")
31 area_result = circle_operations.area(radius)
32 circumference_result = circle_operations.circumference(radius)
33
34 print(f"કૃત્રિમાણ: {area_result:.2f}")
35 print(f"પરિધિ: {circumference_result:.2f}")
36
37 # બ) મોડ્યુલમાંથી ચોક્કસ ફંક્શન આયાત કરો
38 # specific_import.py
39 from circle_operations import area, circumference
40
41 radius = 7
42 print("મેથદ 2: ચોક્કસ ફંક્શન્સ આયાત")
43 area_result = area(radius)
44 circumference_result = circumference(radius)
45
46 print(f"કૃત્રિમાણ: {area_result:.2f}")
47 print(f"પરિધિ: {circumference_result:.2f}")

```

કોષ્ટક 10. મોડ્યુલ ફીચર્સ

ફીચર	ઇમ્પ્લિમેટેશન
ફંક્શન્સ	area(), circumference()
એરર હેન્ડલિંગ	નેગેટિવ નિયમ માટે ચેક
કોન્સ્ટન્ટ્સ	PI વલ્યુ
ડોક્યુમેન્ટેશન	ફંક્શન ડોક્યુમેન્ટેશન

- મોડ્યુલ બનાવવું: ફંક્શન્સને .py ફાઈલમાં સેવ કરો
- આયાત લવરીકર્તા: 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

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

- સિન્ટેક્સ એરર: પ્રોગ્રામ રન થાય તે પહેલાં શોધાય
- નટાઈમ 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]

યુગર-ડિફાઈન્ડ એક્સેપ્શન રેઇઝ સ્ટેટમેન્ટ સાથે સમજાવો.

જવાબ

યુગર-ડિફાઈન્ડ એક્સેપ્શન પ્રોગ્રામર્સ દ્વારા બનાવવામાં આવેલા કસ્ટમ એરર કલાસીસ છે.

કોષ્ટક 12. એક્સેપ્શન કોમ્પોન્ટ્સ

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

Listing 8. યુગર-ડિફાઈન્ડ એક્સેપ્શન ઉદાહરણ

```

1 # કસ્ટમ એક્સેપ્શન ડાફુલિન કરો
2 class AgeValidationError(Exception):
3     def __init__(self, age, message="અયોગ્ય વય આપેલ છે"):
4         self.age = age
5         self.message = message
6         super().__init__(self.message)
7
8     def validate_age(age):
9         if age < 0:
10             raise AgeValidationError(age, "વય નેગેટિવ હોઈ શકે નહીં")
11         elif age > 150:
12             raise AgeValidationError(age, "વય 150 કરતાં વધુ હોઈ શકે નહીં")
13         else:
14             print(f"ચોગ્ય વય: {age}")
15
16 # કસ્ટમ એક્સેપ્શનનો ઉપયોગ
17 try:
18     validate_age(-5)
19 except AgeValidationError as e:
20     print(f"એરર: {e.message}, વય: {e.age}")

```

- કસ્ટમ એક્સેપ્શન ક્લાસ: Exception થી ઇનહેરિટ કરે
- રેઇજ સ્ટેટમેન્ટ: મેન્યુઅલી એક્સેપ્શનન્સ ટ્રિગાર કરે
- અર્થપૂર્ણ મેરોજિઝ: ડિબગિંગમાં મદદ કરે

મેમરી ટ્રીક

“Raise Custom Exceptions for Validation”

પ્રશ્ન 3(ક) [૭ ગુણ]

ટ્રાય-એક્સેપ્ટ-ફાઈનલી કલોડ 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	કલીનઅપ કોડ	હંમેશા એક્ઝિક્યુટ થાય

Listing 9. સંપૂર્ણ એક્સેપ્શન હેન્ડલિંગ ઉદાહરણ

```

1 # સંપૂર્ણ એક્સેપ્શન હેન્ડલિંગ ઉદાહરણ
2 def divide_numbers():
3     try:
4         print("ભાગાકાર ઓપરેશન શરૂ કરી રહ્યા છીએ...")
5
6         # યુઝરથી ઇનપુટ મેળવો
7         num1 = float(input("પહેલો નંબર દાખલ કરો: "))
8         num2 = float(input("બીજો નંબર દાખલ કરો: "))
9
10        # ભાગાકાર કરો
11        result = num1 / num2
12
13    except ValueError:
14        print("એરર: કૃપા કરીને ફક્ત યોગ્ય નંબર્સ દાખલ કરો")
15        return None
16
17    except ZeroDivisionError:
18        print("એરર: શૂન્યથી ભાગાકાર કરી શકતો નથી")
19        return None
20
21    except Exception as e:
22        print(f"અનપેક્ષિત એરર આવ્યું: {e}")
23        return None
24
25    else:
26        print(f"ભાગાકાર સફળ: {num1} / {num2} = {result}")
27        return result
28
29    finally:
30        print("ભાગાકાર ઓપરેશન પૂરણ")

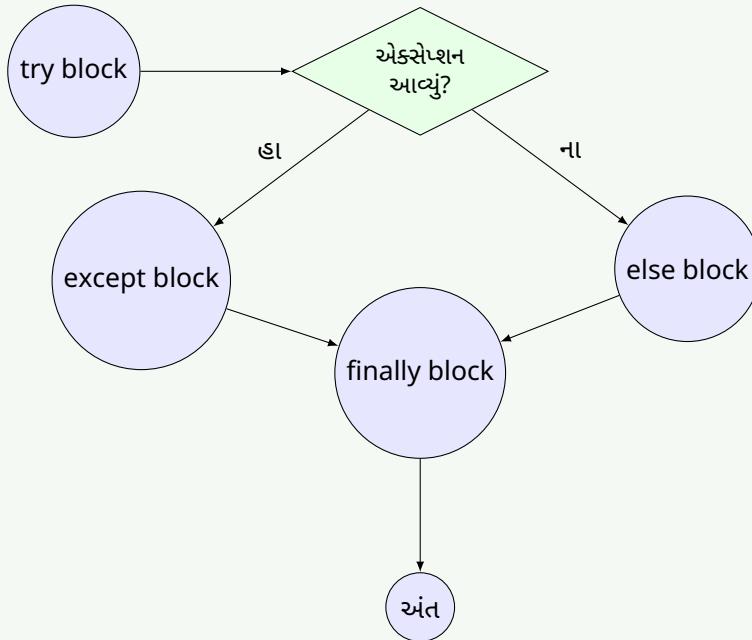
```

```

31     print("રસોર્સસિસ ક્લીન કરી રહ્યા છીએ...")
32
33 # ઉદાહરણ ઉપયોગ
34 result = divide_numbers()
35 if result:
36     print(f"અંતમિ 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? કોઈ પણ બેની તેમના અર્થ સાથે ચાદ્રી 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

વે મુજબ બિલ્ટ-ઇન એક્સેપ્શન્સ:

- ValueError: જ્યારે ફુંક્શનને યોગ્ય ટાઈપ પણ અમાન્ય વેલ્યુ મળે
- TypeError: જ્યારે અયોગ્ય ડેટા ટાઈપ પર ઓપરેશન કરવામાં આવે

મેરી ટ્રીક

“Built-in Exceptions Handle Common Errors”

પ્રશ્ન 3(બ) OR [4 ગુણ]

ટ્રાય-એક્સેપ્ટ કલોઝ ઉદાહરણ સાથે સમજાવો.

જવાબ

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

કોષ્ટક 15. એક્સેપ્શન હેન્ડલિંગ કોમ્પોનેન્ટ્સ

કોમ્પોનેન્ટ	હેતુ	સિસ્ટેક્સ
try	નિષ્ફળ થઈ શકે તેવો કોડ	try:
except	ચોક્સ એક્સેપ્શન હેન્ડલ કરે	except ErrorType:
માલિટિપલ except	વિવિધ એરર્સ હેન્ડલ કરે	માલિટિપલ except બ્લોક્સ
જનરલ except	કોઈપણ એક્સેપ્શન પકડે	except:

Listing 10. ટ્રાય-એક્સેપ્ટ ઉદાહરણ

```

1 # ટ્રાય-એક્સેપ્ટ - કલોઝનું ઉદાહરણ
2 def safe_division():
3     try:
4         # એક્સેપ્શન્સ ઉઠાવી શકે તેવો કોડ
5         dividend = int(input("ભાજ્ય દાખલ કરો: "))
6         divisor = int(input("ભાજક દાખલ કરો: "))
7
8         result = dividend / divisor
9         print(f"પરિણામ: {dividend} / {divisor} = {result}")
10
11     except ValueError:
12         print("એરર: કૃપા કરીને ફક્ત યોગ્ય ઇન્ટાજિસ દાખલ કરો")
13
14     except ZeroDivisionError:
15         print("એરર: શૂન્યથી ભાગાકાર કરી શકતો નથી")
16
17     except Exception as e:
18         print(f"અનપેક્ષિત એરર આવ્યું: {e}")
19
20     print("એક્સેપ્શન હેન્ડલિંગ પછી પ્રોગ્રામ ચાલુ રહે છે")
21

```

22 # ઉદાહરણ ઉપરોગ
23 safe_division()

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

મેમરી ટ્રીક

“Try Risky Code, Except Handles Errors”

પ્રશ્ન 3(ક) OR [7 ગુણ]

ડિવાઇડ બાય જીરો એક્સેપ્શન ફાઈનલી કલોજ સાથે કેચ કરવાનો પ્રોગ્રામ લખો.

જવાબ

Listing 11. ફાઈનલી કલોજ સાથે ડિવાઇડ બાય જીરો

```

1 # ફાઈનલી કલોજ સાથે ડિવાઇડ બાય જીરો હેન્ડલ કરવાનો પ્રોગ્રામ
2 def advanced_calculator():
3     """વ્યાપક એક્સેપ્શન હેન્ડલિંગ સાથે કેલ્ક્યુલેટર"""
4
5     try:
6         print("== એડવાન્સ્ડ કેલ્ક્યુલેટર ==")
7         print("ભાગાકાર માટે બે નંબર દાખલ કરો!")
8
9         # ઇનપુટ સેક્શન
10        numerator = float(input("અંશ દાખલ કરો: "))
11        denominator = float(input("ઉદ્દ દાખલ કરો: "))
12
13        print(f"\n{numerator} ને {denominator} થી ભાગવાનો પ્રયાસ કરી રહ્યા છીએ...")
14
15        # ફરટિકલ ઓપરેશન જે નષ્ટિકુળ થઈ શકે
16        if denominator == 0:
17            raise ZeroDivisionError("શૂન્યથી ભાગાકારની મંજૂરી નથી")
18
19        result = numerator / denominator
20
21        # સફળતાનો સંદેશ
22        print("□ ભાગાકાર સફળ!")
23        print(f"□ પરિણામ: {numerator} / {denominator} = {result:.4f}")
24
25        return result
26
27    except ZeroDivisionError as zde:
28        print(f"□ જીરો ડિવિઝન એરર: {zde}")
29        print("□ કૃપા કરીને શૂન્ય સવિસ્યનો છેદ વાપરો!")
30        return None
31
32    except ValueError as ve:
33        print(f"□ નિયુટ એરર: અમાન્ય ઇનપુટ આપ્યું!")
34        print("□ કૃપા કરીને ફકૃત નંબરકિ વેલ્યુઝ દાખલ કરો!")
35        return None
36
37    except Exception as e:
38        print(f"□ અન્પેક્ષિત એરર: {e}")
39        return None

```

```

40
41     finally:
42         print("\n" + "="*40)
43         print("ક્રીનાયપ ઓપરેશનનુસાર:")
44         print("- કેલ્ક્યુલેટર સેશન બંધ કરી રહ્યા છીએ")
45         print("- આપેશન લોગ સેવ કરી રહ્યા છીએ")
46         print("- મેમરી રસોર્સસિ છોડી રહ્યા છીએ")
47         print("- કેલ્ક્યુલેટર શટાઉન પૂર્ણ")
48         print("*"*40)

49
50 # કેલ્ક્યુલેટર ચલાવો
51 if __name__ == "__main__":
52     result = advanced_calculator()

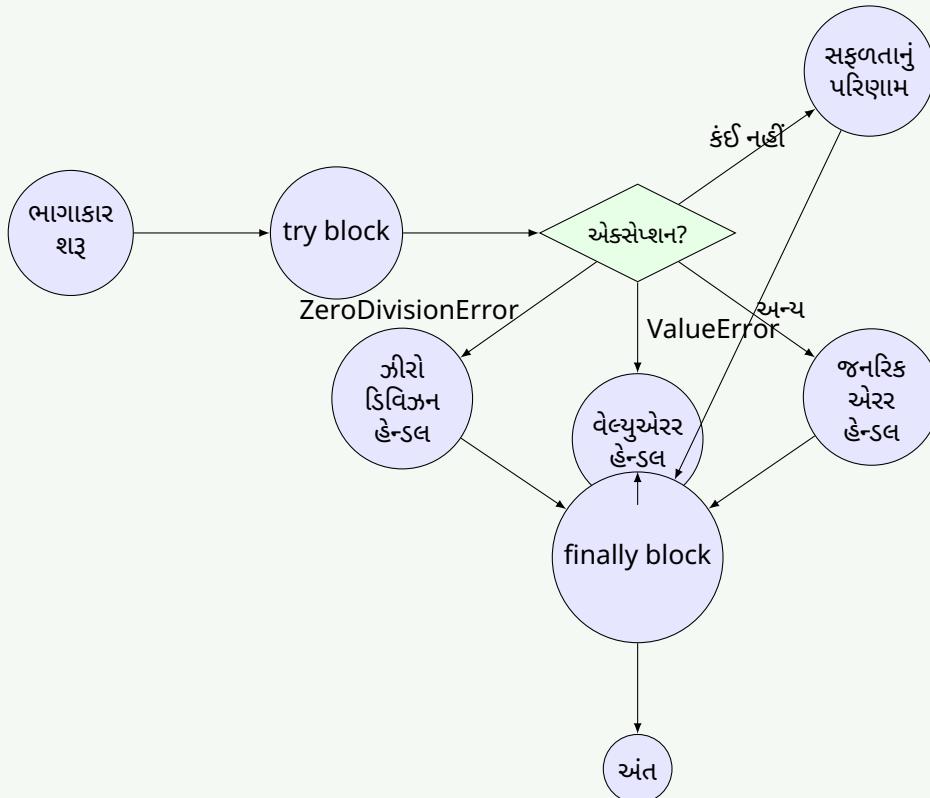
53
54 if result is not None:
55     print("\\"અંતમિન ગણાલું પરિણામ: {result}""")
56 else:
57     print("\\"એરસનાન કારણે ગણતરી નષ્ટફળ""")

```

કોડ 16. એક્સેપ્શન હેન્ડલિંગ ફીચર્સ

ફીચર	દર્શાવેણું
ZeroDivisionError	શૂન્યથી ભાગાકાર માટે ચોક્કસ હેન્ડલિંગ
ValueError	અમાન્ય ઇનપુટ ટાઇપ્સ હેન્ડ કરે
જનરિક એક્સેપ્શન	અનપેક્ષિત એરર્સ પકડે
ફાઈનલી બ્લોક	હંમેશા કલીનાયપ કોડ એક્ઝિક્યુટ

એક્સેપ્શન હેન્ડલિંગ ફ્લો:



આફ્ટિ 3. ડિવાઇડ બાય જીરો એક્સેપ્શન હેન્ડલિંગ ફ્લો

- ચોક્કસ એક્સેપ્શન હેન્ડલિંગ: ZeroDivisionError અલગથી પકડાય

- ફાઈનલી કલોજ: કલીનઅપ માટે હંમેશા એક્ઝિક્યુટ થાય
- રિસોર્સ મેનેજમેન્ટ: એરર્સ છતાં યોગ્ય કલીનઅપ

મેમરી ટ્રીક

“Finally Always Cleans Up Resources”

પ્રશ્ન 4(અ) [3 ગુણ]

વ્યાખ્યાયિત કરો: ફાઈલ, બાઇનરી ફાઈલ, ટેક્સ્ટ ફાઈલ

જવાબ

કોષ્ટક 17. ફાઈલ વ્યાખ્યાઓ

શબ્દ	વ્યાખ્યા	ઉદાહરણ
ફાઈલ	ડિસ્ક પર નામવાળું સ્ટોરેજ સ્થાન	document.txt, image.jpg
બાઇનરી ફાઈલ	બાઇનરી 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
- બાઇનરી ફાઈલ: બાઇનરી 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 ફંક્શન-સ

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

Listing 12. રાઈટ ફંક્શન-સનું ઉદાહરણ

```

1 # write() અને writelines() ફંક્શન-સનું ઉદાહરણ
2 def demonstrate_write_functions():
3
4     # write() ફંક્શન વાપરીને
5     with open("write_demo.txt", "w") as file:
6         file.write("નમસ્કરે વશ્વા!\\n")
7         file.write("આ બીજી લાઈન છે\\n")
8         file.write("આ ત્રીજી લાઈન છે\\n")
9

```

```

10 # writelines() ફંક્શન વાપરીને
11 lines = [
12     "writeln્સ વાપરીને પહેલી લાઈન\n",
13     "writeln્સ વાપરીને બીજી લાઈન\n",
14     "writeln્સ વાપરીને ત્રીજી લાઈન\n"
15 ]
16
17 with open("writeln્સ_demo.txt", "w") as file:
18     file.writeln્સ(lines)
19
20 print("ફાઈલો સફળતાપૂરવક બનાવી!")
21
22 # ટેમેન્સ્યુલેશન ચલાવો
23 demonstrate writeln્સ_functions()

```

મુખ્ય તફાવતો:

- **write()**: એક વખતે એક સ્ટ્રિંગ લખે
- **writeln્સ()**: સિક્વન્સમાંથી મલિટિપલ સ્ટ્રોસ લખે
- ન્યૂલાઈન્સ: \n સાથે મેન્યુઆલી ઉમેરવી જોઈએ
- રિટર્ન વેલ્યુ: બંને લખેલા કરેક્ટર્સની સંખ્યા રિટર્ન કરે

મેમરી ટ્રીક

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

પ્રશ્ન 4(ક) [૭ ગુણ]

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

જવાબ

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

કોષ્ટક 19. પોઝિશન ફંક્શન્સ

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

કોષ્ટક 20. Seek Whence વેલ્યુઝ

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

Listing 13. tell() અને seek() ઉદાહરણ

```

1 # tell() અને seek() ફંક્શન્સનું સંપૂરણ ઉદાહરણ
2 def demonstrate_file_positioning():
3
4     # સેમ્પલ ફાઈલ બનાવો
5     sample_text = "Hello World! This is a sample file for demonstrating tell() and seek() functions."
6

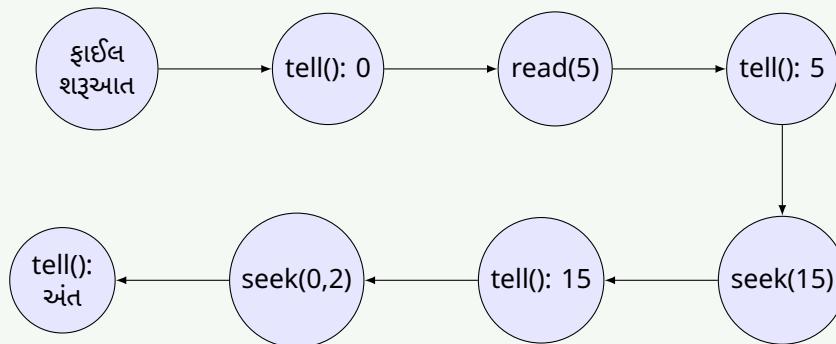
```

```

7   with open("position_demo.txt", "w") as file:
8     file.write(sample_text)
9
10    # tell() અને seek() દર્શાવો
11    with open("position_demo.txt", "r") as file:
12
13      # શરૂઆતની પોઝિશન
14      print(f"1. શરૂઆતની પોઝિશન: {file.tell()}")
15
16      # પહેલા 5 ક્રેક્ટર્સ વાંચો
17      data1 = file.read(5)
18      print(f"2. '{data1}' વાંચ્યું વર્તમાન પોઝિશન: {file.tell()}")
19
20      # પોઝિશન 15 પર જાઓ
21      file.seek(15)
22      print(f"3. seek(15) પછી, પોઝિશન: {file.tell()}")
23
24      # આગળના 10 ક્રેક્ટર્સ વાંચો
25      data2 = file.read(10)
26      print(f"4. '{data2}' વાંચ્યું વર્તમાન પોઝિશન: {file.tell()}")
27
28      # seek(0, 0) વાપરીને શરૂઆતમાં જાઓ
29      file.seek(0, 0)
30      print(f"5. seek(0,0) પછી, પોઝિશન: {file.tell()}")
31
32      # seek(0, 2) વાપરીને અંતમાં જાઓ
33      file.seek(0, 2)
34      print(f"6. seek(0,2) પછી, પોઝિશન: {file.tell()}")
35
36      # વર્તમાન પોઝિશનથી પાછળ જાઓ
37      file.seek(-10, 1)
38      print(f"7. seek(-10,1) પછી, પોઝિશન: {file.tell()}")
39
40      # બાકીનું કન્ટેન્ટ વાંચો
41      remaining = file.read()
42      print(f"8. બાકીનું કન્ટેન્ટ: '{remaining}'")
43
44    # ડેમોનસ્ક્રોલ ચલાવો
45    demonstrate_file_positioning()

```

પોઝિશન કંટ્રોલ ફ્લો:



આકૃતિ 4. ફાઈલ પોઝિશન કંટ્રોલ ફ્લો

- **tell()**: ફાઈલમાં વર્તમાન બાઈટ પોઝિશન રિટર્ન કરે
- **seek()**: ફાઈલ પોઇન્ટરને સ્પેસિફિક પોઝિશન પર મૂવ કરે
- **પોઝિશનિંગ:** રેન્ડમ ફાઈલ એક્સેસ માટે જરૂરી
- **બાઈનરી મોડ:** બાઈટ પોઝિશન સાથે કમ કરે

મેમરી ટ્રીક

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

પ્રશ્ન 4(અ) OR [3 ગુણ]

એબ્સોલ્યુટ અને રિલેટિવ ફાઈલ પાથ શું છે?

જવાબ

કોષ્ટક 21. પાથ ટાઇપ્સ

પાથ ટાઇપ	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:\	ફ્રાઈવ 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 ઓપરિંગ મોડ્સ

મોડ	ટાઇપ	હેતુ	ફાઈલ પોઇન્ટર
'r'	ટેક્સ્ટ	ફક્ત વાંચવા	શરૂઆત
'w'	ટેક્સ્ટ	લખવા (ઓવરરાઇટ)	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

```

1 # વિધિ ફાઈલ મોડ્સના ઉદાહરણો
2 def demonstrate_file_modes():
3
4     # ટેક્સ્ટ ફાઈલ મોડ્સ
5     with open("text_file.txt", "w") as f: # રાઈટ મોડ
6         f.write("\nમસ્તે વશ્વ")
7
8     with open("text_file.txt", "r") as f: # રીડ મોડ
9         content = f.read()
10        print(f"ટેક્સ્ટ કન્ટ૆ન્ટ: {content}")
11
12     # બાઇનરી ફાઈલ મોડ્સ
13     data = b"Binary data example"
14     with open("binary_file.bin", "wb") as f: # બાઇનરી રાઈટ
15         f.write(data)
16
17     with open("binary_file.bin", "rb") as f: # બાઇનરી રીડ
18         binary_content = f.read()
19         print(f"બાઇનરી કન્ટ૆ન્ટ: {binary_content}")
20
21 demonstrate_file_modes()

```

- ટેક્સ્ટ મોડ્સ: એન્કોડિંગ સાથે સ્ટ્રિંગ ડેટા હેન્ડલ કરે
- બાઇનરી મોડ્સ: એન્કોડિંગ વગર રો બાઇટ્સ હેન્ડલ કરે
- ખલ્સ મોડ્સ: રીડિંગ અને રાઈટિંગ બંનેની મંજૂરી

મેમરી ટ્રીક

“Text for Strings, Binary for Bytes”

પ્રશ્ન 4(ક) OR [7 ગુણ]

વિદ્યાર્થીના વિષયનો રેકૉર્ડ જેમ કે બ્રાન્ચ નામ, સેમેસ્ટર, વિષય કોડ અને વિષય નામ બાઇનરી ફાઈલમાં લખવાનો પાયથન પ્રોગ્રામ લખો.

જવાબ

Listing 15. બાઇનરી ફાઈલ વિદ્યાર્થી રેકૉર્ડ્સ

```

1 import pickle
2 import os
3
4 class StudentSubjectRecord:
5     """વિદ્યાર્થી વિષય રેકૉર્ડ્સ હેન્ડલ કરવા માટેનો ફલાસ"""
6
7     def __init__(self, branch_name, semester, subject_code, subject_name):
8         self.branch_name = branch_name
9         self.semester = semester
10        self.subject_code = subject_code
11        self.subject_name = subject_name
12
13    def __str__(self):
14        return f"Branch: {self.branch_name}, Semester: {self.semester}, Code: {self.subject_code}, Subject: {self.subject_name}"
15
16    def write_student_records():
17        """બાઇનરી ફાઈલમાં વિદ્યાર્થી રેકૉર્ડ્સ લખો"""
18

```

```

19 # સેમ્પલ વદ્યારથી રેકોર્ડ્સ
20 records = [
21     StudentSubjectRecord("Information Technology", 2, "4321602", "Advanced Python Programming"),
22     StudentSubjectRecord("Information Technology", 2, "4321601", "Database Management System"),
23     StudentSubjectRecord("Computer Engineering", 3, "4330701", "Data Structure"),
24     StudentSubjectRecord("Information Technology", 2, "4321603", "Web Development"),
25     StudentSubjectRecord("Computer Engineering", 3, "4330702", "Computer Networks")
26 ]
27
28 # pickle વાપરીને બાઈનરી ફાઈલમાં રેકોર્ડ્સ લખો
29 try:
30     with open("student_records.bin", "wb") as binary_file:
31         pickle.dump(records, binary_file)
32
33     print("ઉપયોગથી રેકોર્ડ્સ બાઈનરી ફાઈલમાં સફળતાપૂરવક લખાયા!")
34     print(f"કુલ લખાયેલા રેકોર્ડ્સ: {len(records)}")
35
36 except Exception as e:
37     print(f"બાઈનરી ફાઈલમાં લખતી વખતે એરર: {e}")
38
39 def read_student_records():
40     """બાઈનરી ફાઈલમાંથી વદ્યારથી રેકોર્ડ્સ વાંચો"""
41
42     try:
43         if not os.path.exists("student_records.bin"):
44             print("બાઈનરી ફાઈલ મળી નહીં!")
45             return
46
47         with open("student_records.bin", "rb") as binary_file:
48             records = pickle.load(binary_file)
49
50             print("\n" + "="*60)
51             print("બાઈનરી ફાઈલમાંથી વદ્યારથી વખિય રેકોર્ડ્સ")
52             print("=".*60)
53
54             for i, record in enumerate(records, 1):
55                 print(f"{i}. {record}")
56
57             print("=".*60)
58             print(f"કુલ વાંચેલા રેકોર્ડ્સ: {len(records)}")
59
60     except Exception as e:
61         print(f"બાઈનરી ફાઈલમાંથી વાંચતી વખતે એરર: {e}")
62
63 # મુખ્ય પ્રોગ્રામ એક્ઝિક્યુશન
64 def main():
65     """બાઈનરી ફાઈલ ઓપરેશન્સ દર્શાવવા માટેનો મુખ્ય ફંક્શન"""
66
67     print("== વદ્યારથી વખિય રેકોર્ડ મેનેજમેન્ટ ==\n")
68
69     # શરૂઆતના રેકોર્ડ્સ લખો
70     print("1. બાઈનરી ફાઈલમાં વદ્યારથી રેકોર્ડ્સ લખી રહ્યા છીએ...")
71     write_student_records()
72
73     # રેકોર્ડ્સ વાંચો અને દર્શાવો
74     print("\n2. બાઈનરી ફાઈલમાંથી રેકોર્ડ્સ વાંચી રહ્યા છીએ...")
75     read_student_records()
76
77     # પ્રોગ્રામ એક્ઝિક્યુટ કરો
78     if __name__ == "__main__":
79         main()

```

કોષ્ટક 24. બાઈનરી ફાઈલ ઓપરેશન્સ

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

બાઈનરી ફાઈલ ડેટા ફલો:



આકૃતિ 5. બાઈનરી ફાઈલ સીરિયલાઇઝન ફલો

- બાઈનરી સ્ટોરેજ: ઓફજેક્ટ સીરિયલાઇઝન માટે pickle વાપરે
- કાર્યક્ષમ સ્ટોરેજ: કોમ્પ્યુટર બાઈનરી ફિલ્ટર
- ઓફજેક્ટ પ્રિઝર્વેશન: ડેટા સ્ટકચર ઇન્ટિગ્રિટી જાળવે
- કોસ-પ્લેટફોર્મ: વિવિધ ઓપરેટિંગ સિસ્ટમ્સ પર કામ કરે

મેમરી ટ્રીક

“Pickle Preserves Python Objects”

પ્રશ્ન 5(બ) [3 ગુણ]

વ્યાખ્યાયિત કરો: GUI, CLI

જવાબ

કોષ્ટક 25. ઈન્ટરફેસ વ્યાખ્યાઓ

શાબ્દ	કુલ ફોર્મ	વર્ણન	ઉદાહરણ
GUI	Graphical User Interface	વિન્ડોઝ, બટનો, આઇકોન્સ સાથે વિજ્યુઅલ ઈન્ટરફેસ	Windows, Mac desktop
CLI	Command Line Interface	કમાન્ડ્સ વાપરતું ટેક્સ્ટ-બેઝ ઈન્ટરફેસ	Terminal, Command Prompt

મુખ્ય તફાવતો:

- GUI: યુઝર-ફેન્ડલી, માઉસ-ડ્રિવન, વિજ્યુઅલ એલિમેન્ટ્સ
- CLI: ટેક્સ્ટ-બેઝ, કીબોર્ડ-ડ્રિવન, કમાન્ડ સિન્ટેક્સ
- ઈન્ટરએક્શન: GUI ક્લિક્સ વાપરે, CLI ટાઇપ 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

```

1 import turtle
2
3 def draw_square_with_for_loop():
4     """for લૂપ વાપરીને ચોરસ દોરો"""
5
6     # turtle સેટઅપ
7     screen = turtle.Screen()
8     screen.bgcolor("white")
9     square_turtle = turtle.Turtle()
10    square_turtle.color("blue")
11    square_turtle.pensize(3)
12
13    # for લૂપ વાપરીને ચોરસ દોરો
14    print("for લૂપ વડે ચોરસ દોરી રહ્યા છીએ...")
15    side_length = 100
16
17    for i in range(4):
18        square_turtle.forward(side_length)
19        square_turtle.right(90)
20
21    square_turtle.penup()
22    square_turtle.goto(150, 0)
23    square_turtle.pendown()
24
25    return square_turtle
26
27 def draw_square_with_while_loop(turtle_obj):
28     """while લૂપ વાપરીને ચોરસ દોરો"""
29
30     # બીજા ચોરસ માટે રંગ બદલો
31     turtle_obj.color("red")
32
33     # while લૂપ વાપરીને ચોરસ દોરો
34     print("while લૂપ વડે ચોરસ દોરી રહ્યા છીએ...")
35     side_length = 100
36     sides_drawn = 0
37
38     while sides_drawn < 4:
39         turtle_obj.forward(side_length)
40         turtle_obj.right(90)
41         sides_drawn += 1
42
43     # ટેક્સ્ટ માટે turtle ને સેન્ટરમાં મૂવ કરો
44     turtle_obj.penup()
45     turtle_obj.goto(-50, -150)
46     turtle_obj.write("Blue: for loop, Red: while loop",
47                      font=("Arial", 12, "normal"))
48
49     # મુખ્ય એક્ઝામ્પ્લિયુશન
50     def main():
51         # ચોરસ દોરો
52         turtle_obj = draw_square_with_for_loop()
53         draw_square_with_while_loop(turtle_obj)
54
55         # વન્ડો ખુલ્લી રાખો
56         turtle.Screen().exitonclick()
57
58     # પ્રોગ્રામ ચલાવો
59     main()

```

કોષ્ટક 26. લૂપ તુલના

લૂપ ટાઇપ	સ્ક્રિપ્ટ	ઉપયોગ	કંટ્રોલ
for લૂપ	for i in range(4):	જાણીતા આઈટરેશન્સ	કાઉન્ટર-બેઝિયન્ડ
while લૂપ	while condition:	કન્ડિશનલ આઈટરેશન્સ	કન્ડિશન-બેઝિયન્ડ

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

મેમરી ટ્રીક

``For Count, While Condition''

પ્રશ્ન 5(ક) [૭ ગુણ]

Turtle વાપરીને ચેસબોર્ડ દોરવાનો પાયથન પ્રોગ્રામ લખો.

જવાબ

Listing 17. ચેસબોર્ડ દોરવાનો પ્રોગ્રામ

```

1 import turtle
2
3 def setup_chessboard():
4     """ચેસબોર્ડ માટે turtle સ્ક્રીન અને પ્રોપર્ટીઝ સેટઅપ કરો"""
5
6     screen = turtle.Screen()
7     screen.bgcolor("white")
8     screen.title("Python Turtle વાપરીને ચેસબોર્ડ")
9     screen.setup(width=600, height=600)
10
11    # દોરવા માટે turtle બનાવો
12    chess_turtle = turtle.Turtle()
13    chess_turtle.speed(0) # સૌથી જડપી સ્પીડ
14    chess_turtle.penup()
15
16    return screen, chess_turtle
17
18 def draw_square(turtle_obj, size, fill_color):
19     """આપેલા રંગ સાથે સગિલ ચોરસ દોરો"""
20
21    turtle_obj.pendown()
22    turtle_obj.fillcolor(fill_color)
23    turtle_obj.begin_fill()
24
25    # ચોરસ દોરો
26    for _ in range(4):
27        turtle_obj.forward(size)
28        turtle_obj.right(90)
29
30    turtle_obj.end_fill()
31    turtle_obj.penup()
32
33 def draw_chessboard():

```

```

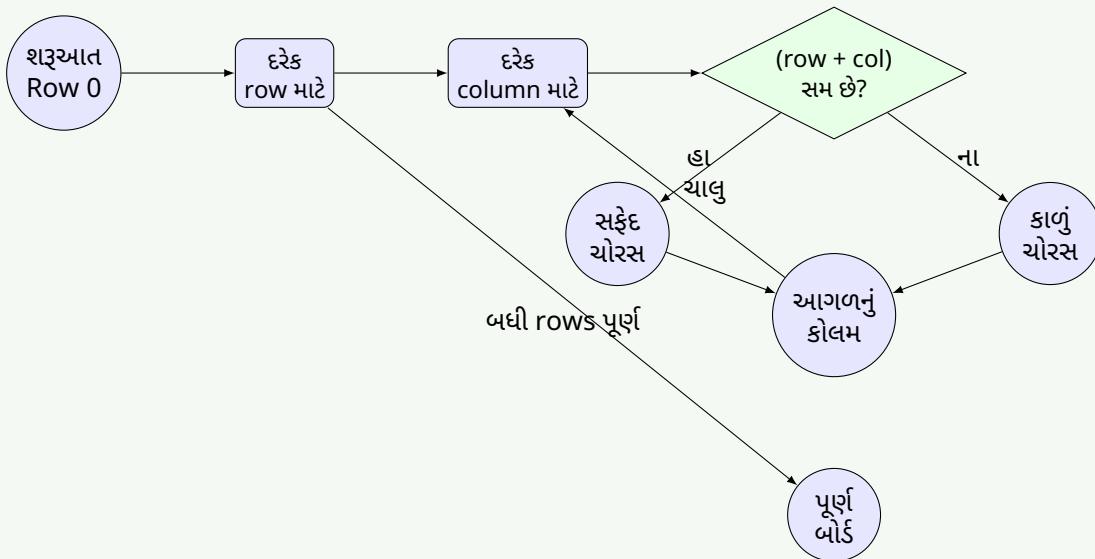
34     """સંપૂર્ણ 8x8 ચેસબોર્ડ દોરો"""
35
36 screen, chess_turtle = setup_chessboard()
37
38 # ચેસબોર્ડ પેરામીટ્રસ
39 square_size = 40
40 board_size = 8
41 start_x = -160
42 start_y = 160
43
44 print("ચેસબોર્ડ દોરી રહ્યા છીએ...")
45
46 # બોર્ડ દોરો
47 for row in range(board_size):
48     for col in range(board_size):
49
50         # પોઝિશન ગણો
51         x = start_x + (col * square_size)
52         y = start_y - (row * square_size)
53
54         # turtle ને પોઝિશન પર મૂવ કરો
55         chess_turtle.goto(x, y)
56
57         # ચોરસનો રંગ નક્કી કરો અલ્ટરનેટિંગ (પેટર્ન)
58         if (row + col) % 2 == 0:
59             color = "white"
60         else:
61             color = "black"
62
63         # ચોરસ દોરો
64         draw_square(chess_turtle, square_size, color)
65
66 # શીર્ષક ઉમેરો
67 chess_turtle.goto(0, start_y + 30)
68 chess_turtle.write("Python Turtle ચેસબોર્ડ", align="center",
69     font=("Arial", 16, "bold"))
70
71 print("ચેસબોર્ડ સફળતાપૂર્વક બન્યું!")
72 return screen
73
74 # મુખ્ય એક્ઝાક્યુશન
75 def main():
76     """ચેસબોર્ડ બનાવવા માટેનો મુખ્ય ફુક્શન"""
77
78     screen = draw_chessboard()
79     print("વન્દો બંધ કરવા માટે સ્ક્રીન પર ફ્લાંક કરો.")
80     screen.exitonclick()
81
82 # પ્રોગ્રામ ચલાવો
83 if __name__ == "__main__":
84     main()

```

કોષ્ટક 27. ચેસબોર્ડ કોમ્પોનન્ટ્સ

કોમ્પોનન્ટ	ઇમ્પ્લિકેશન	હેતુ
ચોરસ	8x8 ગિડ 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	ગિડમાં ઇટરેટ કરે

ચેસબોર્ડ પેટર્ન લોજિક:



આકૃતિ 6. ચેસબોર્ડ પેટર્ન અલોરિધમ

- અલ્ટરનેટિંગ પેટર્ન: $(\text{row} + \text{col}) \% 2$ રંગ નક્કી કરે
- ગિડ સિસ્ટમ: ચોક્કસ પોજિશનિંગ સાથે 8×8 ચોરસ
- સ્કેલેબલ ડિઝાઇન: 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 પેટ-ર્સ	સીધી લાઈનો, વક્ત	forward(), backward()
બહુકોણ	પંચકોણ, ષટકોણ, અષ્ટકોણ	કોણ સાથે લૂપ
જટિલ આકારો	તારાઓ, સર્પિન્ન, ફેક્ટલ્સ	ગાણિતિક પેટન્સ
કસ્ટમ આકારો	યુઝર-ડિફાઈન્ડ પેટન્સ	મૂળનું સંચોળન

વતુળ આકારનું ઉદાહરણ:

Listing 18. વતુળ ઉદાહરણ

```

1 import turtle
2
3 def draw_circle_example():
4     screen = turtle.Screen()
5     circle_turtle = turtle.Turtle()
6
7     # ત્રણ્યા 50 સાથે વતુળ દોરો
  
```

```

8   circle_turtle.circle(50)
9
10  screen.exitonclick()
11
12  draw_circle_example()

```

- બિલ્ડ-ઇન આકારો: વતુળ, ચોરસ, ટ્રિકોણ સહેલાઈથી ઉપલબ્ધ
- કસ્ટમ આકારો: મૂવમેન્ટ સંચોજન 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 જમણો ફેરવો	ડિગ્રીમાં કોણ	turtle.right(90)
left(angle)	turtle ને ડાબે ફેરવો	ડિગ્રીમાં કોણ	turtle.left(45)

Listing 19. મૂળભૂત મેથડ્સ ઉદાહરણ

```

1 import turtle
2
3 def demonstrate_basic_methods():
4     # turtle બનાવો
5     demo_turtle = turtle.Turtle()
6
7     # 1. આગળની મૂવમેન્ટ
8     demo_turtle.forward(100) # 100 પિક્સલ્સ આગળ મૂવ કરો
9
10    # 2. જમણો વળ્ણું
11    demo_turtle.right(90) # 90 ડિગ્રી જમણો ફેરવો
12
13    # 3. પાછળની મૂવમેન્ટ
14    demo_turtle.backward(50) # 50 પિક્સલ્સ પાછળ મૂવ કરો
15
16    # 4. ડાબે વળ્ણું
17    demo_turtle.left(135) # 135 ડિગ્રી ડાબે ફેરવો
18
19    turtle.done()
20
21 demonstrate_basic_methods()

```

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

મેમરી ટ્રીક

"Forward, Backward, Right, Left Basics"

પ્રક્રિયા 5(ક) OR [7 ગુણ]

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

જવાબ

Listing 20. માલિટિપલ આકારો દોરવા

```

1 import turtle
2 import math
3
4 def setup_drawing_environment():
5     """turtle સ્ક્રીન અને દોરવાનું વાતાવરણ સેટઅપ કરો"""
6
7     screen = turtle.Screen()
8     screen.bgcolor("lightblue")
9     screen.title("આકારો દોરવા: ચોરસ, લંબચોરસ, વતુળ")
10    screen.setup(width=800, height=600)
11
12    # મુખ્ય દોરવાનું turtle બનાવો
13    shape_turtle = turtle.Turtle()
14    shape_turtle.speed(3)
15    shape_turtle.pensize(2)
16
17    return screen, shape_turtle
18
19 def draw_square(turtle_obj, size, color, position):
20     """આપેલ સાઈઝ અને રંગ સાથે ચોરસ દોરો"""
21
22     x, y = position
23     turtle_obj.penup()
24     turtle_obj.goto(x, y)
25     turtle_obj.pendown()
26
27     turtle_obj.color(color)
28     turtle_obj.fillcolor(color)
29     turtle_obj.begin_fill()
30
31     # 4 સમાન બાજુઓ વાપરીને ચોરસ દોરો
32     for _ in range(4):
33         turtle_obj.forward(size)
34         turtle_obj.right(90)
35
36     turtle_obj.end_fill()
37
38     # લેબલ ઉમેરો
39     turtle_obj.penup()
40     turtle_obj.goto(x + size//2, y - 30)
41     turtle_obj.color("black")
42     turtle_obj.write(f"ચોરસ ({size}x{size})", align="center",
43                     font=("Arial", 10, "bold"))
44
45 def draw_rectangle(turtle_obj, width, height, color, position):
46     """આપેલ પરમિએટ અને રંગ સાથે લંબચોરસ દોરો"""
47
48     x, y = position

```

```

49 turtle_obj.penup()
50 turtle_obj.goto(x, y)
51 turtle_obj.pendown()
52
53 turtle_obj.color(color)
54 turtle_obj.fillcolor(color)
55 turtle_obj.begin_fill()
56
57 # અલ્ટ્રાનેટિંગ પહોળાઈ અને ઊંચાઈ સાથે લંબચોરસ દોરો
58 for _ in range(2):
59     turtle_obj.forward(width)
60     turtle_obj.right(90)
61     turtle_obj.forward(height)
62     turtle_obj.right(90)
63
64 turtle_obj.end_fill()
65
66 # લેબલ ઉમેરો
67 turtle_obj.penup()
68 turtle_obj.goto(x + width//2, y - height - 20)
69 turtle_obj.color("black")
70 turtle_obj.write(f"લંબચોરસ ({width}x{height})", align="center",
71     font=("Arial", 10, "bold"))
72
73 def draw_circle(turtle_obj, radius, color, position):
74     """આપેલ ત્રણિયા અને રંગ સાથે વતુળ દોરો"""
75
76 x, y = position
77 turtle_obj.penup()
78 turtle_obj.goto(x, y - radius) # વતુળના તળાયિ પોઝિશન કરો
79 turtle_obj.pendown()
80
81 turtle_obj.color(color)
82 turtle_obj.fillcolor(color)
83 turtle_obj.begin_fill()
84
85 # વતુળ દોરો
86 turtle_obj.circle(radius)
87
88 turtle_obj.end_fill()
89
90 # ફ્રેન્ટ્રફલ ગણતતી સાથે લેબલ ઉમેરો
91 area = math.pi * radius * radius
92 turtle_obj.penup()
93 turtle_obj.goto(x, y - radius - 30)
94 turtle_obj.color("black")
95 turtle_obj.write(f"વતુળ (r={radius}, ફ્રેન્ટ્રફલ={area:.1f})", align="center",
96     font=("Arial", 10, "bold"))
97
98 def draw_all_shapes():
99     """બધા ત્રણા આકારો દોરવા માટેનો મુખ્ય ફંક્શન"""
100
101 screen, shape_turtle = setup_drawing_environment()
102
103 print("ભૌમતિક આકારો દોરી રહ્યા છીએ...")
104
105 # ચોરસ દોરો
106 print("1. ચોરસ દોરી રહ્યા છીએ...")
107 draw_square(shape_turtle, 80, "red", (-300, 100))
108
109 # લંબચોરસ દોરો
110 print("2. લંબચોરસ દોરી રહ્યા છીએ...")

```

```

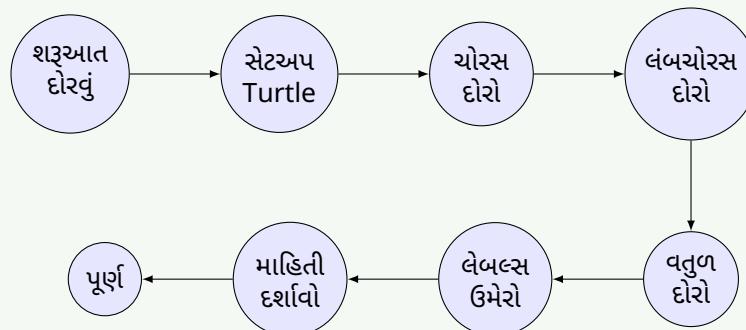
111 draw_rectangle(shape_turtle, 120, 80, "green", (-50, 100))
112
113 # વતુળ દોરો
114 print("3. વતુળ દોરી રહ્યા છીએ...")
115 draw_circle(shape_turtle, 60, "blue", (200, 100))
116
117 # શીર્ષક ઉમેરો
118 shape_turtle.penup()
119 shape_turtle.goto(0, 200)
120 shape_turtle.color("purple")
121 shape_turtle.write("Python Turtle આકારો", align="center",
122         font=("Arial", 18, "bold"))
123
124 print("બધા આકારો સફળતાપૂર્વક દોરાયા!")
125 return screen
126
127 # મુખ્ય એક્ઝાહિયુશન
128 def main():
129     screen = draw_all_shapes()
130     print("\nનિયમ બંધ કરવા માટે સ્ક્રીન પર ક્લાય કરો.")
131     screen.exitonclick()
132
133 # પ્રોગ્રામ ચલાવો
134 if __name__ == "__main__":
135     main()

```

કોષ્ટક 30. આકાર લક્ષણો

આકાર	બાજુઓ	પ્રોપર્ટીજ	ક્ષેત્રફળ સૂત્ર
ચોરસ	4 સમાન	બધા કોણ 90°	બાજુની લાંબીની ગઠન
લંબચોરસ	4 (2 જોડ)	વિરુદ્ધ બાજુઓ સમાન	લંબાઈ × પહોળાઈ
વતુળ	0 (વક)	બધા વિન્દુઓ સમદૂરસ્થ	અનુભૂતિ × ત્રિજ્યા

આકાર દોરવાની પ્રક્રિયા:



આકૃતિ 7. આકાર દોરવાની પ્રક્રિયા ફલો

- ભૌમિતિક ચોક્કસાઈ: ચોક્કસ કોણ અને અંતર માપો
- વિન્યુઅલ આકર્ષણ: વિવિધ રંગો અને લરેલા આકારો
- શૈક્ષણિક મૂલ્ય: સૂત્રો દર્શાવે
- ગાણિતિક ગણતરીઓ: ક્ષેત્રફળ સૂત્રો સામેલ

મેમરી ટ્રીક

“Square Equal, Rectangle Opposite, Circle Round”