

Python Programming (4311601) - Winter 2024 Solution

Milav Dabgar

January 13, 2025

પ્રશ્ન પ્રશ્ન 1(અ) [03 ગુણ]

પ્રોબ્લમ સોલવિંગ, અલોરિધમ અને સ્યુડો કોડ વ્યાખ્યાપિત કરો.

જવાબ

વ્યાખ્યાઓ:

કોષ્ટક 1. Core Concepts

શબ્દ	વ્યાખ્યા
પ્રોબ્લમ સોલવિંગ	તર્કસંગત વિચારસરણી વાપરીને જટિલ સમસ્યાઓનાં ઉકેલ શોધવાની પદ્ધતિ
અલોરિધમ	મર્યાદિત ઓપરેશન સાથે સમસ્યા ઉકેલવાની પગલું-દર-પગલું પ્રક્રિયા
સ્યુડો કોડ	સામાન્ય અંગેજુ જેવા syntax નો ઉપયોગ કરીને program logic નું અનૌપચારિક વર્ણન

મુખ્ય મુદ્દાઓ:

- પ્રોબ્લમ સોલવિંગ: જટિલ સમસ્યાઓને વ્યવસ્થિત પગલાઓમાં વહેંચવું
- અલોરિધમ: મર્યાદિત, નિશ્ચિત, અસરકારક અને યોગ્ય આઉટપુટ આપતું હોવું જોઈએ
- સ્યુડો કોડ: માનવ ભાષા અને programming કોડ વર્ચેનો સેતુ

મેમરી ટ્રીક

"PAP - Problem, Algorithm, Pseudo"

પ્રશ્ન પ્રશ્ન 1(બ) [04 ગુણ]

ફ્લોચાર્ટના જુદા જુદા સિમ્બોલ સમજાવો. બે નંબર માંથી મહત્તમ નંબર શોધતો ફ્લોચાર્ટ ડિઝાઇન કરો.

જવાબ

ફ્લોચાર્ટ સિમ્બોલ:

કોષ્ટક 2. Symbols

સિમ્બોલ	આકાર	હેતુ
અંડાકાર	□	શરૂઆત/અંત
લંબચોરસ	□	પ્રક્રિયા/કિયા
હીરો	□	નિર્ણય
સમાંતર ચતુર્ભુણ	□	ઇનપુટ/આઉટપુટ

બે નંબરના મહત્તમ માટે ફ્લોચાર્ટ:

શરૂઆત

ઇનપુટ A, B

ડા

A > B?

ના

Max = A

Max = B

Max દર્શાવો

અંત

સમજૂતી:

- શરૂઆત/અંત: પ્રવેશ અને બહાર નીકળવાના બિંદુઓ
- ઇનપુટ/આઉટપુટ: ડેટા ફલો ઓપરેશન્સ
- નિષ્ણય: શરતી branching
- પ્રક્રિયા: ગણિતરીના પગલાં

મેમરી ટ્રીક**“SIFO - Start, Input, Process, Output”****પ્રક્રિયા 1(ક) [07 ગુણ]**

પાયથોનના વિવિધ એરિથમેટિક ઓપરેટરોની યાદી બનાવો. વિવિધ એરિથમેટિક ઓપરેશન્સ માટેનો Python કોડ લખો.

જવાબ**એરિથમેટિક ઓપરેટરો:****કોષ્ટક 3. Arithmetic Operators**

ઓપરેટર	સિમ્બોલ	ઉદાહરણ	પરિણામ
ઉમેરો	+	$5 + 3$	8
બાદબાડી	-	$5 - 3$	2
ગુણાકાર	*	$5 * 3$	15
ભાગાકાર	/	$5 / 3$	1.667
ફ્લોર ડિવિઝન	//	$5 // 3$	1
મોડ્યુલસ	%	$5 \% 3$	2
ઘાત	**	$5 ** 3$	125

કોડ:

```

1 a = 10
2 b = 3
3 print(f"Addition: {a + b}")
4 print(f"Subtraction: {a - b}")
5 print(f"Multiplication: {a * b}")
6 print(f"Division: {a / b}")
7 print(f"Floor Division: {a // b}")
8 print(f"Modulus: {a % b}")
9 print(f"Power: {a ** b}")

```

મેમરી ટ્રીક

“Add-Sub-Mul-Div-Floor-Mod-Pow”

પ્રશ્ન પ્રશ્ન 1(ક OR) [07 ગુણ]

પાયથોનના વિવિધ કંપેરિઝન ઓપરેટરોની ચાદી બનાવો. વિવિધ કંપેરિઝન ઓપરેશન્સ માટેનો Python કોડ લખો.

જવાબ

કંપેરિઝન ઓપરેટરો:

કોષ્ટક 4. Comparison Operators

ઓપરેટર	સિમ્બોલ	હેતુ	ઉદાહરણ
સમાન	==	સમાનતા ચકાસો	$5 == 3 \rightarrow False$
અસમાન	!=	અસમાનતા ચકાસો	$5 != 3 \rightarrow True$
મોટું	>	મોટું ચકાસો	$5 > 3 \rightarrow True$
નાનું	<	નાનું ચકાસો	$5 < 3 \rightarrow False$
મોટું સમાન	>=	મોટું/સમાન ચકાસો	$5 >= 3 \rightarrow True$
નાનું સમાન	<=	નાનું/સમાન ચકાસો	$5 <= 3 \rightarrow False$

કોડ:

```

1 x = 8
2 y = 5
3 print(f"Equal: {x == y}")
4 print(f"Not Equal: {x != y}")
5 print(f"Greater: {x > y}")
6 print(f"Less: {x < y}")
7 print(f"Greater Equal: {x >= y}")
8 print(f"Less Equal: {x <= y}")

```

મેમરી ટ્રીક

"Equal-Not-Greater-Less-GreaterEqual-LessEqual"

પ્રશ્ન પ્રશ્ન 2(અ) [03 ગુણ]

મેમબરશિપ ઓપરેટર્સ ઉપર ટૂંક નોંધ લખો.

જવાબ

મેમબરશિપ ઓપરેટર્સ:

કોષ્ટક 5. Membership Operators

ઓપરેટર	હેતુ	ઉદાહરણ
in	એલિમેન્ટ અસ્થિત્વ ચકાસો	'a' in 'apple' → True
not in	એલિમેન્ટ અનસ્થિત્વ ચકાસો	'z' not in 'apple' → True

મુખ્ય મુદ્દાઓ:

- in ઓપરેટર: જો એલિમેન્ટ sequence માં મળે તો True આપે
- not in ઓપરેટર: જો એલિમેન્ટ sequence માં ન મળે તો True આપે
- ઉપયોગ: Lists, strings, tuples, dictionaries માં

મેમરી ટ્રીક

"In-Not-In for membership testing"

પ્રશ્ન પ્રશ્ન 2(બી) [04 ગુણ]

પાયથોન વ્યાખ્યાયિત કરો. પાયથોન પ્રોગ્રામિંગની વિવિધ એપ્લિકેશનો લખો.

જવાબ

પાયથોન વ્યાખ્યા: સરળતા અને વાંચનીયતા માટે જાણીતી high-level, interpreted programming language.

એપ્લિકેશનો:

કોષ્ટક 6. Applications

એપ્લિકેશન ક્ષેત્ર	ઉદાહરણો
વેબ ડેવલપમેન્ટ	Django, Flask frameworks
ડેટા સાયન્સ	NumPy, Pandas, Matplotlib
AI/ML	TensorFlow, Scikit-learn
ડેસ્કટોપ એપ્સ	Tkinter, PyQt
ગેમ ડેવલપમેન્ટ	Pygame library

વિશેષતાઓ:

- Interpreted: compilation ની જરૂર નથી
- Cross-platform: બહુવિધ OS પર ચાલે છે
- વિશાળ libraries: વ્યાપક standard library

મેમરી ટ્રીક

"Web-Data-AI-Desktop-Games"

પ્રશ્ન પ્રશ્ન 2(ક) [07 ગુણ]

પાયથોન પ્રોગ્રામ લખો જે નીચેની વિગતોનો ઉપયોગ કરીને વીજળી બિલની ગણતરી કરે છે.

જવાબ

દરોનું ટેબલ:

કોષ્ટક 7. Electricity Rates

યુનિટ રેન્જ	દર પ્રતી યુનિટ
≤ 100	₹ 5.00
101-200	₹ 7.50
201-300	₹ 10.00
≥ 301	₹ 15.00

કોડ:

```

1 units = int(input("Enter consumed units: "))

2
3 if units <= 100:
4     bill = units * 5.00
5 elif units <= 200:
6     bill = units * 7.50
7 elif units <= 300:
8     bill = units * 10.00
9 else:
10    bill = units * 15.00
11
12 print(f"Total Bill: Rs {bill}")

```

સમજૂતી:

- શરતી તક્ક: if-elif-else structure
- દર ગણતરી: યુનિટ slabs આધારિત
- યુએર ઇનપુટ: interactive billing system

મેમરી ટ્રીક

"Input-Check-Calculate-Display"

પ્રશ્ન પ્રશ્ન 2(અ) OR [03 ગુણ]

આઇડેન્ટિટી ઓપરેટર્સ ઉપર ટૂંક નોંધ લખો.

જવાબ

આઇડેન્ટિટી ઓપરેટર્સ:

કોષ્ટક 8. Identity Operators

ઓપરેટર	હેતુ	ઉદાહરણ
is	સમાન ઓફ્જેક્ટ ચકાસો	a is b
is not	જુદા ઓફ્જેક્ટ ચકાસો	a is not b

મુખ્ય મુદ્દાઓ:

- is ઓપરેટર: ઓફ્જેક્ટ identity સરખાવે, values નહીં
- is not ઓપરેટર: ઓફ્જેક્ટ્સ જુદા છે કે નહીં ચકાસે
- મેમરી સરખામણી: સમાન મેમરી સ્થાન ચકાસે

મેમરી ટ્રીક

``Is-IsNot for object identity''

પ્રશ્ન પ્રશ્ન 2(બ) OR [04 ગુણ]

પાયથોનમાં ઇન્ડેન્ટેશન શું છે? પાયથોનની વિવિધ વિશેષતાઓ સમજાવો.

જવાબ

ઇન્ડેન્ટેશન: કોડ બ્લોક્સ વ્યાખ્યાયિત કરવા માટે લાઈનની શરૂઆતમાં whitespace.

વિશેષતાઓ:

કોષ્ટક 9. Python Features

વિશેષતા	વર્ણન
સરળ Syntax	વાંચવા અને લખવામાં સરળ
Interpreted	compilation step નથી
Object-Oriented	OOP concepts સપોર્ટ કરે
Cross-Platform	બહુવિધ OS પર ચાલે
વિશાળ Library	વ્યાપક standard library

મહત્વ:

- ઇન્ડેન્ટેશન: curly braces ને બદલે છે
- સુસંગત: સામાન્ય રીતે પ્રતિ level 4 spaces
- ફરજિયાત: કોડ માળખું બનાવે છે

મેમરી ટ્રીક

``Simple-Interpreted-Object-Cross-Large''

પ્રશ્ન પ્રશ્ન 2(ક) OR [07 ગુણ]

પાયથોન પ્રોગ્રામ લખો જે નીચેની વિગતોનો ઉપયોગ કરીને વિદ્યાર્થીના વર્ગ/ગ્રેડની ગણતરી કરતો પાયથોન પ્રોગ્રામ લખો.

જવાબ

ગ્રેડિંગ ટેબલ:

કોષ્ટક 10. Grading Scheme

કાવારી	ગ્રેડ
≥ 70	ડિસ્ટિંક્ષન
60-69	ફર્સ્ટ કલાસ
50-59	સેકન્ડ કલાસ
35-49	પાસ કલાસ
< 35	નિષ્ફળ

કોડ:

```

1 percentage = float(input("Enter percentage: "))
2
3 if percentage >= 70:
4     grade = "Distinction"
5 elif percentage >= 60:
6     grade = "First Class"
7 elif percentage >= 50:
8     grade = "Second Class"
9 elif percentage >= 35:
10    grade = "Pass Class"
11 else:
12    grade = "Fail"
13
14 print(f"Grade: {grade}")

```

સમજૂતી:

- બહુવિધ શરતો: Nested if-elif structure
- ગ્રેડ નિર્ધારણ: ટકાવારી ranges આધારિત
- Float ઇનપુટ: દરખાંશ ટકાવારી handle કરે

મેમરી ટ્રીક

``Distinction-First-Second-Pass-Fail''

પ્રશ્ન પ્રશ્ન 3(અ) [03 ગુણ]

સિલેક્શન કંટ્રોલ સ્ટેટમેન્ટ શું છે? તેની યાદી બનાવો.

જવાબ

સિલેક્શન કંટ્રોલ સ્ટેટમેન્ટ:

કોષ્ટક 11. Selection Statements

સ્ટેટમેન્ટ પ્રકાર	હેતુ
if	એક શરત ચકાસણી
if-else	બે-માર્ગી branching
if-elif-else	બહુ-માર્ગી branching
nested if	શરતોની અંદર શરતો

મુખ્ય ઘ્યાલો:

- Selection statements: શરતો આધારે program flow control કરે
- Boolean evaluation: True/False logic વાપરે
- Branching: execution ના જુદા રસ્તાઓ

મેમરી ટ્રીક

"If-IfElse-IfElif-Nested"

પ્રશ્ન પ્રશ્ન 3(બ) [04 ગુણ]

નેસ્ટેડ લૂપ ઉપર ટૂંક નોંધ લખો.

જવાબ

નેસ્ટેડ લૂપ:

કોષ્ટક 12. Loop Structure

લૂપ મકર	માળખું
બાધ્ય લૂપ	iterations control કરે
આંતરિક લૂપ	દરેક બાધ્ય iteration માટે સંપૂર્ણ execute થાય
કુલ Iterations	બાધ્ય × આંતરિક

મુખ્ય મુદ્દાઓ:

- Nested માળખું: બીજા લૂપની અંદર લૂપ
- સંપૂર્ણ execution: આંતરિક લૂપ પૂરું થાય પછી બાધ્ય આગળ વધે
- Pattern creation: 2D structures માટે ઉપયોગી

કોડ ઉદાહરણ:

```

1 for i in range(3):
2     for j in range(2):
3         print(f"i={i}, j={j}")

```

મેમરી ટ્રીક

"Outer-Inner-Complete-Pattern"

પ્રશ્ન પ્રશ્ન 3(ક) [07 ગુણ]

યુઝર ડિફેન્ડન ફંક્શન લખો જે 1 થી 100 સુધીની બધી સંખ્યાઓ દર્શાવે, જે 4 થી વિભાજ્ય છે.

જવાબ

કોડ:

```

1 def display_divisible_by_4():
2     print("Numbers divisible by 4 from 1 to 100:")
3     for num in range(1, 101):
4         if num % 4 == 0:
5             print(num, end=" ")
6     print()
7
8 # Function call
9 display_divisible_by_4()

```

Return સાથે વિકલ્પ:

```

1 def get_divisible_by_4():
2     return [num for num in range(1, 101) if num % 4 == 0]
3
4 result = get_divisible_by_4()
5 print(result)

```

મુખ્ય ઘણાલો:

- ફંક્શન વ્યાખ્યા: def keyword નો ઉપયોગ
- Range ફંક્શન: 1 થી 100 iteration
- Modulus ચકાસણી: num % 4 == 0 શરત
- List comprehension: વૈકલ્પિક અભિગમ

મેમરી ટ્રીક

``Define-Range-Check-Display''

પ્રશ્ન પ્રશ્ન 3(અ OR) [03 ગુણ]

રિપીટેશન કંટ્રોલ સ્ટેટમેન્ટ શું છે? તેની યાદી બનાવો.

જવાબ

રિપીટેશન કંટ્રોલ સ્ટેટમેન્ટ:

કોષ્ટક 13. Loops

સ્ટેટમેન્ટ પ્રકાર	હેતુ
for loop	જાણીતી સંખ્યાના iterations
while loop	શરત આધારિત repetition
nested loop	લૂપની અંદર લૂપ

મુખ્ય ઘણાલો:

- Repetition statements: કોડ બ્લોક્સ વારંવાર execute કરે
- Iteration control: looping ની જુદી પદ્ધતિઓ
- Loop variables: iteration progress track કરે

મેમરી ટ્રીક

``For-While-Nested''

પ્રશ્ન પ્રશ્ન 3(બ OR) [04 ગુણ]

break અને continue સ્ટેટમેન્ટ વર્ણણો તફાવત આપો.

જવાબ

તફાવત:

કોષ્ટક 14. Break vs Continue

પાસું	break	continue
હેતુ	લૂપ સંપૂર્ણ બહાર નીકળો	વર્તમાન iteration છોડો
Execution	લૂપમાંથી બહાર jump કરે	આગલા iteration પર jump કરે
ઉપયોગ	લૂપ જદ્દી સમાપ્ત કરો	ખાસ શરતો છોડો
અસર	લૂપ સમાપ્ત થાય	લૂપ ચાલુ રહે

કોડ ઉદાહરણ:

```

1 # break example
2 for i in range(5):
3     if i == 3:
4         break
5     print(i) # Output: 0, 1, 2
6
7 # continue example
8 for i in range(5):
9     if i == 2:
10        continue
11    print(i) # Output: 0, 1, 3, 4

```

મેમરી ટ્રીક

“Break-Exit, Continue-Skip”

પ્રશ્ન પ્રશ્ન 3(ક) OR [07 ગુણ]

યુઝર ડિફાઇન ફંક્શન લખો જે 1 થી 100 સુધીની બધી બેકી સંખ્યાઓ દર્શાવે.

જવાબ

કોડ:

```

1 def display_even_numbers():
2     print("Even numbers from 1 to 100:")
3     for num in range(2, 101, 2):
4         print(num, end=" ")
5     print()
6
7 # Alternative method
8 def display_even_alt():
9     even_nums = []
10    for num in range(1, 101):
11        if num % 2 == 0:
12            even_nums.append(num)
13    print(even_nums)
14
15 # Function call
16 display_even_numbers()

```

સમજૂતી:

- કાર્યક્રમ range: બેકી સંખ્યાઓ માટે range(2, 101, 2)
- Modulus પદ્ધતિ: $\% 2 == 0$ સાથે વૈકલ્પિક ચકાસણી
- ફંક્શન ડિઝાઇન: પુનઃઉપયોગી કોડ બ્લોક

મેમરી ટ્રીક

"Range-Step-Even-Display"

પ્રશ્ન પ્રશ્ન 4(અ) [03 ગુણ]

ફુંક્શન વ્યાખ્યાયિત કરો. પાયથોનમાં ઉપલબ્ધ વિવિધ પ્રકારના ફુંક્શનની યાદી આપો.

જવાબ

ફુંક્શન: ખાસ કાર્ય કરતો પુનઃઉપયોગી કોડ બ્લોક.

ફુંક્શન પ્રકારો:

કોષ્ટક 15. Types

ફુંક્શન પ્રકાર	વર્ણન
Built-in	પૂર્વ-નિર્ધારિત ફુંક્શન-સ (print, len)
User-defined	પ્રોગ્રામર દ્વારા બનાવાયેલ
Lambda	અનામ એક-લાઇન ફુંક્શન-સ
Recursive	પોતાને call કરતા ફુંક્શન-સ

ફાયદા:

- કોડ પુનઃઉપયોગ: એકવાર લખો, ઘણીવાર વાપરો
- મોઝ્યુલારિટી: જટિલ સમસ્યાઓને નાના ભાગોમાં વહેંચવી
- Parameters: ફુંક્શન-સ માટે છન્હપુટ values

મેમરી ટ્રીક

"Built-User-Lambda-Recursive"

પ્રશ્ન પ્રશ્ન 4(બ) [04 ગુણ]

વેરિએબલના સ્કોપ ઉપર ટૂંક નોંધ લખો.

જવાબ

વેરિએબલ સ્કોપ:

કોષ્ટક 16. Scope Types

સ્કોપ પ્રકાર	વર્ણન	ઉદાહરણ
Local	ફુંક્શનની અંદર જ	ફુંક્શન variables
Global	સમગ્ર પ્રોગ્રામમાં	Module-level variables
Built-in	Python keywords	print, len, type

કોડ ઉદાહરણ:

```

1 x = 10 # Global variable
2
3 def my_function():
4     y = 20 # Local variable
5     print(x) # Access global
6     print(y) # Access local

```

```

7 my_function()
8     # print(y) # Error: y not accessible
9

```

મુખ્ય ઘણાલો:

- **Variable accessibility:** variables ક્યાં વાપરી શકાય
- **LEGB rule:** Local, Enclosing, Global, Built-in

મેમરી ટ્રીક

“Local-Global-Builtin”

પ્રશ્ન પ્રશ્ન 4(ક) [07 ગુણ]

Python કોડ લખો જે ઉપલોકતાને મુખ્ય રિસ્ટ્રિંગ અને સબરિસ્ટ્રિંગ માટે પૂછે છે અને મુખ્ય રિસ્ટ્રિંગમાં સબરિસ્ટ્રિંગની મેમ્બરશિપ તપાસે છે.

જવાબ

કોડ:

```

1 def check_substring():
2     main_string = input("Enter main string: ")
3     substring = input("Enter substring: ")
4
5     if substring in main_string:
6         print(f"{substring} found in {main_string}")
7         print(f"Position: {main_string.find(substring)}")
8     else:
9         print(f"{substring} not found in {main_string}")
10
11 # Enhanced version with case handling
12 def check_substring_enhanced():
13     main_string = input("Enter main string: ")
14     substring = input("Enter substring: ")
15
16     if substring.lower() in main_string.lower():
17         print("Substring found (case-insensitive)")
18     else:
19         print("Substring not found")
20
21 check_substring()

```

સમજૂતી:

- યુઝર ઇન્ટરેક્શન: string collection માટે input()
- **Membership testing:** in operator નો ઉપયોગ
- **Case sensitivity:** વૈકલ્પિક case handling

મેમરી ટ્રીક

“Input-Check-Report-Position”

પ્રશ્ન પ્રશ્ન 4(અ અથવા OR) [03 ગુણ]

લોકલ વેરિએબલ અને ગ્લોબલ વેરિએબલ શું છે?

જવાબ

સરખામણી:

કોષ્ટક 17. Local vs Global

વરિએબલ પ્રકાર	સ્કોપ	આયુષ્ય	પ્રવેશ
Local	ફક્ત ફંક્શનમાં	ફંક્શન execution	મર્યાદિત
Global	સમગ્ર પ્રોગ્રામ	પ્રોગ્રામ execution	વ્યાપક

ઉદાહરણ:

```

1 global_var = 100 # Global
2
3 def function():
4     local_var = 50 # Local
5     print(global_var) # Accessible
6     print(local_var) # Accessible
7
8     print(global_var) # Accessible
9     # print(local_var) # Error

```

મેમરી ટ્રીક

“Local-Limited, Global-Everywhere”

પ્રશ્ન પ્રશ્ન 4(બં OR) [04 ગુણ]

પાયથોનના કોઈપણ ચાર બિલ્ટ-ઇન ફંક્શન સમજાવો.

જવાબ

બિલ્ટ-ઇન ફંક્શન:

કોષ્ટક 18. Functions

ફંક્શન	હેતુ	ઉદાહરણ
len()	લંબાઈ આપે	len("hello") → 5
type()	ડેટા ટાઇપ આપે	type(10) → < class' int' >
input()	યુકર ઇનપુટ લે	name = input("Name: ")
print()	આઉટપુટ દર્શાવે	print("Hello")

વધારાના ઉદાહરણો:

```

1 # len() function
2 print(len([1, 2, 3, 4])) # Output: 4
3
4 # type() function
5 print(type(3.14)) # Output: <class 'float'>
6
7 # input() function
8 age = input("Enter age: ")
9
10 # print() function
11 print("Your age is:", age)

```

મેમરી ટ્રીક

"Length-Type-Input-Print"

પ્રશ્ન પ્રશ્ન 4(ક) OR [07 ગુણ]

Python કોડ લખો જે આપેલ સ્ટ્રિંગમાં સબસ્ટ્રિંગને શોધે છે.

જવાબ

કોડ:

```

1 def locate_substring():
2     main_string = input("Enter main string: ")
3     substring = input("Enter substring to find: ")
4
5     # Method 1: Using find()
6     position = main_string.find(substring)
7     if position != -1:
8         print(f"Found at index: {position}")
9     else:
10        print("Substring not found")
11
12    # Method 2: Using index() with exception handling
13    try:
14        position = main_string.index(substring)
15        print(f"Located at index: {position}")
16    except ValueError:
17        print("Substring not found")
18
19    # Method 3: Find all occurrences
20    positions = []
21    start = 0
22    while True:
23        pos = main_string.find(substring, start)
24        if pos == -1:
25            break
26        positions.append(pos)
27        start = pos + 1
28
29    if positions:
30        print(f"All positions: {positions}")
31
32 locate_substring()

```

મુખ્ય મેથ્ડ:

- **find() method:** index આપે અથવા -1
- **index() method:** index આપે અથવા exception raise કરે
- **બન્દુંધ �occurrences:** બધી સ્થિતિઓ શોધવા માટે લૂપ

મેમરી ટ્રીક

"Find-Index-Exception-Multiple"

પ્રશ્ન પ્રશ્ન 5(અ) [03 ગુણ]

સ્ટ્રિંગ વ્યાખ્યાપિત કરો. વિવિધ સ્ટ્રિંગ ઓપરેશન્સની યાદી બનાવો.

જવાબ

સ્ટ્રિંગ: quotes માં બંધ characters ની sequence.

ઓપરેશન્સ:

કોષ્ટક 19. String Operations

ઓપરેશન	મેથડ	ઉદાહરણ
સંયોજન	+	"Hello" + "World"
પુનરાવર્તન	*	"Hi" * 3
સ્લાઇસિંગ	[start:end]	"Hello"[1:4]
લંબાઈ	len()	len("Hello")
કેસ	upper(), lower()	"hello".upper()

વિશેષતાઓ:

- **Immutable:** સ્ટ્રિંગ બનાવ્યા પછી બદલી શકતી નથી
- **Indexing:** વ્યક્તિગત characters access કરવું
- **Methods:** manipulation માટે built-in functions

મેરી ટ્રીક

“Concat-Repeat-Slice-Length-Case”

પ્રશ્ન પ્રશ્ન 5(બ) [04 ગુણ]

આપણે કેવી રીતે ઓળખી શકીએ કે એલિમેન્ટ એ લિસ્ટનો સભ્ય છે કે નહીં? યોગ્ય ઉદાહરણ સાથે સમજાવો.

જવાબ

પદ્ધતિઓ:

કોષ્ટક 20. Membership Check

પદ્ધતિ	ઓપરેટર	પરિણામ
in	element in list	True/False
not in	element not in list	True/False
count()	list.count(element)	occurrences ની સંખ્યા

ઉદાહરણ:

```

1 fruits = ["apple", "banana", "orange", "mango"]
2
3 # Using 'in' operator
4 if "apple" in fruits:
5     print("Apple is available")
6
7 # Using 'not in' operator
8 if "grapes" not in fruits:
9     print("Grapes not available")
10
11 # Using count() method
12 count = fruits.count("apple")

```

```

13 if count > 0:
14     print(f"Apple found {count} times")

```

મેમરી ટ્રીક**“In-NotIn-Count for membership”****પ્રશ્ન પ્રશ્ન 5(ક) [07 ગુણ]**

Python કોડ લખો જે આપેલ સ્ટ્રિંગના બીજા સબસ્ટ્રિંગ સાથે સબસ્ટ્રિંગને બદલે છે. આપેલ સ્ટ્રિંગ 'Welcome to GTU' તરીકે ધ્યાનમાં લો અને સબસ્ટ્રિંગ 'GTU' ને 'Gujarat Technological University' સાથે બદલો.

જવાબ**કોડ:**

```

1 def replace_substring():
2     # Given string
3     original = "Welcome to GTU"
4     old_substring = "GTU"
5     new_substring = "Gujarat Technological University"
6
7     # Method 1: Using replace()
8     result1 = original.replace(old_substring, new_substring)
9     print(f"Original: {original}")
10    print(f"Modified: {result1}")
11
12    # Method 2: Manual replacement
13    if old_substring in original:
14        index = original.find(old_substring)
15        result2 = original[:index] + new_substring + original[index + len(old_substring):]
16        print(f"Manual method: {result2}")
17
18    # Method 3: Replace all occurrences
19    test_string = "GTU offers GTU degree from GTU"
20    result3 = test_string.replace("GTU", "Gujarat Technological University")
21    print(f"Multiple replacements: {result3}")
22
23 replace_substring()

```

આઉટપુટ:

Original: Welcome to GTU

Modified: Welcome to Gujarat Technological University

મુખ્ય મુદ્દાઓ:

- replace() method:** built-in string function
- Slicing method:** મેન્યુઅલ string manipulation
- બધી occurrences:** દરેક instance બદલે છે

મેમરી ટ્રીક**“Find-Replace-Slice-All”**

પ્રશ્ન પ્રશ્ન 5(અ OR) [03 ગુણ]

લિસ્ટ વ્યાખ્યાપિત કરો. વિવિધ લિસ્ટ ઓપરેશન્સની યાદી બનાવો.

જવાબ

લિસ્ટ: કમબદ્ધ items નો collection જે modify કરી શકાય છે.

ઓપરેશન્સ:

કોષ્ટક 21. List Operations

ઓપરેશન	મેથડ	ઉદાહરણ
ઉમેરો	append(), insert()	list.append(item)
ફૂર કરો	remove(), pop()	list.remove(item)
પ્રવેશ	[index]	list[0]
સ્લાઇસ	[start:end]	list[1:3]
સોર્ટ	sort()	list.sort()

વિશેષતાઓ:

- Mutable:** લિસ્ટ બનાવ્યા પછી બદલી શકાય છે
- Indexed:** સ્થિતિ દ્વારા elements access કરાય છે
- Dynamic:** કદ વધી અથવા ઘટી શકે છે

મેરી ટ્રીક

``Add-Remove-Access-Slice-Sort''

પ્રશ્ન પ્રશ્ન 5(બ OR) [04 ગુણ]

સ્ટ્રિંગ સ્લાઇસિંગ ઉપર ટૂંક નોંધ લખો. ચોગ્ય ઉદાહરણ સાથે સમજાવો.

જવાબ

સ્ટ્રિંગ સ્લાઇસિંગ: [start:end:step] વાપરીને string ના ભાગો extract કરવું.

Syntax:

કોષ્ટક 22. Slicing Syntax

Syntax	વર્ણન	ઉદાહરણ
[start:]	start થી અંત સુધી	"Hello"[1:] → "ello"
[:end]	શરૂઆત થી end સુધી	"Hello":[3] → "Hel"
[start:end]	ખાસ રેન્જ	"Hello"[1:4] → "ell"
[::-1]	રિવર્સ સ્ટ્રિંગ	"Hello"[::-1] → "olleH"

ઉદાહરણ:

```

1 text = "Python Programming"
2
3 print(text[0:6]) # "Python"
4 print(text[7:]) # "Programming"
5 print(text[:6]) # "Python"
6 print(text[::-2]) # "Pto rgamn"
7 print(text[::-1]) # "gnimmargorP nohtyP"

```

મેમરી ટ્રીક

"Start-End-Step-Reverse"