

પાયથોન પ્રોગ્રામિંગ (1323203) - ઉનાળુ 2024 સોલ્યુશન

Milav Dabgar

જૂન 18, 2024

પ્રશ્ન 1(અ) [3 ગુણ]

ફ્લોચાર્ટ અને અલ્ગોરિધમના મહત્વની યાદી આપો.

જવાબ

કોષ્ટક 1. ફ્લોચાર્ટ અને અલ્ગોરિધમનું મહત્વ

ફ્લોચાર્ટનું મહત્વ	અલ્ગોરિધમનું મહત્વ
પ્રોગ્રામ લોજિકનું દૃશ્ય નિરૂપણ	સમસ્યાને ઉકેલવા માટેનું પગલાંવાર પ્રક્રિયા
ભૂલોને સરળતાથી શોધવા અને સુધારવા	ભાષાથી સ્વતંત્ર ઉકેલ અભિગમ
જટિલ પ્રક્રિયાઓને સમજવામાં મદદ	પ્રોગ્રામિંગની પાયારુપ શરૂઆત
ટીમના સભ્યો વચ્ચે સંદેશાવ્યવહાર સુધારે	કોડિંગ શરૂ કરતા પહેલા લોજિક નિર્ધારિત કરે

મેમરી ટ્રીક

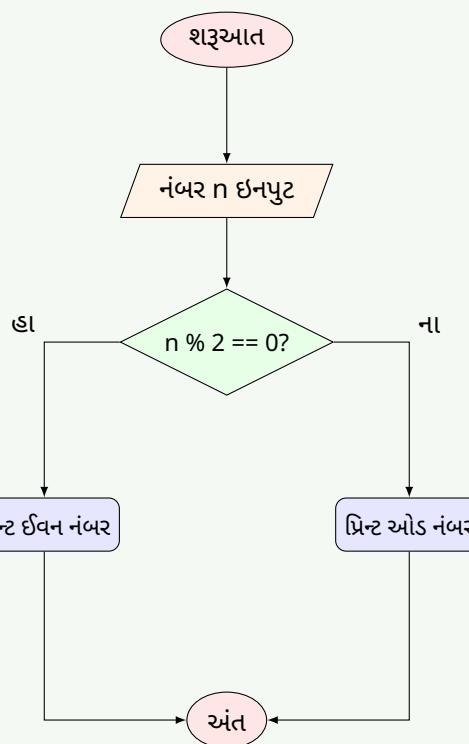
“VASE Decisions” - Visualize, Analyze, Sequence, Execute

પ્રશ્ન 1(બ) [4 ગુણ]

દાખલ કરેલ સંખ્યા ઈવન કે ઓડ છે તે શોધવા માટે ફ્લોચાર્ટ દોરો.

જવાબ

આકૃતિ 1. ઈવન કે ઓડ સંખ્યા તપાસવા માટે ફ્લોચાર્ટ



મુખ્ય પગલાં:

- ડેટા એક્સ્ટ્રાક્શન: વપરાશકર્તા પાસેથી નંબર મેળવો
- મોડ્યુલો ઓપરેશન: 2 વડે ભાગીને શેષ તપાસો
- શરતી આઉટપુટ: શેષના આધારે પરિણામ દર્શાવો

મેમરી ટ્રીક

“MODE” - Modulo Operation Determines Evenness

પ્રશ્ન 1(ક) [7 ગુણ]

બધા લોજિકલ ઓપરેટરોની ચાદી બનાવો અને પાયથોન કોડનું ઉદાહરણ આપીને દરેકને સમજાવો.

જવાબ

કોષ્ટક 2. લોજિકલ ઓપરેટરો

ઓપરેટર	વર્ણન	ઉદાહરણ	આઉટપુટ
and	બંને સ્ટેટમેન્ટ સાચા હોય તો True રિટર્ન કરે	x = 5; print(x > 3 and x < 10)	True
or	બે સ્ટેટમેન્ટમાંથી એક સાચું હોય તો True રિટર્ન કરે	x = 5; print(x > 10 or x == 5)	True
not	પરિણામને ઉલટાવે, જો પરિણામ સાચું હોય તો False રિટર્ન કરે	x = 5; print(not(x > 3))	False

કોડ ઉદાહરણ:

```

1 # લોજિકલ AND ઉદાહરણ
2 age = 25
3 income = 50000
4 print("Loan eligibility:", age > 18 and income > 30000) # True
5
6 # લોજિકલ OR ઉદાહરણ
7 has_credit_card = False
  
```

```

8 has_cash = True
9 print("Can purchase:", has_credit_card or has_cash) # True
10
11 # લોજિકિલ NOT ઉદાહરણ
12 is_holiday = False
13 print("Should work today:", not is_holiday) # True

```

મેમરી ટ્રીક

``AON Clarity'' - And, Or, Not for logical clarity

OR

પ્રશ્ન 1(ક) [7 ગુણ]

એક પાયથોન પ્રોગ્રામ લખો કરો જે આપેલ ડેટા પર સાદા વ્યાજ અને ચક્રવૃદ્ધિ વ્યાજની ગણતરી કરી શકે.

જવાબ

```

1 # સાદા અને ચક્રવૃદ્ધિવ્યાજની ગણતરી માટેનો પ્રોગ્રામ
2
3 # ઈનપુટ મૂલ્યો
4 principal = float(input("Enter principal amount: "))
5 rate = float(input("Enter rate of interest (in %): "))
6 time = float(input("Enter time period (in years): "))
7
8 # સાદા વ્યાજની ગણતરી
9 simple_interest = (principal * rate * time) / 100
10
11 # ચક્રવૃદ્ધિવ્યાજની ગણતરી
12 compound_interest = principal * ((1 + rate/100) ** time - 1)
13
14 # પરિણામો દર્શાવો
15 print("Simple Interest:", round(simple_interest, 2))
16 print("Compound Interest:", round(compound_interest, 2))

```

રાસાયણિક સમીકરણ/સૂત્ર

- સાદું વ્યાજ (SI): મૂળ રકમ \times દર \times સમય / 100
- ચક્રવૃદ્ધિ વ્યાજ (CI): મૂળ રકમ \times $((1 + દર/100)^{\text{સમય}} - 1)$

મેમરી ટ્રીક

``PRT Money Grows'' - Principal, Rate, Time make money grow

પ્રશ્ન 2(અ) [3 ગુણ]

આપેલ ત્રણ નંબરોમાંથી ન્યૂનતમ સંખ્યા શોધવા માટે પાયથોન પ્રોગ્રામ બનાવો.

જવાબ

```

1 # તુરણ નંબરોમાંથી ન્યૂનતમ શોધવાનો પ્રોગ્રામ
2
3 # તુરણ નંબર ઇનપુટ લો
4 num1 = float(input("Enter first number: "))
5 num2 = float(input("Enter second number: "))
6 num3 = float(input("Enter third number: "))
7
8 # બલિટઇન- min() ફંક્શનનો ઉપયોગ કરીને ન્યૂનતમ શોધો
9 minimum = min(num1, num2, num3)
10
11 # પરિણામ દર્શાવો
12 print("Minimum number is:", minimum)

```

મેમરી ટ્રીક

“MIN Finds Least” - Minimum Is Numerically Found with Least

પ્રશ્ન 2(બ) [4 ગુણ]

સ્યુડોકોડ વ્યાખ્યાયિત કરો. x, y અને z ત્રણમાંથી સૌથી મોટી સંખ્યા શોધવા માટે સ્યુડોકોડ લખો.

જવાબ

કોષ્ટક 3. સ્યુડોકોડની વ્યાખ્યા

વ્યાખ્યા

કમ્પ્યુટર પ્રોગ્રામે શું કરવું જોઈએ તેનું વિગતવાર અને વાંચી શકાય તેવું વર્ણન, જે પ્રોગ્રામિંગ ભાષાને બદલે ઓપચારિક શૈલીમાં લખાયેલો કુદરતી ભાષામાં વ્યક્ત કરવામાં આવે છે.

ત્રણ નંબરોમાંથી સૌથી મોટો શોધવા માટે સ્યુડોકોડ:

```

1 BEGIN
2   INPUT x, y, z
3   SET largest = x
4
5   IF y > largest THEN
6     SET largest = y
7   END IF
8
9   IF z > largest THEN
10    SET largest = z
11  END IF
12
13  OUTPUT "Largest number is: ", largest
14 END

```

મેમરી ટ્રીક

“PIE Writing” - Program Ideas Expressed in simple writing

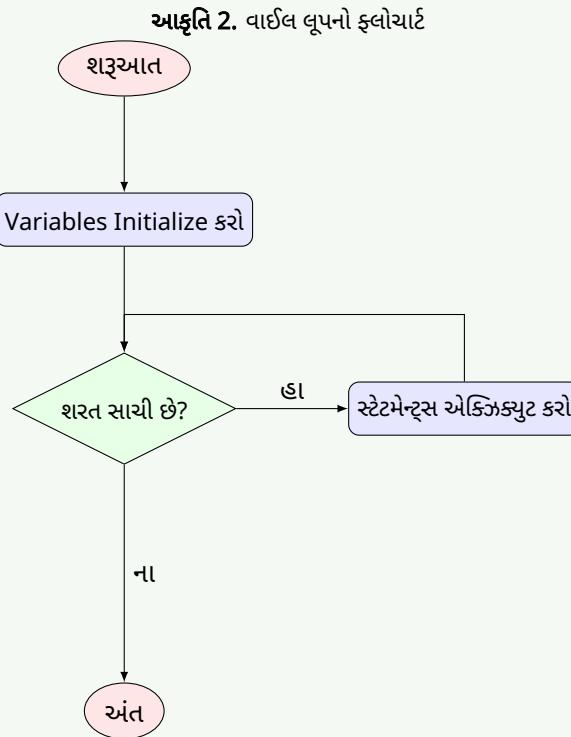
પ્રક્રિયા 2(ક) [7 ગુણ]

પાયથોનમાં વાઈલ લૂપને તેના સિન્ટેક્સ, ફ્લોચાર્ટ અને પાયથોન કોડના ઉદાહરણ સાથે સમજાવો.

જવાબ

સિન્ટેક્સ:

```
1 while condition:  
2     # code to be executed
```



કોડ ઉદાહરણ:

```
1 # પ્રથમ 5 કુદરતી સંખ્યાઓ while લૂપનો ઉપયોગ કરીને પ્રણિત કરો
2 count = 1
3
4 while count <= 5:
5     print(count)
6     count += 1 # કાઉન્ટર વધારો
7
8 # આઉટપુટ:
9 # 1
10 # 2
11 # 3
12 # 4
13 # 5
```

મુખ્ય લક્ષણો:

- એટ્રી કંટ્રોલ: લૂપ એક્ઝિક્યુશન પહેલાં શરત ચકાસવામાં આવે છે
- ઇનિશિયલાઇઝન: લૂપ પહેલાં વેરિએબલ્સ સેટ કરવામાં આવે છે
- અપડેશન: લૂપની અંદર વેરિએબલ્સ અપડેટ કરવામાં આવે છે
- ટર્મિનેશન: શરત ખોટી થાય ત્યારે લૂપ બહાર નીકળે છે

મેમરી ટ્રીક

``IUTE Loop'' - Initialize, Update, Test for Exit

OR

પ્રશ્ન 2(અ) [3 ગુણ]

પાયથોળમાં કન્ટિન્યુ સ્ટેટમેન્ટનું ટૂંકમાં વર્ણન કરો.

જવાબ

કોષ્ટક 4. કન્ટિન્યુ સ્ટેટમેન્ટ

વર્ણન
કન્ટિન્યુ સ્ટેટમેન્ટ લૂપના વર્તમાન ઇટરેશનને છોડી દે છે અને આગામા ઇટરેશનથી ચાલુ રાખે છે
જ્યારે એનકાઉન્ટર થાય, ત્યારે કન્ટિન્યુ સ્ટેટમેન્ટ પછીનો લૂપનો કોડ છોડી દેવામાં આવે છે
ચોક્કસ શરતોને છોડીને લૂપને ચાલુ રાખવા માટે ઉપયોગી છે

કોડ ઉદાહરણઃ

```

1 # બેકી સંખ્યાઓ પ્રનિટ કરવાનું છોડી દો
2 for i in range(1, 6):
3     if i % 2 == 0:
4         continue
5     print(i) # માત્ર 1, 3, 5 પ્રનિટ થાય

```

મેમરી ટ્રીક

``SKIP Ahead'' - Skip Keeping Iteration Process

OR

પ્રશ્ન 2(બ) [4 ગુણ]

નીચેના કોડનું આઉટપુટ શું હશે?

જવાબ

કોડઃ

```

1 x=8
2 y=2
3 print(x*y)
4 print(x ** y)
5 print(x % y)
6 print(x>y)

```

કોષ્ટક 5. આઉટપુટ વિશ્લેષણ

ઓપરેશન	પરિણામ	સમજૂતી
$x*y$	16	ગુણાકાર: $8 \times 2 = 16$
$x**y$	64	પાવર: $8^2 = 64$
$x%y$	0	મોડ્યુલો (શેષ): $8 \div 2 = 4$ શેષ 0
$x>y$	True	તુલના: $8 > 2$ સાચું છે

મેમરી ટ્રીક

“MEMO” - Multiply, Exponent, Modulo, Operator comparison

OR

પ્રશ્ન 2(ક) [7 ગુણ]

પાયથોનમાં ઈફ-ઇએલઇએફ-એલ્સ લેડરને તેના સિન્ટેક્સ, ફ્લોચાર્ટ અને પાયથોન કોડના ઉદાહરણ સાથે સમજાવો.

જવાબ

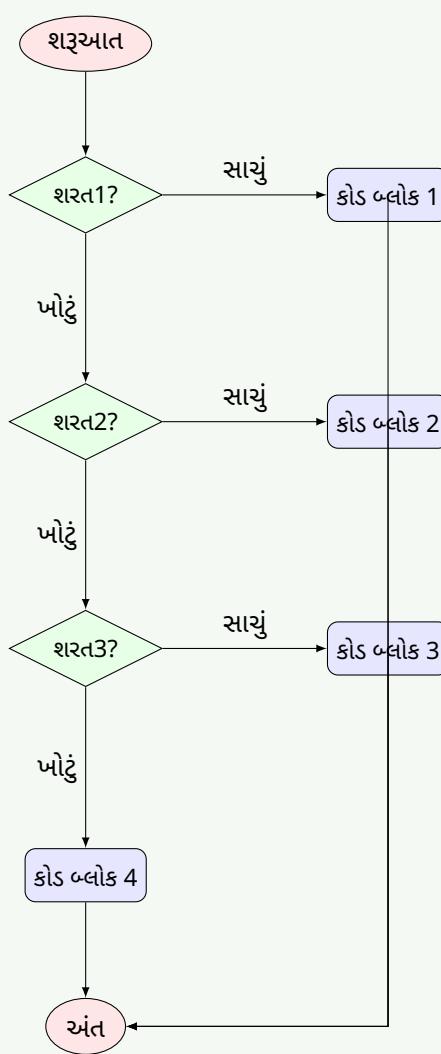
સિન્ટેક્સ:

```

1 if condition1:
2     # code block 1
3 elif condition2:
4     # code block 2
5 elif condition3:
6     # code block 3
7 else:
8     # code block 4

```

આકૃતિ 3. ઈફ-ઇએલઇએફ-એલ્સ લેડરનો ફ્લોચાર્ટ



કોડ ઉદાહરણ:

```

1 # માર્ક્સના આધારે ગ્રેડની ગણતરી
2 marks = 75
3
4 if marks >= 90:
5     grade = "A+"
6 elif marks >= 80:
7     grade = "A"
8 elif marks >= 70:
9     grade = "B"
10 elif marks >= 60:
11     grade = "C"
12 else:
13     grade = "D"
14
15 print("Grade:", grade) # આઉટપુટ: Grade: B
  
```

મુખ્ય લક્ષણો:

- અનુક્રમિક મૂલ્યાંકન: શરતો ઉપરથી નીચે તપાસવામાં આવે છે
- અનન્ય એક્ઝિક્યુશન: માત્ર એક બ્લોક એક્ઝિક્યુટ થાય છે
- ડિફોન્ડ એક્શન: જો કોઈ શરત સાચી ન હોય તો else બ્લોક એક્ઝિક્યુટ થાય છે

મેમરી ટ્રીક

“SEEP Logic” - Sequential Evaluation with Exclusive Path

પ્રશ્ન 3(અ) [3 ગુણ]

લૂપનો ઉપયોગ કરીને 1 થી 20 વચ્ચેની એકી સંખ્યાઓ છાપવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

```

1 # 1 થી 20 વચ્ચેની એકી સંખ્યાઓ છાપવાનો પ્રોગ્રામ
2
3 # range અને step સાથે for લૂપનો ઉપયોગ
4 for number in range(1, 21, 2):
5     print(number, end=" ")
6
7 # આઉટપુટ: 1 3 5 7 9 11 13 15 17 19

```

વૈકલ્પિક અભિગમ:

```

1 # if શરત સાથે for લૂપનો ઉપયોગ
2 for number in range(1, 21):
3     if number % 2 != 0:
4         print(number, end=" ")

```

મેમરી ટ્રીક

“STEO” - Skip Two, Extract Odds

પ્રશ્ન 3(બ) [4 ગુણ]

નેર્સ્ટેડ ઈફ સ્ટેટમેન્ટને સંક્ષિપ્તમાં સમજાવો.

જવાબ

કોષ્ટક 6. નેર્સ્ટેડ ઈફ સ્ટેટમેન્ટ

વર્ણન
બીજા if સ્ટેટમેન્ટની અંદર એક if સ્ટેટમેન્ટ
વધુ જટિલ શરતી લોજિકની મંજૂરી આપે છે
બાહ્ય if સાચું હોય ત્યારે જ આંતરિક if મૂલ્યાંકન કરવામાં આવે છે
નેસ્ટિંગના ઘણા સ્તરો હોઈ શકે છે

કોડ ઉદાહરણ:

```

1 age = 25
2 income = 50000
3
4 if age > 18:
5     print("પુખ્ફૂત")
6     if income > 30000:
7         print("કરેડટ કાર્ડ માટે પાત્ર")
8     else:

```

```

9     print("કુરેડટિ કાર્ડ માટે અપાત્ર")
10    else:
11        print("સગીર")

```

મેમરી ટ્રીક

``LION'' - Layered If-statements Operating Nested

પ્રશ્ન 3(ક) [૭ ગુણા]

યુઝર ડિફેન્ડિન ફુંક્શનનો ઉપયોગ કરીને પ્રોગ્રામ લખો જે દાખલ કરેલ નંબર 'આર્મ્સ્ટ્રોંગ' નંબર અથવા પેલનિંડ્રોમ છે તે તપાસવા માટે પ્રોગ્રામ લખો એ જેમાં કોલિંગ ફુંક્શનમાં આગ્યુર્મેટ તરીકે નંબર આપવામા આવે છે.

જવાબ

```

1 # આર્મ્સ્ટ્રોંગ નંબર અથવા પેલનિંડ્રોમ તપાસવાનો પ્રોગ્રામ
2
3 def check_number(num):
4     # આર્મ્સ્ટ્રોંગ નંબર તપાસો
5     temp = num
6     digits = len(str(num))
7     sum = 0
8
9     while temp > 0:
10         digit = temp % 10
11         sum += digit ** digits
12         temp //= 10
13
14     is_armstrong = (sum == num)
15
16     # પેલનિંડ્રોમ તપાસો
17     is_palindrome = (str(num) == str(num)[::-1])
18
19     # પરિણામો પાછા આપો
20     return is_armstrong, is_palindrome
21
22 # વપરાશકરૂતા પાસેથી ઇનપુટ લો
23 number = int(input("એક નંબર દાખલ કરો: "))
24
25 # ફુંક્શન કોલ કરો અને પરિણામો દર્શાવો
26 armstrong, palindrome = check_number(number)
27
28 if armstrong:
29     print(number, "is an Armstrong number")
30 else:
31     print(number, "is not an Armstrong number")
32
33 if palindrome:
34     print(number, "is a Palindrome")
35 else:
36     print(number, "is not a Palindrome")

```

ઉદાહરણો:

- આર્મ્સ્ટ્રોંગ: 153 ($1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$)
- પેલનિંડ્રોમ: 121 (આગામ અને પાછા સમાન)

મેમરી ટ્રીક

``APTEST'' - Armstrong Palindrome Test Equal Sum Test

OR

પ્રશ્ન 3(અ) [3 ગુણ]

૧ થી ૧૦૦ સુધી નો સરવાળો શોધવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

```

1 # 1 થી 100 સુધીની સંખ્યાઓનો સરવાળો શોધવાનો પ્રોગ્રામ
2
3 # પદ્ધતિ1: લૂપનો ઉપયોગ
4 total = 0
5 for num in range(1, 101):
6     total += num
7     print("Sum using loop:", total)
8
9 # પદ્ધતિ2: સૂત્ર  $n(n+1)/2$  નો ઉપયોગ
10 n = 100
11 sum_formula = n * (n + 1) // 2
12 print("Sum using formula:", sum_formula)
13
14 # આઉટપુટ:
15 # Sum using loop: 5050
16 # Sum using formula: 5050

```

મેમરી ટ્રીક

``SUM Formula'' - Sum Using Mathematical Formula

OR

પ્રશ્ન 3(બ) [4 ગુણ]

નીચેની પેટન છાપવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

પેટન:

1
2 3
4 5 6
7 8 9 10

કોડ:

```

1 # સંખ્યા પેટન છાપવાનો પ્રોગ્રામ
2
3 num = 1
4 for i in range(1, 5): # 4 પંકૃતાઓ
5     for j in range(i): # પંકૃતનિંબર જેટલા કોલમ

```

```

6     print(num, end=" ")
7     num += 1
8 print() # દરેક પંકૃતિપછી નવી લાઈન

```

પેટન્ લોજિક:

- પંક્તિ 1: 1 સંખ્યા (1)
- પંક્તિ 2: 2 સંખ્યાઓ (2, 3)
- પંક્તિ 3: 3 સંખ્યાઓ (4, 5, 6)
- પંક્તિ 4: 4 સંખ્યાઓ (7, 8, 9, 10)

મેમરી ટ્રીક

“CNIR” - Counter Number Increases with Rows

OR

પ્રશ્ન 3(ક) [૭ ગુણ]

ફુલશનનો ઉપયોગ કરીને પ્રોગ્રામ લખો જે દાખલ કરેલ નંબરને ઉલટાવે

જવાબ

```

1 # દાખલ કરેલ મૂલ્યને ઉલટાવવા માટે ફુલશન ઉપયોગ કરતો પ્રોગ્રામ
2
3 def reverse_number(num):
4     """સંખ્યાને ઉલટાવવા માટેનું ફુલશન"""
5     return int(str(num)[::-1])
6
7 def reverse_string(text):
8     """સૂત્રાળિને ઉલટાવવા માટેનું ફુલશન"""
9     return text[::-1]
10
11 # મુખ્ય પ્રોગ્રામ
12 def main():
13     choice = input("What do you want to reverse? (n for number, s for string): ")
14
15     if choice.lower() == 'n':
16         num = int(input("Enter a number: "))
17         print("Reversed number:", reverse_number(num))
18     elif choice.lower() == 's':
19         text = input("Enter a string: ")
20         print("Reversed string:", reverse_string(text))
21     else:
22         print("Invalid choice!")
23
24 # મુખ્ય ફુલશન કોલ કરો
25 main()

```

નંબર ઉલટાવવા માટે વૈકલ્પિક પદ્ધતિ:

```

1 def reverse_number_algorithm(num):
2     reversed_num = 0
3     while num > 0:
4         digit = num % 10
5         reversed_num = reversed_num * 10 + digit
6         num //= 10
7     return reversed_num

```

મેમરી ટ્રીક

``FLIP Digits'' - Function Logic Inverts Position of Digits

પ્રક્રિયા 4(અ) [3 ગુણ]

યોગ્ય પાયથોન કોડ ઉદાહરણ સાથે પાયથોન મેથ મોડ્યુલનું વર્ણન કરો.

જવાબ

કોષ્ટક 7. પાયથોન મેથ મોડ્યુલ

વિશેષતાઓ
ગાણિતિક ફંક્શન્સ અને સ્થિરાંકો પ્રદાન કરે છે
નિકોણમિત્ય, લોગરિધમિક અને અન્ય ફંક્શન્સ શામેલ છે
pi અને e જેવા ગાણિતિક સ્થિરાંકો ધરાવે છે
ઉપયોગ કરતા પહેલાં import કરવું જરૂરી છે

કોડ ઉદાહરણ:

```

1 import math
2
3 # સ્થિરાંકો
4 print("Value of pi:", math.pi) # 3.141592653589793
5 print("Value of e:", math.e) # 2.718281828459045
6
7 # મૂળભૂત ગાણિતિક ફંક્શન્સ
8 print("16 નો વર્ગમૂળ:", math.sqrt(16)) # 4.0
9 print("5 ની ઘાત 3:", math.pow(5, 3)) # 125.0
10
11 # ત્રિકોણમત્તિય ફંક્શન્સ રેહિન()
12 print("90° નો સાઇન:", math.sin(math.pi/2)) # 1.0
13 print("0° નો કોસાઇન:", math.cos(0)) # 1.0
14
15 # લોગરથ્મિક ફંક્શન્સ
16 print("100 નો આધાર 10 લોગ:", math.log10(100)) # 2.0
17 print("e નો નેચરલ લોગ:", math.log(math.e)) # 1.0

```

મેમરી ટ્રીક

``CALM Operations'' - Constants And Logarithmic Mathematical Operations

પ્રક્રિયા 4(બ) [4 ગુણ]

વેરીએબલના સ્કોપને સમજાવતો પાયથોન પ્રોગ્રામ લખો.

જવાબ

```

1 # પાયથોનમાં વેરીએબલ સ્કોપ દર્શાવતો પ્રોગ્રામ
2
3 # ગ્લોબલ વેરીએબલ
4 global_var = "I am global"
5

```

```

6 def demonstration():
7     # લોકલ વેરીએબલ
8     local_var = "I am local"
9
10    # ગ્લોબલ વેરીએબલ એક્સેસ કરવું
11    print("Inside function - Global variable:", global_var)
12
13    # લોકલ વેરીએબલ એક્સેસ કરવું
14    print("Inside function - Local variable:", local_var)
15
16    # ગ્લોબલ નામ ધરાવતું લોકલ વેરીએબલ બનાવવું
17    global_var = "I am local with global name"
18    print("Inside function - Shadowed global:", global_var)
19
20    # ફંક્શન કોલ
21    demonstration()
22
23    # ગ્લોબલ વેરીએબલ એક્સેસ કરવું
24    print("Outside function - Global variable:", global_var)
25
26    # લોકલ વેરીએબલ એક્સેસ કરવાનો પ્રયાસ ભૂલ ઉત્પન્ન કરશે
27    # print("Outside function - Local variable:", local_var) # Error!

```

મેમરી ટ્રીક

“GLOVES” - Global Local Variable Encapsulation System

પ્રશ્ન 4(ક) [૭ ગુણ]

લિસ્ટ પદ્ધતિઓ અને તેના બિલ્ટ-ઇન કાચો સમજાવો

જવાબ

કોષ્ટક 8. લિસ્ટ પદ્ધતિઓ અને ફંક્શન્સ

પદ્ધતિ	વર્ણન	ઉદાહરણ	આઉટપુટ
append()	અંતે એલિમેન્ટ ઉમેરે છે	I=['a']; I.append('b')	['a', 'b']
insert()	ચોક્કસ પોઝિશન પર એલિમેન્ટ ઉમેરે	I=[1,3]; I.insert(1,2)	[1, 2, 3]
remove()	ચોક્કસ આઈટમ દૂર કરે	I=['r','b']; I.remove('r')	['b']
pop()	ચોક્કસ ઇન્ડેક્સ પર આઈટમ દૂર કરે	I=['a','b']; I.pop(1)	'b'
clear()	બધા એલિમેન્ટ્સ દૂર કરે	I=[1,2]; I.clear()	[]
len()	એલિમેન્ટ્સની સંખ્યા પાછી આપે	len([1, 2, 3])	3
sorted()	સોર્ટ લિસ્ટ પાછી આપે	sorted([3, 1, 2])	[1, 2, 3]
max()	મહત્તમ મૂલ્ય પાછું આપે	max([5, 10, 3])	10

કોડ ઉદાહરણ:

```

1 # લિસ્ટ બનાવવી
2 my_list = [3, 1, 4, 1, 5]
3 my_list.append(9)      # અંતે ઉમેરો
4 my_list.insert(2, 7)   # ઇન્ડેક્સ 2 પર ઉમેરો
5 my_list.remove(1)     # પ્રથમ 1 દૂર કરો
6 popped = my_list.pop() # છેલ્લું એલિમેન્ટ દૂર કરો
7
8 print("લંબાઈ:", len(my_list))

```

```

9 print("સોર્ટેડ:", sorted(my_list))
10 print("સરવાળો:", sum(my_list))
11 print("1 ની સંખ્યા:", my_list.count(1))

```

મેમરી ટ્રીક**“LISP Operations” - List Insert Sort Pop Operations**

OR

પ્રશ્ન 4(અ) [3 ગુણ]

પાયથોન સ્ટાન્ડર્ડ લાઇબ્રેરી ગાણિતિક કાચોની સૂચિ બનાવો.

જવાબ**કોષ્ટક 9. ગાણિતિક ફંક્શન્સ**

ગાણિતિક ફંક્શન	વર્ણન	ઉદાહરણ
abs()	નિરપેક્ષ મૂલ્ય પાછું આપે	abs(-5) → 5
round()	નજીકના પૂર્ણાંક સુધી ગોળ કરે	round(3.7) → 4
max()	સૌથી મોટો આઈટમ પાછી આપે	max(1, 5) → 5
min()	સૌથી નાની આઈટમ પાછી આપે	min(1, 5) → 1
sum()	ઇટેબલની આઈટમ્સનો સરવાળો કરે	sum([1, 2]) → 3
pow()	x ને y ની ઘાત પાછી આપે	pow(2, 3) → 8

math મોડ્યુલમાંથી વધારાના:

- math.sqrt(): વર્ગમૂળ
- math.floor(): નીચે ગોળ કરે
- math.ceil(): ઉપર ગોળ કરે
- math.factorial(): ફેક્ટોરીયલ
- math.gcd(): મહત્તમ સામાન્ય અવયવ

મેમરી ટ્રીક**“SMART Calculations” - Standard Mathematical Arithmetic Routines and Tools**

OR

પ્રશ્ન 4(બ) [4 ગુણ]

પાયથોનમાં બિલ્ટ ઇન ફંક્શન સમજાવો.

જવાબ**કોષ્ટક 10. બિલ્ટ-ઇન ફંક્શન્સ**

વર્ણન

કોઈપણ મૌજુલ ઇમ્પોર્ટ કર્યા વિના પાયથોનમાં ઉપલબ્ધ પ્રી-ડિફાઇન ફંક્શન્સ
કોઈપણ પ્રોફિક્સ વિના સીધા જ કોલ કરી શકાય છે
સામાન્ય ઓપરેશન્સ કરવા માટે ડિજાઇન કરેલ છે
ઉદાહરણોમાં print(), len(), type(), input(), range() શામેલ છે

ક્રેટારીઓ સાથે ઉદાહરણો:

```

1 # ટાઇપ કન્ટ્રોલ
2 print(int("10"))    # 10
3 print(str(10))     # "10"
4
5 # ગાણિતિક ફંક્શન્સ
6 print(abs(-7))      # 7
7 print(max(5, 10, 3)) # 10
8
9 # કલેક્શન પ્રોસેસની
10 print(len("hello")) # 5
11 print(sorted([3,1,2])) # [1, 2, 3]

```

મેમરી ટ્રીક

“EPIC Functions” - Embedded Python Integrated Core Functions

OR

પ્રશ્ન 4(ક) [૭ ગુણા]

વાક્યમાં રહેલ સ્વરો, વ્યંજન, અપરકેસ, લોઅરકેસ અક્ષરોની સંખ્યા ગણવા અને દર્શાવવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

```

1 # સૂટરગિમાં સ્વરો, વ્યંજન, અપરકેસ, લોઅરકેસ ગણતરી માટેનો પ્રોગ્રામ
2
3 def analyze_string(text):
4     # કાઉન્ટર્સ ઇનશિયલાઇઝ કરો
5     vowels = 0
6     consonants = 0
7     uppercase = 0
8     lowercase = 0
9
10    # સ્વરો ડફિન કરો
11    vowel_set = {'ા', 'એ', 'િ', 'ો', 'ુ'}
12
13    # દરેક અક્ષરનું વણીલેખણ
14    for char in text:
15        # તપાસો કે શું અક્ષર છે
16        if char.isalpha():
17            # કેસ તપાસો
18            if char.isupper():
19                uppercase += 1
20            else:
21                lowercase += 1
22
23    # તપાસો કે સ્વર છે કેસસેન્સટિવિ(-)

```

```

24     if char.lower() in vowel_set:
25         vowels += 1
26     else:
27         consonants += 1
28
29     # પરિણામો પાછા આપો
30     return vowels, consonants, uppercase, lowercase
31
32 # ઇનપુટ લો
33 text = input("Enter a string: ")
34
35 # ગણતરી મેળવો
36 vowels, consonants, uppercase, lowercase = analyze_string(text)
37
38 # પરિણામો દરેકાંથી
39 print("Number of vowels:", vowels)
40 print("Number of consonants:", consonants)
41 print("Number of uppercase characters:", uppercase)
42 print("Number of lowercase characters:", lowercase)

```

મેમરી ટ્રીક

“VOCAL Analysis” - Vowels Or Consonants And Letter case

પ્રશ્ન 5(અ) [3 ગુણ]

લિસ્ટ માં આપેલ બે એલીમેન્ટ ને સ્વેપ કરવા માટે પાયથોન કોડ લખો.

જવાબ

```

1 # લસ્ટમાં બે એલીમેન્ટ્સ સ્વેપ કરવાનો પ્રોગ્રામ
2
3 def swap_elements(lst, pos1, pos2):
4     """લસ્ટમાં બે એલીમેન્ટ્સ સ્વેપ કરવા માટેનું ફક્કુશન"""
5     lst[pos1], lst[pos2] = lst[pos2], lst[pos1]
6     return lst
7
8 # ઉદાહરણ ઉપરોગ
9 my_list = [10, 20, 30, 40, 50]
10 print("મૂળ લસ્ટ: ", my_list)
11
12 # પોઝિશન 1 અને 3 પરના એલીમેન્ટ્સ સ્વેપ કરો
13 result = swap_elements(my_list, 1, 3)
14 print("પોઝિશન 1 અને 3 પરના એલીમેન્ટ્સ સ્વેપ કર્યા પછી: ", result)
15
16 # આઉટપુટ:
17 # મૂળ લસ્ટ: [10, 20, 30, 40, 50]
18 # પોઝિશન 1 અને 3 પરના એલીમેન્ટ્સ સ્વેપ કર્યા પછી: [10, 40, 30, 20, 50]

```

મેમરી ટ્રીક

“STEP Logic” - Swap Two Elements with Python Logic

પ્રશ્ન 5(બ) [4 ગુણ]

આપેલ સ્ટ્રિંગમાં સબસ્ટ્રિંગ હાજર છે કે કેમ તે તપાસવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

```

1 # સ્ટ્રાંગમાં સબસ્ટ્રાંગની હાજરી તપાસવાનો પ્રોગ્રામ
2
3 def check_substring(main_string, sub_string):
4     """સ્ટ્રાંગમાં સબસ્ટ્રાંગની હાજરી તપાસવા માટેનું ફંક્શન"""
5     if sub_string in main_string:
6         return True
7     else:
8         return False
9
10 # વપરાશકર્તા પાસેથી ઇનપુટ લો
11 main_string = input("Enter main string: ")
12 sub_string = input("Enter substring: ")
13
14 # તપાસો અને પરિણામ દર્શાવો
15 if check_substring(main_string, sub_string):
16     print(f"{sub_string} is present in '{main_string}'")
17 else:
18     print(f"{sub_string} is not present in '{main_string}'")

```

મેમરી ટ્રીક

“FIND Method” - Find IN Directly with Methods

પ્રશ્ન 5(ક) [7 ગુણ]

ટપલ ઓપરેશન, ફુંક્શન અને મેથડ સમજાવો.

જવાબ

કોષ્ટક 11. ટપલ ઓપરેશન્સ

ઓપરેશન	વર્ણન	ઉદાહરણ	પરિણામ
બનાવટ	કૌંસ સાથે બનાવવું	t=(1,2)	(1, 2)
ઇન્ડેક્સિંગ	એલિમેન્ટ્સ એક્સેસ કરવા	t[1]	2
સ્લાઇસિંગ	સબસેટ મેળવવો	t[0:1]	(1,)
કેટનેશન	બે ટપલ જોડવા	(1)+(2)	(1, 2)
રિપિટેશન	રિપીટ કરવા	(1)*2	(1, 1)
મેન્બરશિપ	અસ્તિત્વ તપાસવું	1 in t	True
len()	આઇટમ્સની સંખ્યા	len(t)	2
count()	સંખ્યા ગણવી	t.count(1)	1
index()	પોઝિશન શોધવી	t.index(2)	1

કોડ ઉદાહરણ:

```

1 my_tuple = (3, 1, 4, 1, 5, 9)
2 print("પ્રથમ:", my_tuple[0])
3 print("સ્લાઇસ:", my_tuple[1:4])
4 print("1 ની સંખ્યા:", my_tuple.count(1))

```

```

5 print("4 નો ઇન્ડિક્સ:", my_tuple.index(4))
6 a, b, c, *rest = my_tuple # અનપેક્ચર

```

મેમરી ટ્રીક

“ICONS” - Immutable Collection Operations, Numbering, and Searching

OR

પ્રશ્ન 5(અ) [3 ગુણ]

લિસ્ટ મા આપેલ એલીમેન્ટ નો સરવાળો શોધવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

```

1 # લસ્ટના એલમીન્ટ્સનો સરવાળો શોધવાનો પ્રોગ્રામ
2
3 def sum_of_list(numbers):
4     """લસ્ટના બધા એલમીન્ટ્સનો સરવાળો શોધવા માટેનું ફંક્શન"""
5     total = 0
6     for num in numbers:
7         total += num
8     return total
9
10 # ઉદાહરણ
11 num_elements = int(input("Enter number of elements: "))
12 my_list = []
13
14 # એલમીન્ટ્સ મેળવો
15 for i in range(num_elements):
16     element = float(input(f"Enter element {i+1}: "))
17     my_list.append(element)
18
19 # સરવાળો ગણો
20 result1 = sum_of_list(my_list)
21 print("Sum using custom function:", result1)
22
23 # બલ્ટિટેઇન- sum()
24 result2 = sum(my_list)
25 print("Sum using built-in function:", result2)

```

મેમરી ટ્રીક

“SALT” - Sum All List Together

OR

પ્રશ્ન 5(બ) [4 ગુણ]

સેટ ફંક્શન અને ઓપરેશન દર્શાવવા માટે એક પ્રોગ્રામ લખો.

જવાબ

```

1 # સેટ ફંક્શન અને ઓપરેશન્સ દર્શાવતો પ્રોગ્રામ
2
3 set1 = {1, 2, 3, 4, 5}
4 set2 = {4, 5, 6, 7, 8}
5
6 # સેટ ઓપરેશન્સ
7 print("યુનિયન: ", set1 | set2)
8 print("ઇન્ટરસેક્શન: ", set1 & set2)
9 print("ડિફરન્સ: ", set1 - set2)
10 print("સમિટ્રાક ડફિરન્સ: ", set1 ^ set2)
11
12 # સેટ મેથ્ડ્સ
13 set3 = set1.copy()
14 set3.add(6)
15 set3.remove(1)
16 set3.discard(10) # ભૂલ નહીં
17 popped = set3.pop()
18 set3.clear()

```

મેમરી ટ્રીક

“COSI Methods” - Create, Operate, Search, Investigate with Set Methods

OR

પ્રશ્ન 5(ક) [૭ ગુણ]

ડિક્શનેરી ફંક્શન અને ઓપરેશન સમજાવવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

```

1 # ડિક્શનેરી ફંક્શન અને ઓપરેશન્સ દર્શાવતો પ્રોગ્રામ
2
3 # ડિક્શનેરી બનાવવી
4 student = {
5     'name': 'John',
6     'roll_no': 101,
7     'marks': 85
8 }
9
10 # એક્સેસ
11 print("Name:", student['name'])
12 print("Roll No:", student.get('roll_no'))
13
14 # સુધારવું અને ઉમેરવું
15 student['marks'] = 90
16 student['address'] = 'New York'
17
18 # દૂર કરવું
19 removed = student.pop('address')
20 last_item = student.popitem()
21
22 # મેથ્ડ્સ
23 print("Keys:", list(student.keys()))
24 print("Values:", list(student.values()))

```

```
25 print("Items:", list(student.items()))
26
27 # કલારિસ
28 student.clear()
```

મુખ્ય ઓપરેશન-સ્સ:

- એક્સેસ: કી અથવા get() મેથડનો ઉપયોગ કરીને
- મોડિફાય: અસ્થિત્વમાં રહેલી કીને નવું મૂલ્ય આપવું
- એડ: નવી કીને મૂલ્ય આપવું
- રિમૂવ: pop(), popitem(), અથવા del

મેમરી ટ્રીક

“ACME Dictionary” - Access, Create, Modify, Extract from Dictionary