

# પાયથોન પ્રોગ્રામિંગ (1323203) - સમર 2023 સોલ્યુશન

Milav Dabgar

૦૮ ઓગસ્ટ, ૨૦૨૩

## પ્રશ્ન 1(અ) [૩ ગુણ]

અલગોરિધમ વ્યાખ્યાપિત કરો. અલગોરિધમનાં ફાયદા શું છે?

### જવાબ

અલગોરિધમ એ ચોક્કસ સમસ્યાને ઉકેલવા માટે પગલાઓના ક્રમબદ્ધ સમૂહ અથવા નિયમોનો સેટ છે.

અલગોરિધમનાં ફાયદા:

- સ્પષ્ટતા (Clarity): સ્પષ્ટ, અસંદિગ્ધ સૂચનાઓ પ્રદાન કરે છે
- કાર્યક્ષમતા (Efficiency): સમય અને સંસાધનોને અનુકૂળ બનાવવામાં મદદ કરે છે
- પુનઃઉપયોગ (Reusability): સમાન સમસ્યાઓ માટે વારંવાર ઉપયોગ કરી શકાય છે
- ચકાસણી (Verification): અમલીકરણ પહેલાં પરીક્ષણ અને ડિબગ કરવું સરળ
- સંદેશાવ્યવહાર (Communication): ઉકેલને સંદેશાવ્યવહાર કરવા માટે બ્લૂપ્રિન્ટ તરીકે કામ કરે છે

### મેમરી ટ્રીક

"CERVC" (Clarity, Efficiency, Reusability, Verification, Communication)

## પ્રશ્ન 1(બ) [૪ ગુણ]

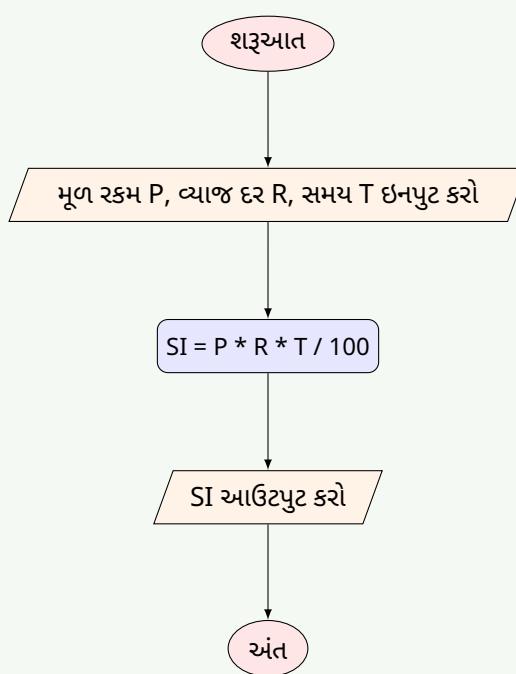
ફ્લોચાર્ટનો ઉપયોગ કરીને સમસ્યા ઉકેલવાના નિયમો શું છે? સાંદું વ્યાજ શોધવા માટેનો ફ્લોચાર્ટ ડિઝાઇન કરો.

### જવાબ

ફ્લોચાર્ટનો ઉપયોગ કરીને સમસ્યા ઉકેલવાના નિયમો:

- યોગ્ય સિમ્બોલ: વિવિધ ઓપરેશન માટે માનક સિમ્બોલનો ઉપયોગ કરવો
- દિશાનો પ્રવાહ: હમેશા ઉપરથી નીચે, ડાબેથી જમણે સ્પષ્ટ પ્રવાહ જાળવવો
- એક એન્ટ્રી/એન્ટ્રીન્ટ: સ્પષ્ટ શરૂઆત અને અંત બિંદુ હોવા જોઈએ
- સ્પષ્ટતા: પગલાં સ્પષ્ટ અને સંક્ષિપ્ત રાખવા
- સુસંગતતા: વિગતોનું સુસંગત સ્તર જાળવવું

આકૃતિ 1. સાંદું વ્યાજ ગણતરી માટેનો ફ્લોચાર્ટ



મેમરી ટ્રીક

"PDRSC" (Proper symbols, Direction flow, Required entry/exit, Simplicity, Consistency)

## પ્રશ્ન 1(ક) [૭ ગુણ]

પાયથોનનાં અસાઇમેન્ટ ઓપરેટરની ચાદી બનાવો અને કોઈપણ ત્રણ અસાઇમેન્ટ ઓપરેટરોની કામગીરી દર્શાવવા માટે પાયથોન કોડ બનાવો.

### જવાબ

પાયથોન અસાઇમેન્ટ ઓપરેટર્સ:

કોષ્ટક 1. અસાઇમેન્ટ ઓપરેટર્સ

| ઓપરેટર | ઉદાહરણ  | સમકક્ષ     |
|--------|---------|------------|
| =      | x = 5   | x = 5      |
| +=     | x += 5  | x = x + 5  |
| -=     | x -= 5  | x = x - 5  |
| *=     | x *= 5  | x = x * 5  |
| /=     | x /= 5  | x = x / 5  |
| %=     | x %= 5  | x = x % 5  |
| //=    | x //= 5 | x = x // 5 |
| **=    | x **= 5 | x = x ** 5 |
| &=     | x &= 5  | x = x & 5  |
| =      | x  = 5  | x = x   5  |
| ^=     | x ^= 5  | x = x ^ 5  |
| >>=    | x >>= 5 | x = x >> 5 |
| <<=    | x <<= 5 | x = x << 5 |

અસાઇમેન્ટ ઓપરેટર્સ દર્શાવતો કોડ:

```

1 # અસાઇન્મેટ ઓપરેટર્સનું પ્રદર્શન
2 num = 10
3 print("પ્રારંભકી મૂલ્યઃ", num)
4
5 # += ઓપરેટરનો ઉપયોગ
6 num += 5
7 print("+= 5 પછીઃ", num) # આઉટપુટ: 15
8
9 # -= ઓપરેટરનો ઉપયોગ
10 num -= 3
11 print("-= 3 પછીઃ", num) # આઉટપુટ: 12
12
13 # *= ઓપરેટરનો ઉપયોગ
14 num *= 2
15 print("*= 2 પછીઃ", num) # આઉટપુટ: 24

```

### મેમરી ટ્રીક

"VALUE" (Variable Assignment is Like Updating Existing values)

OR

## પ્રક્રિયા 1(ક) [૭ ગુણ]

પાયથોનનાં ડેટા ટાઇપ્સની યાદી બનાવો અને કોઈપણ ત્રણ ડેટા ટાઇપ્સને ઓળખવા માટેનો પાયથોન કોડ બનાવો.

### જવાબ

પાયથોન ડેટા ટાઇપ્સ:

કોષ્ટક 2. પાયથોન ડેટા ટાઇપ્સ

| ડેટા ટાઇપ | વર્ણન                      | ઉદાહરણ           |
|-----------|----------------------------|------------------|
| int       | ઇન્ટીજર (પૂર્ણક સંખ્યાઓ)   | 42               |
| float     | ફ્લોટિંગ પોઇન્ટ (દશાંશ)    | 3.14             |
| str       | સ્ટ્રિંગ (ટેક્સ્ટ)         | "Hello"          |
| bool      | બૂલિયન (True/False)        | True             |
| list      | ક્રમિક, પરિવર્તનશીલ સંગ્રહ | [1, 2, 3]        |
| tuple     | ક્રમિક, અપરિવર્તનીય સંગ્રહ | (1, 2, 3)        |
| set       | અક્રમિક સંગ્રહ             | {1, 2, 3}        |
| dict      | કી-વેલ્ચુ જોડી             | {"name": "John"} |
| complex   | કોમ્પ્લેક્સ નંબર           | 2+3j             |
| NoneType  | નોન દર્શાવે છે             | None             |

ત્રણ ડેટા ટાઇપ્સ ઓળખવા માટેનો કોડ:

```

1 # ડેટા ટાઇપ્સ ઓળખવાનો પ્રોગ્રામ
2 def identify_data_type(value):
3     data_type = type(value).__name__
4     print("મૂલ્ય: {value}")
5     print("ડેટા ટાઇપ: {data_type}")
6     print("-" * 20)
7
8 # 3 અલગઅલગ - ડેટા ટાઇપ્સ સાથે ટેસ્ટિંગ

```

```

9 identify_data_type(42)      # Integer
10 identify_data_type(3.14)    # Float
11 identify_data_type("Hello World") # String
12
13 # આઉટપુટ:
14 # મૂલ્ય: 42
15 # ડાટા ટાઇપ: int
16 #
17 # -----
18 # મૂલ્ય: 3.14
19 # ડાટા ટાઇપ: float
20 #
21 # -----
22 # મૂલ્ય: Hello World
23 # ડાટા ટાઇપ: str
24 #

```

**મેમરી ટ્રીક****"TYPE-ID"** (Tell Your Python Elements - Identify Data)**પ્રશ્ન 2(અ) [3 ગુણ]**

સ્યુડોકોડ વ્યાખ્યાપિત કરો. કોઈપણ બે સંખ્યા માંથી સૌથી નાની સંખ્યા શોધવા માટે સ્યુડોકોડ લખો.

**જવાબ**

સ્યુડોકોડ એ એલગોરિધમનું ઉચ્ચ-સ્તરીય વર્ણન છે જે પ્રોગ્રામ્બિંગ ભાષાના માળખાકીય સંકેતોનો ઉપયોગ કરે છે પરંતુ મશીન વાંચન કરતાં માનવ વાંચન માટે ડિજાઇન કરેલ છે.

બે સંખ્યાઓમાંથી સૌથી નાની શોધવા માટે સ્યુડોકોડ:

```

1 BEGIN
2   INPUT first_number, second_number
3   IF first_number < second_number THEN
4     smallest = first_number
5   ELSE
6     smallest = second_number
7   END IF
8   OUTPUT smallest
9 END

```

**મેમરી ટ્રીક****"RISE"** (Read Input, Select smallest, Echo result)**પ્રશ્ન 2(બ) [4 ગુણ]**

યુઝર્સ પાસેથી ત્રણ ઇનપુટ વાંચો અને સંખ્યાઓની સરેરાશ શોધવા માટેનો પાયથોન કોડ વિકસાવો.

**જવાબ**

```

1 # ત્રણ સંખ્યાઓની સરેરાશ ગણવા માટેનો પ્રોગ્રામ
2 num1 = float(input("પ્રથમ સંખ્યા દાખલ કરો: "))
3 num2 = float(input("બોજી સંખ્યા દાખલ કરો: "))

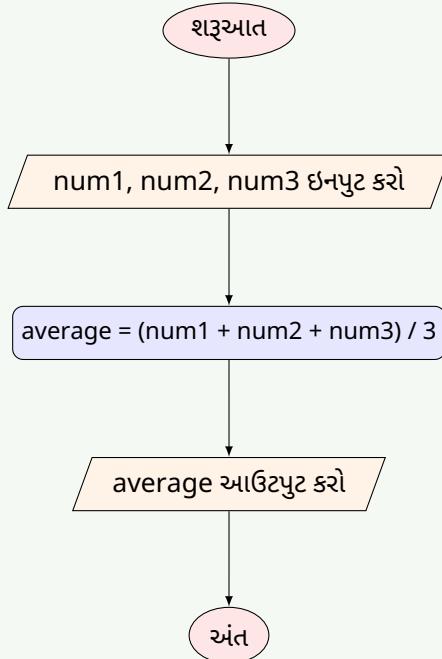
```

```

4 num3 = float(input("તૃજી સંખ્યા દાખલ કરો: "))
5
6 # સરેરાશની ગણતરી
7 average = (num1 + num2 + num3) / 3
8
9 # પરિણામ દર્શાવો
10 print(f"{num1}, {num2}, અને {num3}ની સરેરાશ: {average}")

```

આકૃતિ 2. સરેરાશ ગણતરી માટેનો ફ્લોચાર્ટ



## મેમરી ટ્રીક

"I-ADD-D" (Input three, ADD them up, Divide by 3)

## પ્રશ્ન 2(ક) [7 ગુણ]

દાખલ કરેલ સંખ્યા prime છે કે નહીં તે બતાવવા પાયથોન કોડ લખો.

## જવાબ

```

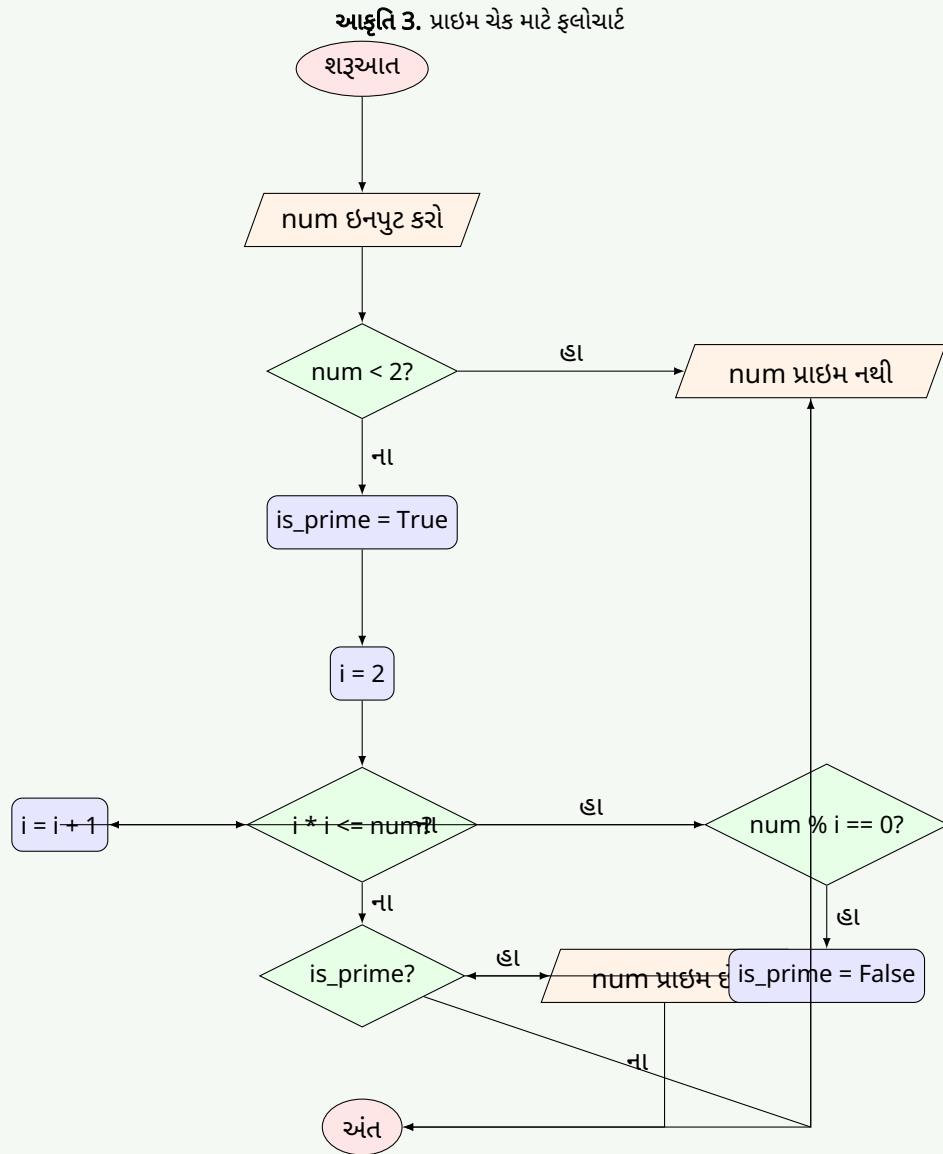
1 # સંખ્યા પ્રાઇમ છે કે નહીં તે તપાસવાનો પ્રોગ્રામ
2 num = int(input("એક સંખ્યા દાખલ કરો: "))
3
4 # થી2 ઓછી સંખ્યા છે કે નહીં તપાસો
5 if num < 2:
6     print(f"{num} એક પ્રાઇમ સંખ્યા નથી")
7 else:
8     # is_prime ને True તરીકે આરંભો
9     is_prime = True
10
11 # 2 થી sqrt(num) સુધી તપાસો
12 for i in range(2, int(num**0.5) + 1):
13     if num % i == 0:

```

```

14     is_prime = False
15     break
16
17     # પરિણામ દર્શાવો
18     if is_prime:
19         print(f"{num} એક પ્રાઇમ સંખ્યા છે")
20     else:
21         print(f"{num} એક પ્રાઇમ સંખ્યા નથી")

```



### મેમરી ટ્રીક

"PRIME" (Positive number, Range check from 2 to  $\sqrt{n}$ , If divisible it's Multiple, Else it's prime)

OR

## પ્રશ્ન 2(અ) [3 ગુણ]

ફ્લોચાર્ટ અને એલોરિધમ વર્ચેનો તફાવત લખો.

### જવાબ

કોષ્ટક 3. ફ્લોચાર્ટ અને એલોરિધમ વર્ચેનો તફાવત

| ફ્લોચાર્ટ   | એલોરિધમ                                      |
|---|--|
| માનક સિમ્બોલ અને આકારોનો ઉપયોગ કરીને દૃશ્ય પ્રતિનિધિત્વ | લેખિત વર્ણન માળખાડીય ભાષાનો ઉપયોગ કરીને      |
| ગ્રાફિકલ પ્રક્રિયાને કારણે સમજૃતું સરળ                  | સિટેક્સ અને શબ્દાવલીનું જ્ઞાન જરૂરી          |
| તાર્કિક પ્રવાહ અને સંબંધોને રૂપાંતર રીતે દર્શાવે        | કમિક કમમાં વિગતવાર પગલાં પ્રદાન કરે          |
| બનાવવા માટે સમય-લેતી પરંતુ સમજવા માટે સરળ               | જડપથી ડ્રાફ્ટ પરંતુ સમજવામાં મુશ્કેલ હોઈ શકે |
| ફેરફાર કરવા કે અપડેટ કરવા વધુ મુશ્કેલ                   | ફેરફાર કરવા કે અપડેટ કરવા વધુ સરળ            |

### મેમરી ટ્રીક

"VITAL" (Visual vs Textual, Interpretation ease, Time to create, Alteration flexibility, Logical representation)

OR

## પ્રશ્ન 2(બ) [4 ગુણ]

નીચેનાં કોડનું આઉટપુટ શું છે?

### જવાબ

```

1 x=10
2 y=2
3 print(x*y)
4 print(x ** y)
5 print(x//y)
6 print(x % y)

```

### જવાબ:

કોષ્ટક 4. આઉટપુટ સમજૂતી

| ઓપરેશન | સમજૂતી                        | આઉટપુટ |
|--------|-------------------------------|--------|
| x*y    | ગુણાકાર: $10 \times 2$        | 20     |
| x**y   | ઘાતાંક: $10^2$                | 100    |
| x//y   | પૂર્ણાંક ભાગાકાર: $10 \div 2$ | 5      |
| x%y    | મોડ્યુલસ (શેષ): $10 \div 2$   | 0      |

### મેમરી ટ્રીક

"MEMO" (Multiply, Exponent, Modulo, Operations)

OR

## પ્રશ્ન 2(ક) [૭ ગુણ]

નીચેની પેટર્ન દર્શાવવા પાયથોન કોડ લખો:

### જવાબ

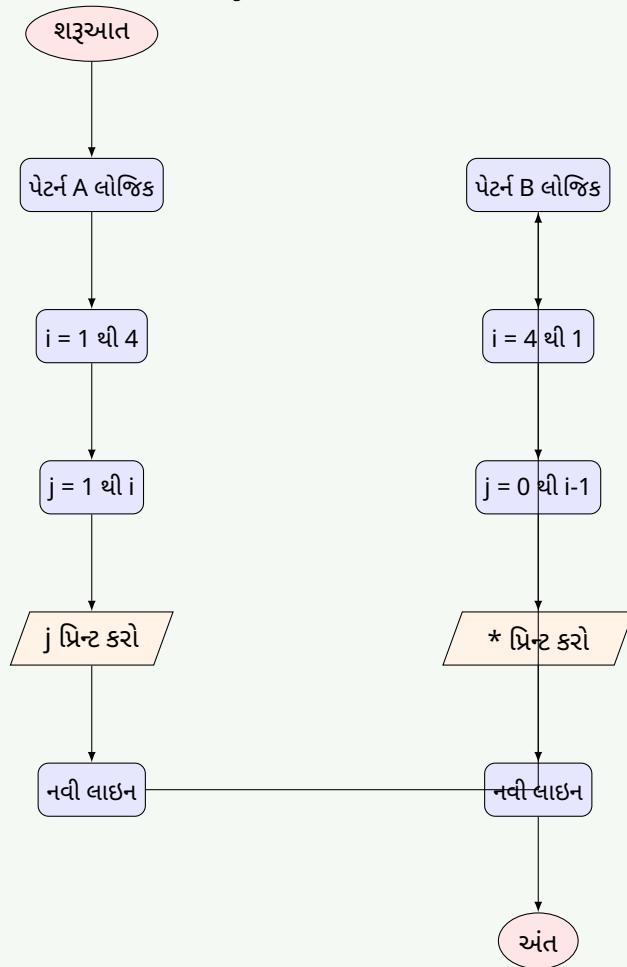
|         |         |
|---------|---------|
| A)      | B)      |
| 1       | * * * * |
| 1 2     | * * *   |
| 1 2 3   | * *     |
| 1 2 3 4 | *       |

```

1 # પેટર્ન A: સંખ્યા પેટર્ન
2 print("પેટર્ન A:")
3 for i in range(1, 5):
4     for j in range(1, i + 1):
5         print(j, end=" ")
6         print()
7
8 # પેટર્ન B: તારા પેટર્ન
9 print("પેટર્ન B:")
10 for i in range(4, 0, -1):
11     for j in range(i):
12         print("*", end=" ")
13         print()

```

આકૃતિ 4. પેટર્ન માટે ફ્લોચાર્ટ



## મેમરી ટ્રીક

"LOOP-NED" (Loop Outer, Order Pattern, Nested loops, End with newline, Display)

## પ્રક્રિયા 3(અ) [3 ગુણ]

જૂરી ઉદાહરણો સાથે break statementનાં ઉપયોગનું વર્ણન કરો.

## જવાબ

break સ્ટેટમેન્ટનો ઉપયોગ લૂપને વર્ચ્યેથી સમાપ્ત કરવા માટે થાય છે, જ્યારે કોઈ ચોક્કસ શરત પૂરી થાય.

## ઉદાહરણ:

```

1 # લખિએટમાં પ્રથમ વખિમ સંખ્યા શોધવી
2 numbers = [2, 4, 6, 7, 8, 10]
3 for num in numbers:
4     if num % 2 != 0:
5         print(f"વખિમ સંખ્યા મળી: {num}")
6         break
7     print(f"{num} તપાસી રહ્યા છીએ")

```

## આઉટપુટ:

```

1 2 તપાસી રહ્યા છીએ
2 4 તપાસી રહ્યા છીએ
3 6 તપાસી રહ્યા છીએવખિમ
4 સંખ્યા મળી: 7

```

## મેમરી ટ્રીક

"EXIT" (EXecute until condition, Immediately Terminate)

## પ્રક્રિયા 3(બ) [4 ગુણ]

યોગ્ય ઉદાહરણ સાથે if...else statement સમજાવો.

## જવાબ

if...else સ્ટેટમેન્ટ એ એક કન્ડિશનલ સ્ટેટમેન્ટ છે જે નિર્દિષ્ટ શરત True કે False હોવાના આધારે અલગ-અલગ કોડ બ્લોક્સ એક્ઝિક્યુટ કરે છે.

## સિન્ફોન્સ:

```

1 if શરત:
2     # જો શરત True હોય તો આ કોડ એક્ઝિક્યુટ થશે
3 else:
4     # જો શરત False હોય તો આ કોડ એક્ઝિક્યુટ થશે

```

## ઉદાહરણ:

```

1 # સંખ્યા સમ છે કે વખિમ તે તપાસવું
2 number = int(input("એક સંખ્યા દાખલ કરો: "))
3
4 if number % 2 == 0:
5     print(f"{number} એક સમ સંખ્યા છે")

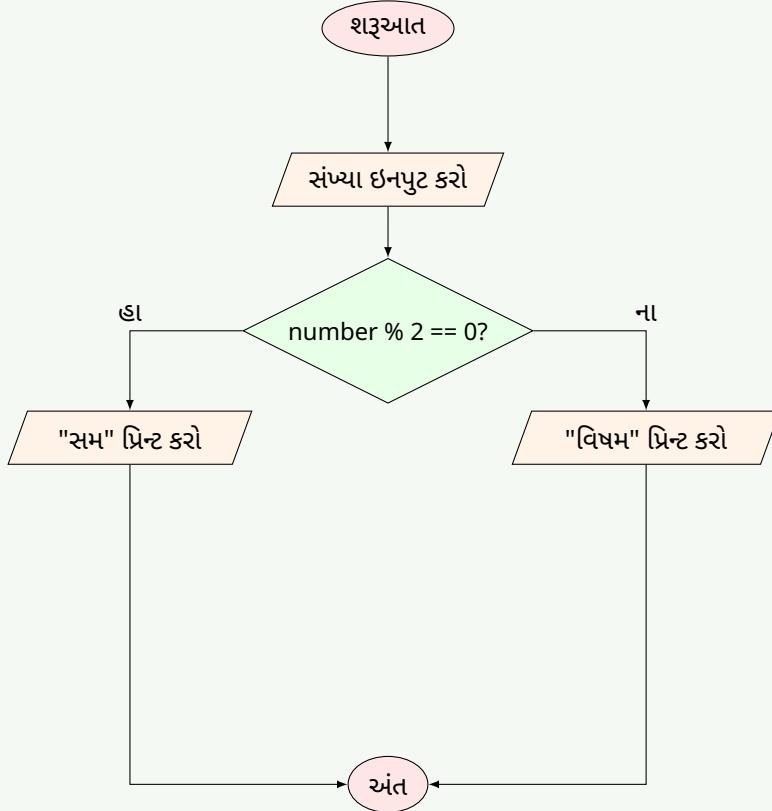
```

```

6 else:
7     print(f"{number} એક વિષમ સંખ્યા છે")

```

## આકૃતિ 5. If-Else માટે ફ્લોચાર્ટ



## મેમરી ટ્રીક

"CITE" (Check condition, If True Execute this, Else execute that)

## પ્રશ્ન 3(ક) [7 ગુણા]

0 થી N સંખ્યા સુધીની ફીબોનાકી શ્રેણી પ્રિન્ટ કરવા માટે યુઝર ડેફાઇન ફંક્શન બનાવો.

## જવાબ

```

1 # ફીબોનાકી શ્રેણી પ્રનિટ કરવા માટેનું ફંક્શન
2 def print_fibonacci(n):
3     # પરથમ બે પદો ઇનશિયલાઇઝ કરો
4     a, b = 0, 1
5
6     # n માન્ય છે કે નહીં તે તપાસો
7     if n < 0:
8         print("કૃપા કરીને એક હકારાત્મક સંખ્યા દાખલ કરો!")
9         return
10
11    # ફીબોનાકી શ્રેણી પ્રનિટ કરો
12    print(n, "સુધીની ફીબોનાકી શ્રેણી:")
13

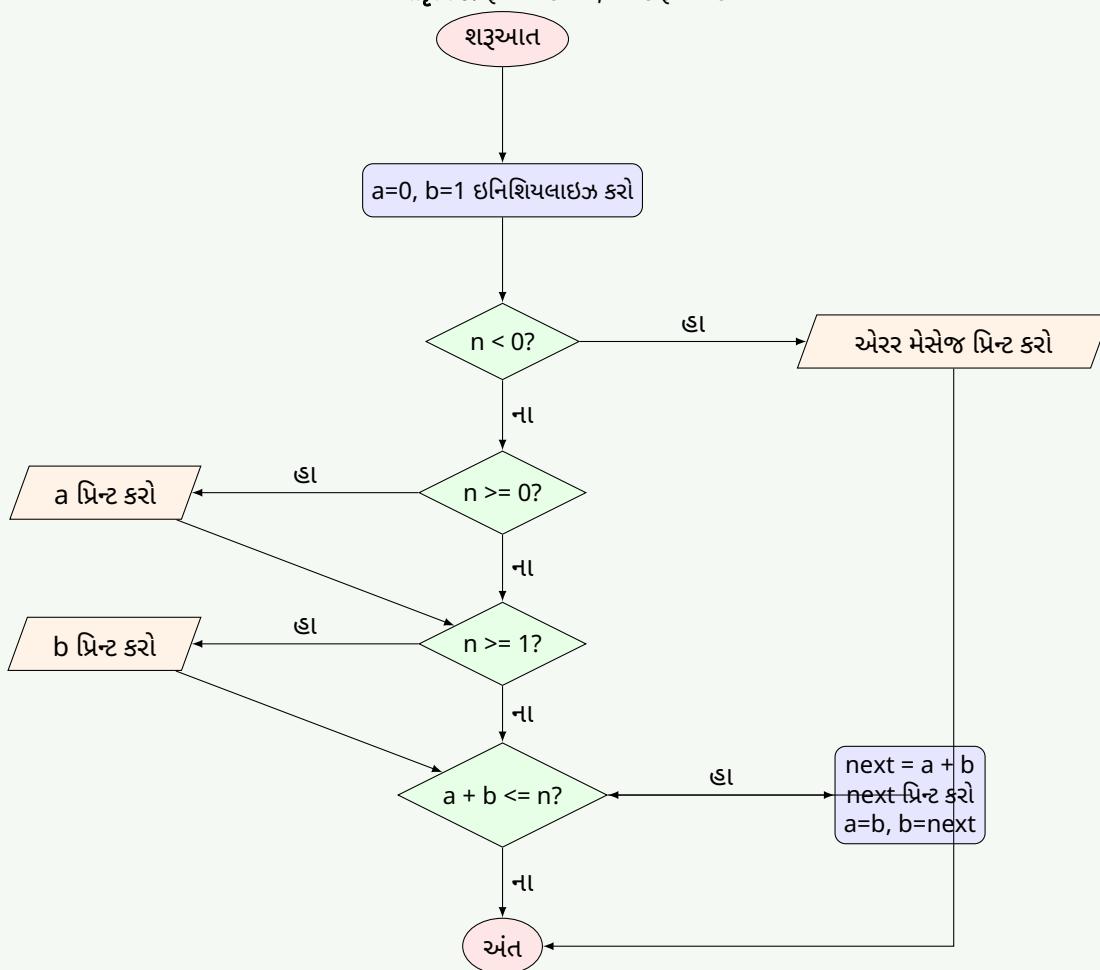
```

```

14 if n >= 0:
15     print(a, end=" ") # પ્રથમ પદ પ્રનિટ કરો
16
17 if n >= 1:
18     print(b, end=" ") # બીજો પદ પ્રનિટ કરો
19
20 # બાકીની શરેખી બનાવો અને પ્રનિટ કરો
21 while a + b <= n:
22     next_term = a + b
23     print(next_term, end=" ")
24     a, b = b, next_term
25
26 # ફંક્શનનું ટેસ્ટિંગ
27 print_fibonacci(55)

```

### આકૃતિ 6. ફીબોનાકી શ્રેણી માટે ફ્લોચાર્ટ



### મેમરી ટ્રીક

"FIBER" (First terms set, Initialize variables, Build next term, Echo results, Repeat until limit)

OR

## પ્રશ્ન 3(અ) [3 ગુણ]

જરૂરી ઉદાહરણો સાથે continue statementના ઉપયોગનું વર્ણન કરો.

### જવાબ

continue સ્ટેમ્પેન્ટનો ઉપયોગ લૂપની વર્તમાન ઇટરેશન છોડીને આગળની ઇટરેશન પર જવા માટે થાય છે.  
ઉદાહરણ:

```
1 # 1 થી 10 સુધીની માત્ર વખિમ સંખ્યાઓ પ્રનિટ કરવી
2 for i in range(1, 11):
3     if i % 2 == 0:
4         continue # સમ સંખ્યાઓ છોડી દો
5         print(i)
```

### આઉટપુટ:

```
1
2
3
4
5
6
7
8
9
```

### મેમરી ટ્રીક

"SKIP" (Skip current iteration, Keep looping, Ignore remaining statements, Proceed to next iteration)

OR

## પ્રશ્ન 3(બ) [4 ગુણ]

ઉદાહરણ સાથે For loop statement સમજાવો.

### જવાબ

For લૂપનો ઉપયોગ કોઈ સિક્વન્સ (જેમ કે લિસ્ટ, ટ્પલ, સ્ટ્રિંગ) પર ઇટરેશન કરવા માટે થાય છે.  
સિન્ક્રિપ્શન:

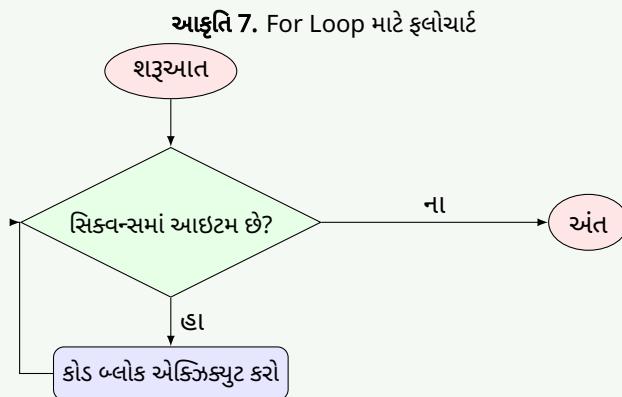
```
1 for વેરાયિબલ in સિક્વન્સ:
2     # દરેક આઇટમ માટે એક્ઝિયુટ થનાર કોડ
```

### ઉદાહરણ:

```
1 # 1 થી 5 સુધીની સંખ્યાઓના વર્ગ પ્રનિટ કરવા
2 for num in range(1, 6):
3     square = num ** 2
4     print(f"{num}નો વર્ગ {square} છે")
```

### આઉટપુટ:

```
1 નો
2 1 વર્ગ 1 છેનો
3 2 વર્ગ 4 છેનો
4 3 વર્ગ 9 છેનો
5 4 વર્ગ 16 છેનો
6 5 વર્ગ 25 છે
```

**મેમરી ટ્રીક**

"FIRE" (For each Item, Run commands, Execute until end)

OR

**પ્રશ્ન 3(ક) [૭ ગુણા]**

આપેલ નંબર આર્મ્સ્ટ્રોગ નંબર છે કે પેલિન્ડ્રોમ તે નિર્ધારિત કરવા પાયથોન કોડ લખો.

**જવાબ**

```

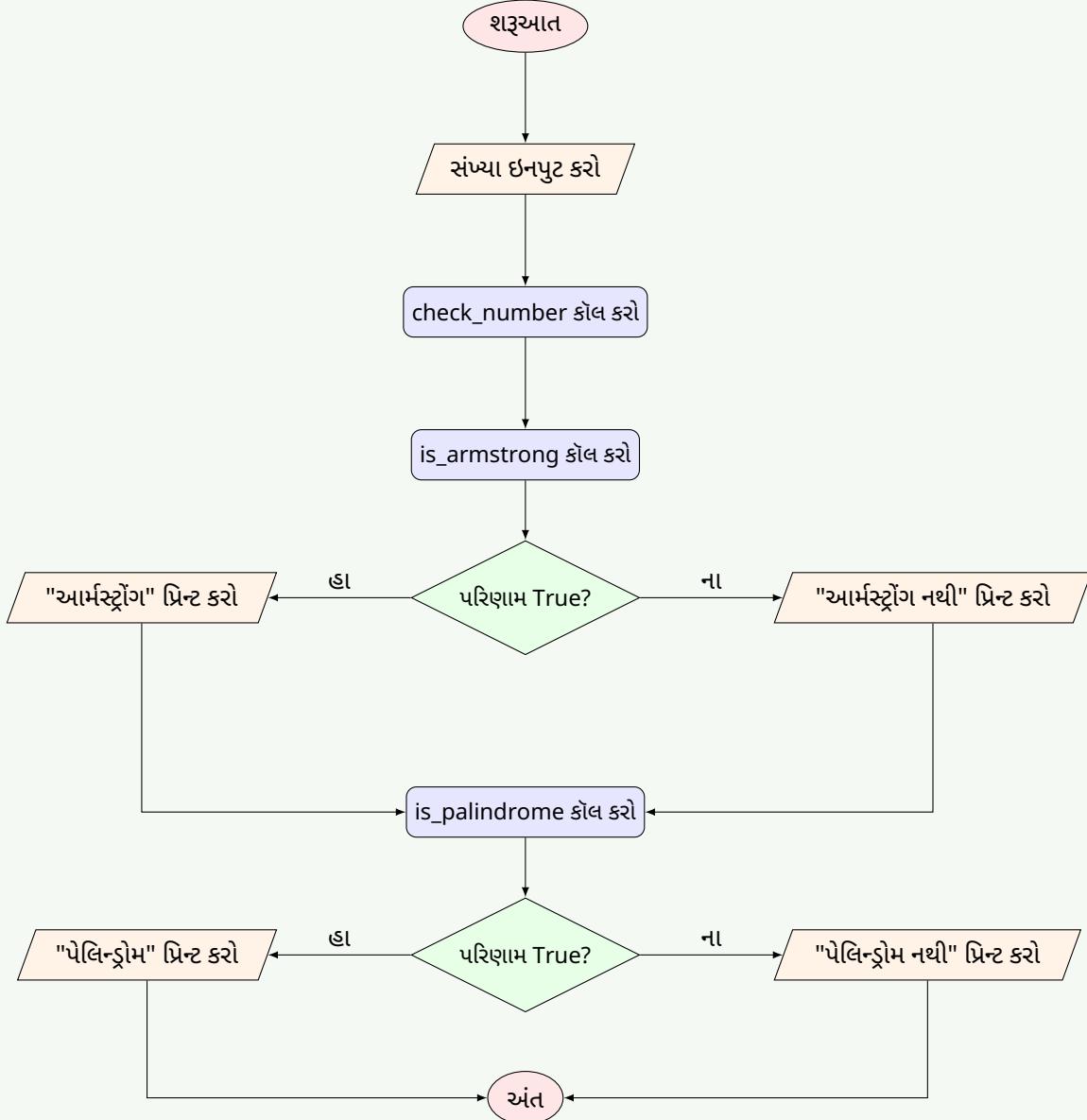
1 # સંખ્યા આર્મ્સ્ટ્રોગ નંબર છે કે નહીં તે તપાસવાનું ફ્ક્ષન
2 def is_armstrong(num):
3     # ડાયલિસ્ટની સંખ્યા ગણવા
4     num_str = str(num)
5     n = len(num_str)
6
7     # દરેક ડાયલિને n ઘાત પર ઊંચકી તેનો સરવાળો કરો
8     sum_of_powers = sum(int(digit)**n for digit in num_str)
9
10    # સરવાળો મૂળ સંખ્યા સાથે સરખાવો
11    return sum_of_powers == num
12
13 # સંખ્યા પેલિન્ડ્રોમ છે કે નહીં તે તપાસવાનું ફ્ક્ષન
14 def is_palindrome(num):
15     num_str = str(num)
16     # સ્ટ્રેચ તેના રવિરક્ષ સાથે સરખાવો
17     return num_str == num_str[::-1]
18
19 # મુખ્ય ફ્ક્ષન
20 def check_number(num):
21     if is_armstrong(num):
22         print(f"{num} એક આર્મ્સ્ટ્રોગ નંબર છે")
23     else:
24         print(f"{num} એક આર્મ્સ્ટ્રોગ નંબર નથી")
25
26     if is_palindrome(num):
27         print(f"{num} એક પેલિન્ડ્રોમ છે")
28     else:
29         print(f"{num} એક પેલિન્ડ્રોમ નથી")
30
  
```

```

31 # ફંક્શનનું ટેસ્ટિંગ
32 number = int(input("એક સંખ્યા દાખલ કરો: "))
33 check_number(number)

```

આફ્ટિ 8. આર્મ્સ્ટ્રોંગ અને પેલિન્ડ્રોમ ચેક માટે ફ્લોચાર્ટ



### મેમરી ટ્રીક

"APC" (Armstrong check: Power sum of digits, Palindrome check: Compare with reverse)

## પ્રશ્ન 4(અ) [3 ગુણ]

સ્કેન કરેલ નંબર even છે કે odd તે શોધવા પાયથોન કોડ વિકસાવો અને ચોગ્ય મેસેજ પ્રિન્ટ કરો.

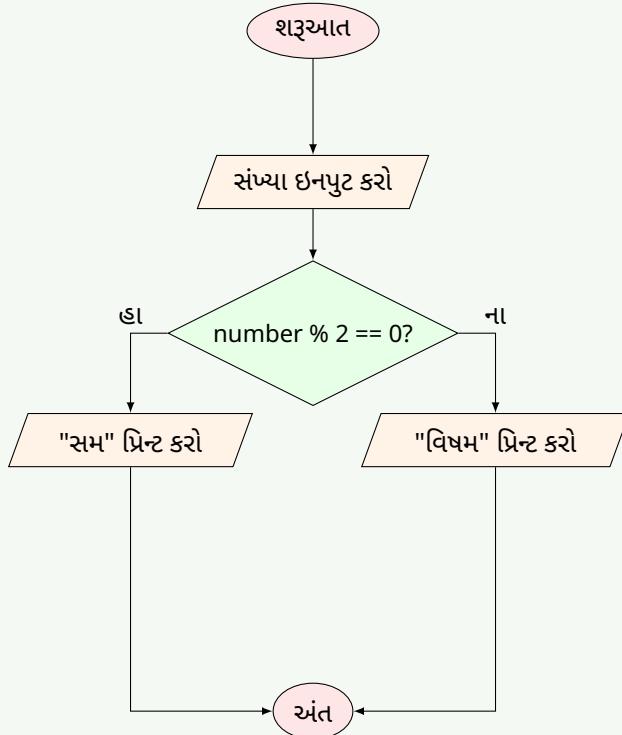
**જવાબ**

```

1 # સંખ્યા સમ છે કે વિષમ તે તપાસવાનો પ્રોગ્રામ
2 number = int(input("એક સંખ્યા દાખલ કરો: "))
3
4 if number % 2 == 0:
5     print(f"{number} એક સમ સંખ્યા છે")
6 else:
7     print(f"{number} એક વિષમ સંખ્યા છે")

```

આકૃતિ 9. સમ/વિષમ માટે ફ્લોચાર્ટ

**મેમરી ટ્રીક**

"MODE" (Modulo Operation Determines Even-odd)

**પ્રશ્ન 4(બ) [4 ગુણ]**

ફુંક્શનની વ્યાખ્યા આપો. યુઝર ડિફાઇન ફુંક્શન યોગ્ય ઉદાહરણ આપી સમજાવો.

**જવાબ**

ફુંક્શન એ કોડનો એવો બ્લોક છે જે ચોક્કસ કાર્ય કરવા માટે વ્યવસ્થિત અને ફરીથી ઉપયોગ કરી શકાય છે.

**યુઝર-ડિફાઇન ફુંક્શનના ઘટકો:**

- **def કીવર્ક:** ફુંક્શન વ્યાખ્યાની શરૂઆત દર્શાવે છે
- **ફુંક્શન નામ:** ફુંક્શન માટે ઓળખતરી
- **પેરામીટર્સ:** ઇનપુટ વેલ્યુઝ (વૈકલ્પિક)
- **ડોક્સિન્ગ:** ફુંક્શનનું વાર્ણન (વૈકલ્પિક)
- **ફુંક્શન બોડી:** એક્ઝિક્યુટ થનાર કોડ
- **રિટર્ન સ્ટેમ્પેન્ટ:** આઉટપુટ વેલ્યુ (વૈકલ્પિક)

**ઉદાહરણ:**

```

1 # લંબચોરસનું ફ્લેટ્રફણ ગણવા માટેનું યુઝરડિઝિએન- ફક્શન
2 def calculate_area(length, width):
3     """
4     લંબચોરસનું ફ્લેટ્રફણ ગણે છે
5     """
6     area = length * width
7     return area
8
9 # ફક્શન કોલ કરો
10 result = calculate_area(5, 3)
11 print("લંબચોરસનું ફ્લેટ્રફણ: " + str(result))

```

**મેમરી ટ્રીક**

"DRAPE" (Define function, Receive parameters, Acquire result, Process data, End with return)

**પ્રશ્ન 4(ક) [7 ગુણ]**

વિવિધ સ્ટ્રિંગ ઓપરેશનની યાદી બનાવો અને કોઈપણ ત્રણ ઉદાહરણનો ઉપયોગ કરીને સમજાવો.

**જવાબ**

પાયથોનમાં સ્ટ્રિંગ ઓપરેશન્સ:

**કોષ્ટક 5. સ્ટ્રિંગ ઓપરેશન્સ**

| ઓપરેશન             | વર્ણન   |
|--------------------|---|
| Concatenation      | + નો ઉપયોગ કરીને સ્ટ્રિંગ્સ જોડવી             |
| Repetition         | * નો ઉપયોગ કરીને સ્ટ્રિંગ રિપીટ કરવી          |
| Indexing           | પોઝિશન દ્વારા કેરેક્ટર એક્સેસ કરવા            |
| Slicing            | સ્ટ્રિંગનો ભાગ એક્સ્ટ્રેક્ટ કરવો              |
| Methods            | બિલ્ટ-ઇન ફંક્શન્સ (len, upper, lower, વગેરે)  |
| Membership Testing | સ્ટ્રિંગમાં સબસ્ટ્રિંગ છે કે નાહીં તે તપાસવું |
| Formatting         | ફોર્મેટ સ્ટ્રિંગ્સ બનાવવી                     |
| Escape Sequences   | \થી શરૂ થતા સ્પેશિયલ કેરેક્ટર્સ               |

**ઉદાહરણ:****1. સ્ટ્રિંગ Concatenation:**

```

1 first_name = "John"
2 last_name = "Doe"
3 full_name = first_name + " " + last_name
4 print(full_name) # આઉટપુટ: John Doe

```

**2. સ્ટ્રિંગ Slicing:**

```

1 message = "Python Programming"
2 print(message[0:6]) # આઉટપુટ: Python
3 print(message[7:]) # આઉટપુટ: Programming
4 print(message[-11:]) # આઉટપુટ: Programming

```

**3. સ્ટ્રિંગ Methods:**

```

1 text = "python programming"
2 print(text.upper()) # આઉટપુટ: PYTHON PROGRAMMING
3 print(text.capitalize()) # આઉટપુટ: Python programming
4 print(text.replace("python", "Java")) # આઉટપુટ: Java programming

```

**મેમરી ટ્રીક****"CSM"** (Concatenate strings, Slice portions, Manipulate with methods)**OR****પ્રશ્ન 4(અ) [3 ગુણ]**

પોઝિટિવ અને નેગેટિવ નંબર તપાસવા પાયથોન કોડ બનાવો.

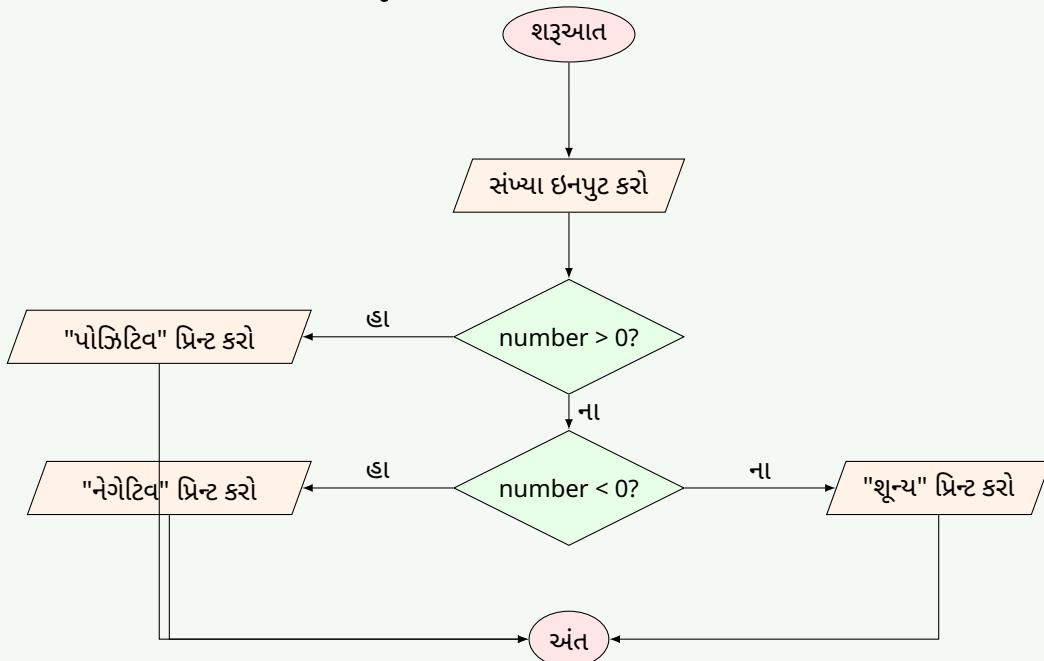
**જવાબ**

```

1 # સંખ્યા પોઝિટિવ છે કે નેગેટિવ તે તપાસવાનો પ્રોગ્રામ
2 number = float(input("એક સંખ્યા દાખલ કરો: "))
3
4 if number > 0:
5     print(f"{number} એક પોઝિટિવ સંખ્યા છે")
6 elif number < 0:
7     print(f"{number} એક નેગેટિવ સંખ્યા છે")
8 else:
9     print("સંખ્યા શૂન્ય છે")

```

આકૃતિ 10. પોઝિટિવ/નેગેટિવ માટે ફ્લોચાર્ટ

**મેમરી ટ્રીક****"SIGN"** (See If Greater than 0, Negative otherwise)

OR

## પ્રશ્ન 4(બ) [4 ગુણ]

યોગ્ય ઉદાહરણો સાથે local અને global વેરિએબલ સમજાવો.

### જવાબ

પાયથોનમાં વેરિએબલ્સના અલગ-અલગ સ્કોપ્સ હોઈ શકે છે:

કોષ્ટક 6. વેરિએબલ સ્કોપ્સ

| વેરિએબલ પ્રકાર  | વર્ણન  |
|-----------------|--|
| Local Variable  | ફુકશનની અંદર વ્યાપ્તાયિત અને માત્ર તે ફુકશનની અંદર જ એક્સેસિબલ |
| Global Variable | ફુકશનની બહાર વ્યાપ્તાયિત અને પ્રોગ્રામના તમામ ભાગમાં એક્સેસિબલ |

### ઉદાહરણ:

```

1 # Global વેરિએબલ
2 count = 0
3
4 def update_count():
5     # Local વેરિએબલ
6     local_var = 5
7
8     # ફુકશનની અંદર Global વેરિએબલ એક્સેસ કરવો
9     global count
10    count += 1
11
12    print(f"Local: {local_var}")
13    print(f"Global (inside): {count}")
14
15    # ફુકશન કોલ કરો
16    update_count()
17
18    # ફુકશનની બહાર વેરિએબલ એક્સેસ કરવા
19    print(f"Global (outside): {count}")

```

### મેમરી ટ્રીક

"SCOPE" (Some variables Confined to function Only, Program-wide Exposure for others)

OR

## પ્રશ્ન 4(ક) [7 ગુણ]

વિવિધ લિસ્ટ ઓપરેશનની યાદી બનાવો અને કોઈપણ ત્રાણ ઉદાહરણનો ઉપયોગ કરીને સમજાવો.

### જવાબ

પાયથોનમાં લિસ્ટ ઓપરેશન્સ:

કોષ્ટક 7. લિસ્ટ ઓપરેશન્સ

| ઓપરેશન       | વર્ણન                                |
|--------------|--------------------------------------|
| લિસ્ટ બનાવવી | સ્ક્રેચ બ્રેક્ટ્સ [] નો ઉપયોગ        |
| ઇન્ડેક્સિંગ  | પોઝિશન દ્વારા એલિમેન્ટ એક્સેસ કરવા   |
| સ્લાઇસિંગ    | લિસ્ટના ભાગો એક્સ્ટ્રેક્ટ કરવા       |
| અપેન્ડ       | છેલ્લે એલિમેન્ટ ઉમેરવા               |
| ઇન્સ્ટર્ટ    | ચોક્કસ પોઝિશન પર એલિમેન્ટ ઉમેરવા     |
| રિમૂવ        | ચોક્કસ એલિમેન્ટ દૂર કરવા             |
| પોપ          | એલિમેન્ટ દૂર કરવું અને પાછું મેળવવું |
| સોટ્         | લિસ્ટ એલિમેન્ટ્સ ઓર્ડર કરવા          |
| રિવર્સ       | લિસ્ટનો કમ ઊલટાવવો                   |
| એક્સ્ટેન્ડ   | લિસ્ટ્સ જોડવી                        |

ઉદાહરણ:

## 1. લિસ્ટ ઇન્ડેક્સિંગ અને સ્લાઇસિંગ:

```

1 fruits = ["apple", "banana", "cherry", "orange", "kiwi"]
2 print(fruits[1])      # આઉટપુટ: banana
3 print(fruits[-1])     # આઉટપુટ: kiwi
4 print(fruits[1:4])    # આઉટપુટ: ['banana', 'cherry', 'orange']

```

## 2. લિસ્ટ મેથ્ડ્સ (append, insert, remove):

```

1 numbers = [1, 2, 3]
2 numbers.append(4)    # છેલ્લે 4 ઉમેરો
3 numbers.insert(0, 0)  # પોઝિશન 0 પર 0 ઇન્સર્ટ કરો
4 numbers.remove(2)    # 2 વેલ્યુ ધરાવતો એલમિન્ટ દૂર કરો
5 print(numbers)       # આઉટપુટ: [0, 1, 3, 4]

```

## 3. લિસ્ટ કોમ્પ્રેહેન્શન્સ:

```

# સ્ક્રેચસની લિસ્ટ બનાવવી
squares = [x**2 for x in range(1, 6)]
print(squares) # આઉટપુટ: [1, 4, 9, 16, 25]

```

## મેમરી ટ્રીક

"AIM" (Access with index, Insert/modify elements, Make using comprehensions)

## પ્રક્રિયા [3 ગુણ]

લિસ્ટમાં આપેલ બે એલિમેન્ટ્સને સ્વેપ કરવા પાયથોન કોડ લખો.

## જવાબ

```

1 # લિસ્ટમાં બે એલમિન્ટ્સને સ્વેપ કરવાનો પ્રોગ્રામ
2 def swap_elements(my_list, pos1, pos2):
3     # પોઝિશન્સ માન્ય છે કે નહીં તે તપારો
4     if 0 <= pos1 < len(my_list) and 0 <= pos2 < len(my_list):
5         # એલમિન્ટ્સ સ્વેપ કરો
6         my_list[pos1], my_list[pos2] = my_list[pos2], my_list[pos1]
7         return True
8     else:
9         return False

```

```

10 # ઉદાહરણ
11 numbers = [10, 20, 30, 40, 50]
12 print("મૂળ લિસ્ટ:", numbers)
13
14 # પોર્ટશિન 1 અને 3 પરના એલમિન્ટ્સ સ્વેપ કરો
15 if swap_elements(numbers, 1, 3):
16     print("સ્વેપ પછી:", numbers)
17 else:
18     print("અમાન્ય પોર્ટશિન્સ")
19

```

**મેમરી ટ્રીક****"SWAP"** (Select positions, Watch boundaries, Assign simultaneously, Print result)**પ્રશ્ન 5(બ) [4 ગુણ]**

પાયથોનનાં Math મોડ્યુલ અને random મોડ્યુલ ઉદાહરણનાં ઉપયોગ કરીને સમજાવો.

**જવાબ**

Math અને random મોડ્યુલ મેથેમેટિકલ ઓપરેશન્સ અને રેન્ડમ નંબર જનરેશન માટેના ફંક્શન્સ પ્રદાન કરે છે.

**Math મોડ્યુલ:**

```

1 import math
2 print(math.pi)      # આઉટપુટ: 3.14159...
3 print(math.sqrt(16)) # આઉટપુટ: 4.0
4 print(math.ceil(4.2)) # આઉટપુટ: 5

```

**Random મોડ્યુલ:**

```

1 import random
2 print(random.random())    # રેન્ડમ ફૂલોટ
3 print(random.randint(1, 10)) # રેન્ડમ ઇન્ટીજર
4 colors = ["red", "green"]
5 print(random.choice(colors)) # રેન્ડમ પસંદગી

```

**મેમરી ટ્રીક****"MR-CS"** (Math for Calculations, Random for Choice and Shuffling)**પ્રશ્ન 5(ક) [7 ગુણ]**

Tuple ફંક્શન અને ઓપરેશન દર્શાવવા પાયથોન કોડ લખો.

**જવાબ**

```

1 # Tuples બનાવવા
2 mixed_tuple = (1, "Hello", 3.14, True)
3
4 # એલમિન્ટ એક્સ્ટ્રોસ કરવા
5 print(mixed_tuple[0])    # આઉટપુટ: 1

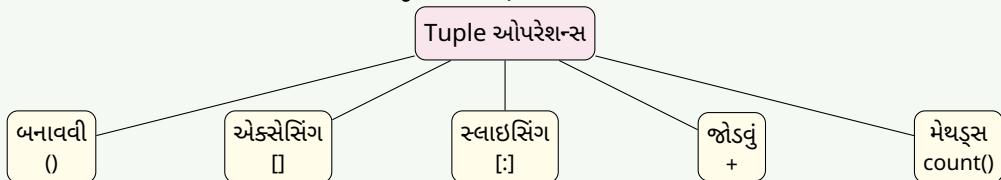
```

```

6 # Tuple સ્લાઇસને
7 print(mixed_tuple[1:3]) # આઉટપુટ: ("Hello", 3.14)
8
9 # Tuple જોડવા
10 tuple1 = (1, 2)
11 tuple2 = (3, 4)
12 print(tuple1 + tuple2) # આઉટપુટ: (1, 2, 3, 4)
13
14 # Tuple મેથ્ડ્સ
15 numbers = (1, 2, 2)
16 print(numbers.count(2)) # આઉટપુટ: 2
17
18 # મેમ્બરશપિ ટેસ્ટની
19 print(1 in numbers) # આઉટપુટ: True
20

```

### આકૃતિ 11. Tuple ઓપરેશન્સ



#### મેમરી ટ્રીક

"CASC-RUMTC" (Create, Access, Slice, Concatenate, Repeat, Use methods, Membership test, Tuple conversion)

OR

### પ્રશ્ન 5(અ) [3 ગુણ]

લિસ્ટમાં સામેલ એલિમેન્ટનો સરવાળો કરવા પાયથોન કોડ લખો.

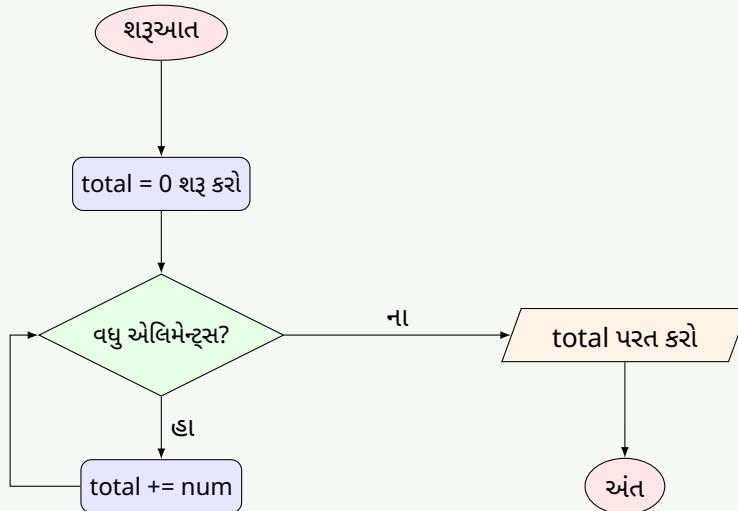
#### જવાબ

```

1 # લિસ્ટના એલિમેન્ટ્સનો સરવાળો કરવા માટેનો પ્રોગ્રામ
2 def sum_of_elements(numbers):
3     total = 0
4     for num in numbers:
5         total += num
6     return total
7
8 # ઉદાહરણ
9 my_list = [10, 20, 30, 40, 50]
10 print("સરવાળો:", sum_of_elements(my_list))

```

### આકૃતિ 12. સરવાળો માટે ફ્લોચાર્ટ



મેમરી ટ્રીક

"SITE" (Sum Initialized To zero, Elements added one by one)

OR

### પ્રશ્ન 5(બ) [4 ગુણ]

નીચે આપેલ built in functionsનો ઉપયોગ સમજાવો: 1) Print() 2) Min() 3) Sum() 4) Input()

#### જવાબ

કોષ્ટક 8. Built-in ફંક્શન્સ

| ફંક્શન  | હેતુ                             | ઉદાહરણ         |
|---------|----------------------------------|----------------|
| print() | કન્સોલ પર આઉટપુટ દર્શાવે છે      | print("Hi")    |
| min()   | સૌથી નાના આઇટમને પરત કરે છે      | min([5, 1])    |
| sum()   | તમામ આઇટમ્સનો સરવાળો આપે છે      | sum([1, 2])    |
| input() | વપરાશકર્તા પારોથી ઇનપુટ વાંચે છે | input("Naam:") |

```

1 print("Hello")
2 print(min([5, 3, 8]))
3 print(sum([1, 2, 3]))
4 name = input("Enter name: ")
  
```

મેમરી ટ્રીક

"PMSI" (Print to display, Min for smallest, Sum for total, Input for reading)

OR

### પ્રશ્ન 5(ક) [7 ગુણ]

સેટ ફંક્શન અને ઓપરેશન દર્શાવવા પાયથોન કોડ લખો.

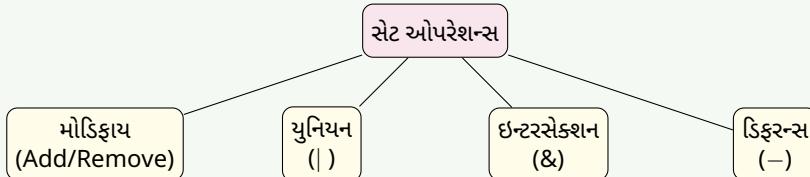
## જવાબ

```

1 # સેટ બનાવવા
2 numbers = {1, 2, 3}
3
4 # એલમેન્ટ ઉમેરવા
5 numbers.add(4)
6
7 # અપડેટ
8 numbers.update([5, 6])
9
10 # દૂર કરવા
11 numbers.remove(3)
12
13 # સેટ ઓપરેશન્સ
14 set1 = {1, 2, 3}
15 set2 = {3, 4, 5}
16
17 # ચુનાવિન
18 print(set1 | set2)    # આઉટપુટ: {1, 2, 3, 4, 5}
19
20 # ઇન્ટરસેક્શન
21 print(set1 & set2)    # આઉટપુટ: {3}
22
23 # સફ્ટન્સ
24 print(set1 - set2)    # આઉટપુટ: {1, 2}

```

### આફ્ટિ 13. સેટ ઓપરેશન્સ



## મેમરી ટ્રીક

"CARDS-UI" (Create, Add, Remove, Discard elements, Set operations - Union, Intersection)