પાયથોન પ્રોગ્રામિંગ (1323203) - સમર 2023 સોલ્યુશન

Milav Dabgar

૦૯ ઓગસ્ટ, ૨૦૨૩

પ્રશ્ન 1(અ) [3 ગુણ]

અલગોરીધમ વ્યાખ્યાયિત કરો. અલગોરીધમનાં ફાયદા શું છે?

જવાબ

અલગોરીધમ એ ચોક્કસ સમસ્યાને ઉકેલવા માટે પગલાઓના ક્રમબદ્ધ સમૂહ અથવા નિયમોનો સેટ છે.
અલગોરીધમના ફાયદા:

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

મેમરી ટ્રીક

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

પ્રશ્ન 1(બ) [4 ગુણ]

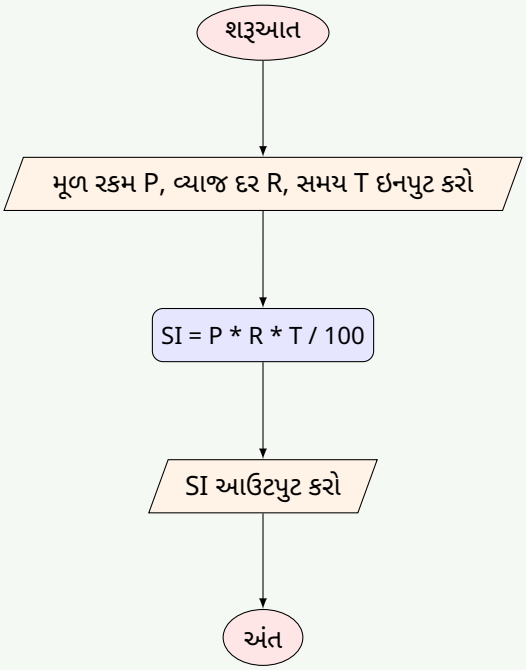
ફ્લોચાર્ટનો ઉપયોગ કરીને સમસ્યા ઉકેલવાના નિયમો શું છે? સાદું વ્યાજ શોધવા માટેનો ફ્લોચાર્ટ ડીઝાઈન કરો.

જવાબ

ફ્લોચાર્ટનો ઉપયોગ કરીને સમસ્યા ઉકેલવાના નિયમો:

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

આકૃતિ 1. સાદું વ્યાજ ગણતરી માટેનો ફ્લોચાર્ટ



મેમરી ટ્રીક

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

પ્રશ્ન 1(ક) [7 ગુણ]

પાયથોનનાં અસાઇમેંટ ઓપરટેરની યાદી બનાવો અને કોઈપણ ત્રણ અસાઇમેંટ ઓપરટેરોની કામગીરી દશાર્વવા માટે પાયથોન કોડ બનાવો.

જવાબ

પાયથોન અસાઇમેંટ ઓપરેટર્સ:

કોષ્ટક 1. અસાઇમેંટ 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

અસાઇમેંટ ઓપરેટર્સ દર્શાવતો કોડ:

```python
# અસાઇમેંટ ઓપરેટર્સનું પ્રદર્શન
num = 10
print("પ્રારંભિક મૂલ્ય:", num)

# += ઓપરેટરનો ઉપયોગ
num += 5
print("+= 5 પછી:", num)  # આઉટપુટ: 15

# -= ઓપરેટરનો ઉપયોગ
num -= 3
print("-= 3 પછી:", num)  # આઉટપુટ: 12

# *= ઓપરેટરનો ઉપયોગ
num *= 2
print("*= 2 પછી:", num)  # આઉટપુટ: 24
```

મેમરી ટ્રીક

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

OR

પ્રશ્ન 1(ક) [7 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 (પૂર્ણાંક 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-વેલ્યુ જોડી	{"name": "John"}
complex	કોમ્પ્લેક્સ નંબર	2+3j
NoneType	None દર્શાવે છે	None

ત્રણ ડેટા ટાઇપ્સ ઓળખવા માટેનો કોડ:

```python
# ડેટા ટાઇપ્સ ઓળખવાનો પ્રોગ્રામ
def identify_data_type(value):
    data_type = type(value).__name__
    print(f"મૂલ્ય: {value}")
    print(f"ડેટા ટાઇપ: {data_type}")
    print("-" * 20)

# 3 અલગઅલગ- ડેટા ટાઇપ્સ સાથે ટેસ્ટિંગ
```

```
identify_data_type(42)          # Integer
identify_data_type(3.14)        # Float
identify_data_type("Hello World") # String

# આઉટપુટ:
# મૂલ્ય: 42
# ડેટા ટાઇપ: int
# ----------
# મૂલ્ય: 3.14
# ડેટા ટાઇપ: float
# ----------
# મૂલ્ય: Hello World
# ડેટા ટાઇપ: str
# ----------
```

મેમરી ટ્રીક

"TYPE-ID" (Tell Your Python Elements - Identify Data)

પ્રશ્ન 2(અ) [3 ગુણ]

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

જવાબ

સ્યુડોકોડ એ એલ્ગોરિધમનું ઉચ્ચ-સ્તરીય વર્ણન છે જે પ્રોગ્રામિંગ ભાષાના માળખાકીય સંકેતોનો ઉપયોગ કરે છે પરંતુ મશીન વાંચન કરતાં માનવ વાંચન માટે ડિઝાઇન કરેલ છે.
બે સંખ્યાઓમાંથી સૌથી નાની શોધવા માટે સ્યુડોકોડ:

```
BEGIN
   INPUT first_number, second_number
   IF first_number < second_number THEN
      smallest = first_number
   ELSE
      smallest = second_number
   END IF
   OUTPUT smallest
END
```

મેમરી ટ્રીક

"RISE" (Read Input, Select smallest, Echo result)

પ્રશ્ન 2(બ) [4 ગુણ]

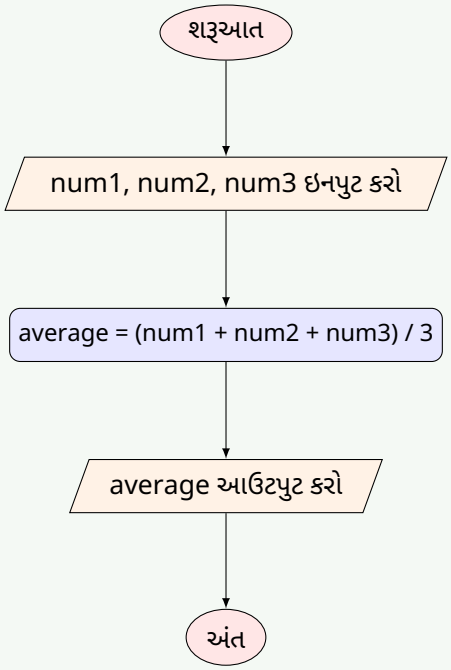
યુઝર્સ પાસેથી ત્રણ ઇનપુટ વાંચો અને સંખ્યાઓની સરેરાશ શોધવા માટેનો પાયથોન કોડ વિકસાવો.

જવાબ

```
# ત્રણ સંખ્યાઓની સરેરાશ ગણવા માટેનો પ્રોગ્રામ
num1 = float(input("પ્રથમ સંખ્યા દાખલ કરો: "))
num2 = float(input("બીજી સંખ્યા દાખલ કરો: "))
```

```python
num3 = float(input("ત્રીજી સંખ્યા દાખલ કરો: "))

# સરેરાશની ગણતરી
average = (num1 + num2 + num3) / 3

# પરિણામ દર્શાવો
print(f"{num1}, {num2}, અને {num3ની} 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]

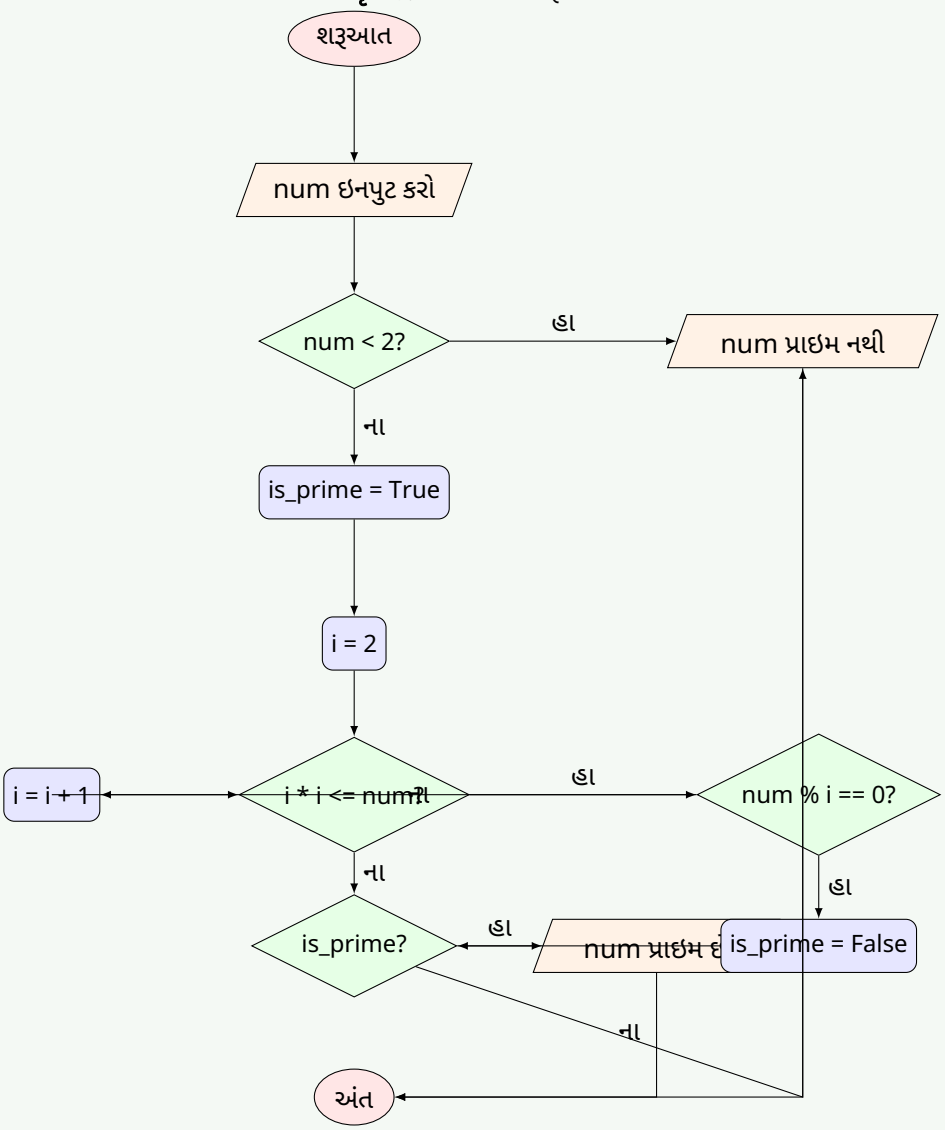
દાખલ કરેલ સંખ્યા prime છે કે નહીં તે બતાવવા પાયથોન કોડ લખો.

જવાબ

```python
# સંખ્યા પ્રાઇમ છે કે નહી તે તપાસવાનો પ્રોગ્રામ
num = int(input("એક સંખ્યા દાખલ કરો: "))

# થી2 ઓછી સંખ્યા છે કે નહી તપાસો
if num < 2:
    print(f"{num} એક પ્રાઇમ સંખ્યા નથી")
else:
    # is_prime ને True તરીકે આરંભો
    is_prime = True

    # 2 થી sqrt(num) સુધી તપાસો
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
```

```
        is_prime = False
        break

# પરિણામ દર્શાવો
if is_prime:
    print(f"{num} એક પ્રાઇમ સંખ્યા છે")
else:
    print(f"{num} એક પ્રાઇમ સંખ્યા નથી")
```

આકૃતિ 3. પ્રાઇમ ચેક માટે ફ્લોચાર્ટ



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 એલ્ગોરિધમ વચ્ચેનો તફાવત લખો.

જવાબ

કોષ્ટક 3. ફ્લોચાર્ટ અને એલ્ગોરિધમ વચ્ચેનો તફાવત

ફ્લોચાર્ટ	એલ્ગોરિધમ
માનક સિમ્બોલ અને આકારોનો ઉપયોગ કરીને **દૃશ્ય પ્રતિનિધિત્વ**	**લેખિત વર્ણન** માળખાકીય ભાષાનો ઉપયોગ કરીને
ગ્રાફિકલ પ્રકૃતિને કારણે **સમજવું સરળ**	સિન્ટેક્સ અને શબ્દાવલીનું જ્ઞાન જરૂરી
તાર્કિક પ્રવાહ અને સંબંધોને સ્પષ્ટ રીતે દર્શાવે	ક્રમિક ક્રમમાં **વિગતવાર પગલાં** પ્રદાન કરે
બનાવવા માટે **સમય-લેતી** પરંતુ સમજવા માટે સરળ	**ઝડપથી 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

```
x=10
y=2
print (x*y)
print (x ** y)
print (x//y)
print (x % y)
```

જવાબ:

કોષ્ટક 4. આઉટપુટ સમજૂતી

ઓપરેશન	સમજૂતી	આઉટપુટ
x*y	ગુણાકાર: 10×2	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) [7 ગુણ]

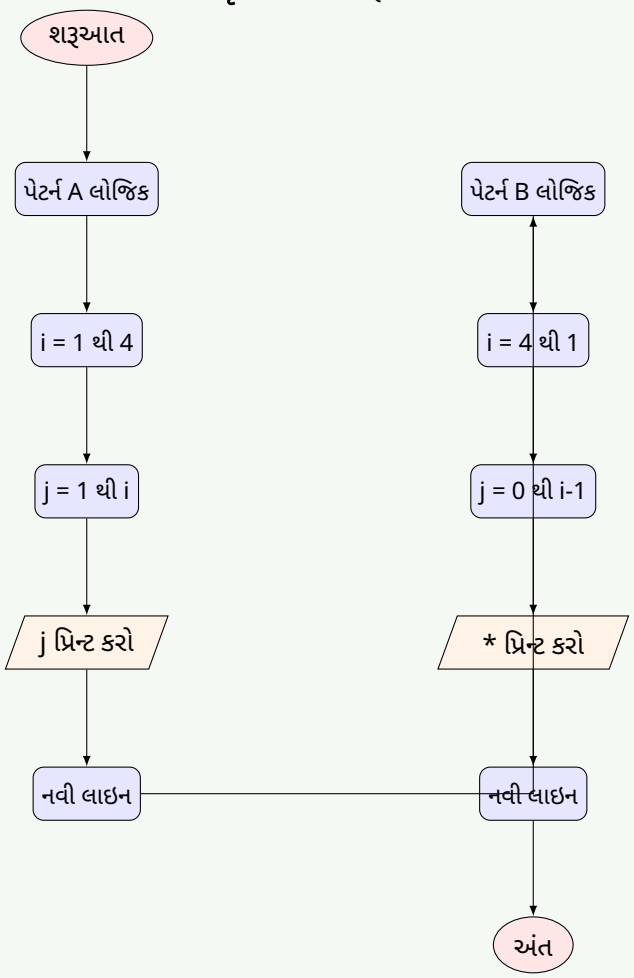
નીચેની પેટર્ન દર્શાવવા પાયથોન કોડ લખો:

જવાબ

```
A)              B)
1               * * * *
1 2             * * *
1 2 3           * *
1 2 3 4         *
```

```python
# પેટર્ન A: સંખ્યા પેટર્ન
print("પેટર્ન A:")
for i in range(1, 5):
    for j in range(1, i + 1):
        print(j, end=" ")
    print()

# પેટર્ન B: તારા પેટર્ન
print("\પેટર્નn B:")
for i in range(4, 0, -1):
    for j in range(i):
        print("*", end=" ")
    print()
```

આકૃતિ 4. પેટર્ન માટે ફ્લોચાર્ટ



મેમરી ટ્રીક

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

પ્રશ્ન 3(અ) [3 ગુણ]

જૂરી ઉદાહરણો સાથે break statementનાં ઉપયોગનું વર્ણન કરો.

જવાબ

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

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

આઉટપુટ:

```
2 તપાસી રહ્યા છીએ
4 તપાસી રહ્યા છીએ
6 તપાસી રહ્યા છીએવિષમ
 સંખ્યા મળી: 7
```

મેમરી ટ્રીક

"EXIT" (EXecute until condition, Immediately Terminate)

પ્રશ્ન 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.
સિન્ટેક્સ:

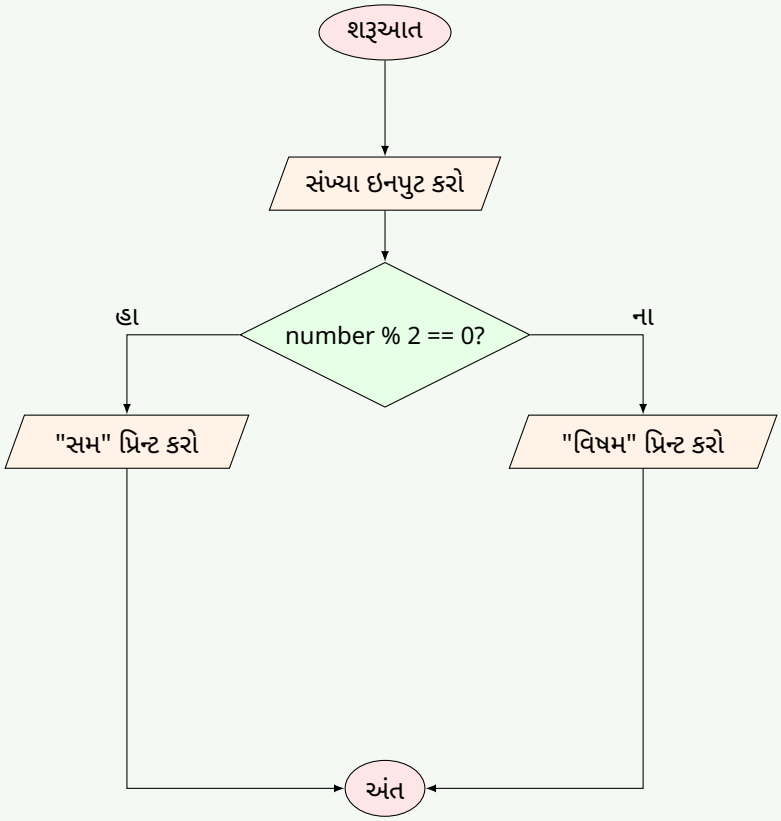
```
if શરત:
    # જો શરત True હોય તો આ કોડ એક્ઝિક્યુટ થશે
else:
    # જો શરત False હોય તો આ કોડ એક્ઝિક્યુટ થશે
```

ઉદાહરણ:

```
# સંખ્યા સમ છે કે વિષમ તે તપાસવું
number = int(input("એક સંખ્યા દાખલ કરો: "))

if number % 2 == 0:
    print(f"{number} એક સમ સંખ્યા છે")
```

```
else:
    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 ગુણ]

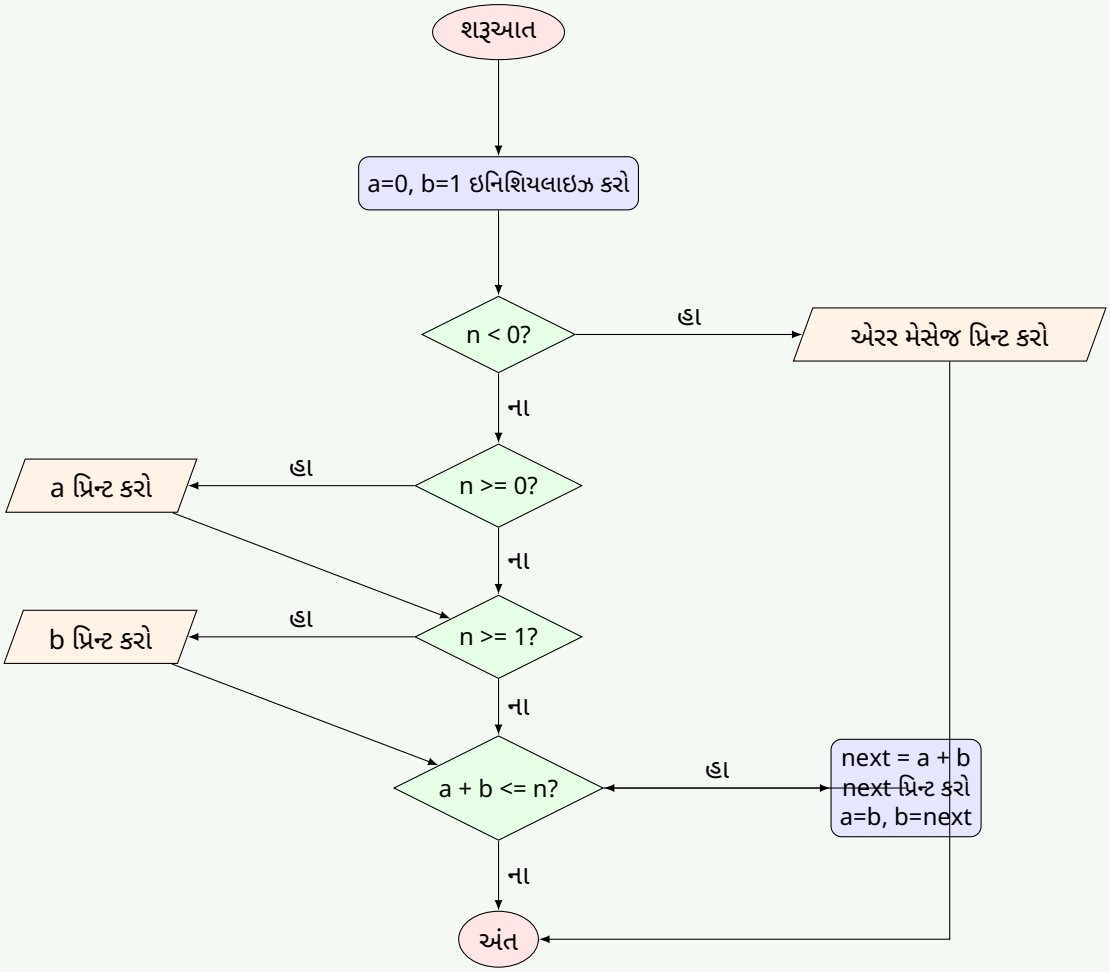
0 થી N સંખ્યા સુધીની ફીબોનાકી શ્રેણી પ્રિન્ટ કરવા માટે યુઝર ડેફાઇન ફંકશન બનાવો.

જવાબ

```
# ફીબોનાકી શ્રેણી પ્રિન્ટ કરવા માટેનું ફંકશન
def print_fibonacci(n):
    # પ્રથમ બે પદો ઇનિશિયલાઇઝ કરો
    a, b = 0, 1

    # n માન્ય છે કે નહીં તે તપાસો
    if n < 0:
        print("કૃપા કરીને એક હકારાત્મક સંખ્યા દાખલ કરો")
        return

    # ફીબોનાકી શ્રેણી પ્રિન્ટ કરો
    print(n, "સુધીની ફીબોનાકી શ્રેણી:")

```

```
    if n >= 0:
        print(a, end=" ")  # પ્રથમ પદ પ્રિન્ટ કરો

    if n >= 1:
        print(b, end=" ")  # બીજો પદ પ્રિન્ટ કરો

    # બાકીની શ્રેણી બનાવો અને પ્રિન્ટ કરો
    while a + b <= n:
        next_term = a + b
        print(next_term, end=" ")
        a, b = b, next_term

# ફંક્શનનું ટેસ્ટિંગ
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 statementનાં ઉપયોગનું વર્ણન કરો.

જવાબ

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

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

આઉટપુટ:

```
1
3
5
7
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, ટપલ, સ્ટ્રિંગ) પર ઇટરેશન કરવા માટે થાય છે.
સિન્ટેક્સ:

```
for વેરિએબલ in સિક્વન્સ:
    # દરેક આઇટમ માટે એક્ઝિક્યુટ થનાર કોડ
```

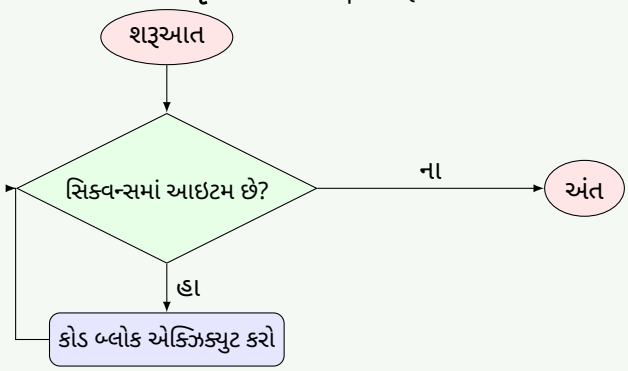
ઉદાહરણ:

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

આઉટપુટ:

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

આકૃતિ 7. For Loop માટે ફ્લોચાર્ટ



મેમરી ટ્રીક

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

OR

પ્રશ્ન 3(ક) [7 ગુણ]

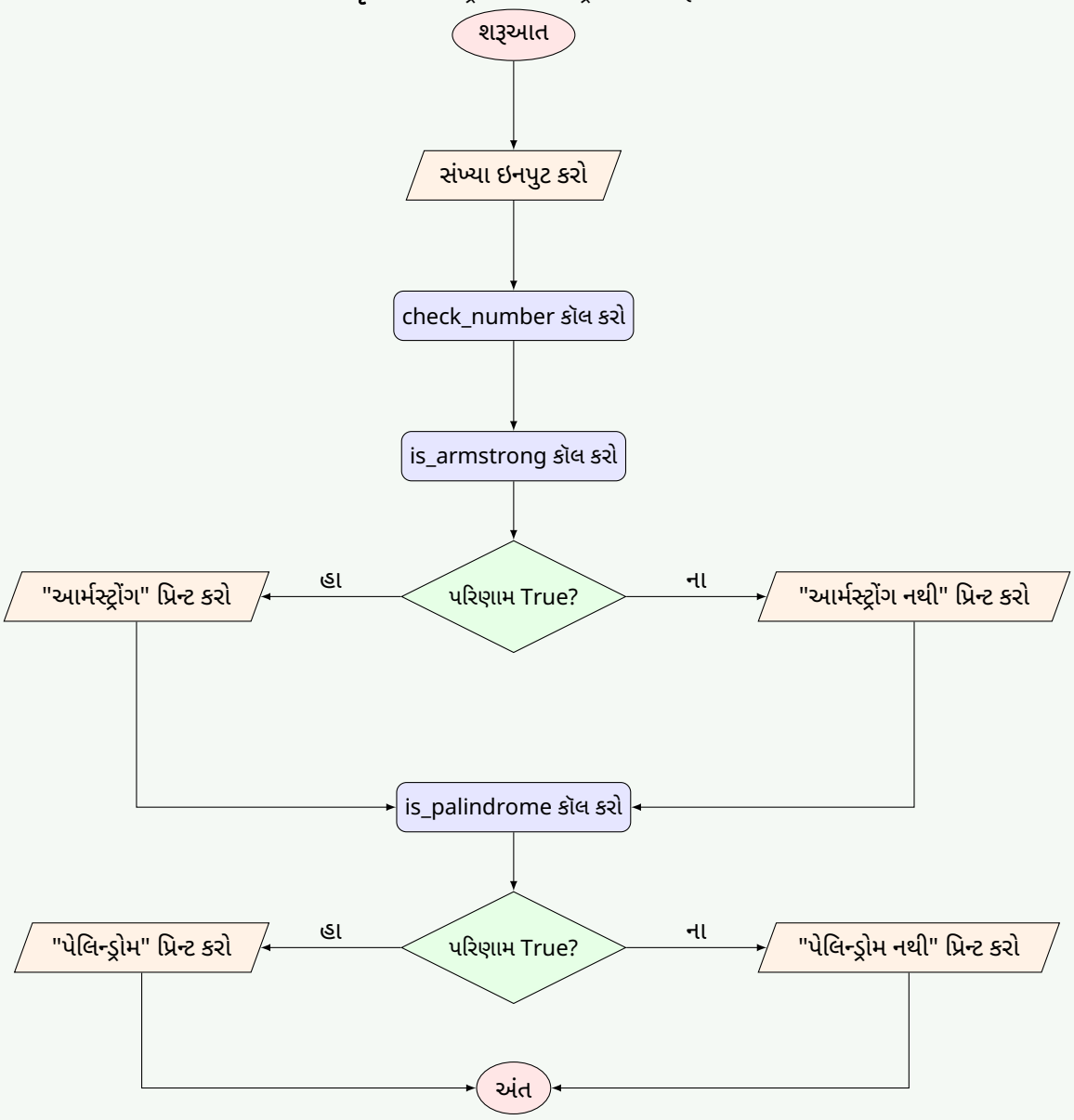
આપેલ નંબર આર્મસ્ટ્રોંગ નંબર છે કે પેલિન્ડ્રોમ તે નિર્ધારિત કરવા પાયથોન કોડ લખો.

જવાબ

```python
# સંખ્યા આર્મસ્ટ્રોંગ નંબર છે કે નહીં તે તપાસવાનું ફંકશન
def is_armstrong(num):
    # ડિજિટ્સની સંખ્યા ગણવા
    num_str = str(num)
    n = len(num_str)

    # દરેક ડિજિટને n ઘાત પર ઊંચકી તેનો સરવાળો કરો
    sum_of_powers = sum(int(digit) ** n for digit in num_str)

    # સરવાળો મૂળ સંખ્યા સાથે સરખાવો
    return sum_of_powers == num

# સંખ્યા પેલિન્ડ્રોમ છે કે નહીં તે તપાસવાનું ફંકશન
def is_palindrome(num):
    num_str = str(num)
    # સ્ટ્રિંગ તેના રિવર્સ સાથે સરખાવો
    return num_str == num_str[::-1]

# મુખ્ય ફંકશન
def check_number(num):
    if is_armstrong(num):
        print(f"{num} એક આર્મસ્ટ્રોંગ નંબર છે")
    else:
        print(f"{num} એક આર્મસ્ટ્રોંગ નંબર નથી")

    if is_palindrome(num):
        print(f"{num} એક પેલિન્ડ્રોમ છે")
    else:
        print(f"{num} એક પેલિન્ડ્રોમ નથી")

```

```
# ફંકશનનું ટેસ્ટિંગ
number = int(input("એક સંખ્યા દાખલ કરો: "))
check_number(number)
```

આકૃતિ 8. આર્મસ્ટ્રોંગ અને પેલિન્ડ્રોમ ચેક માટે ફ્લોચાર્ટ



મેમરી ટ્રીક

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

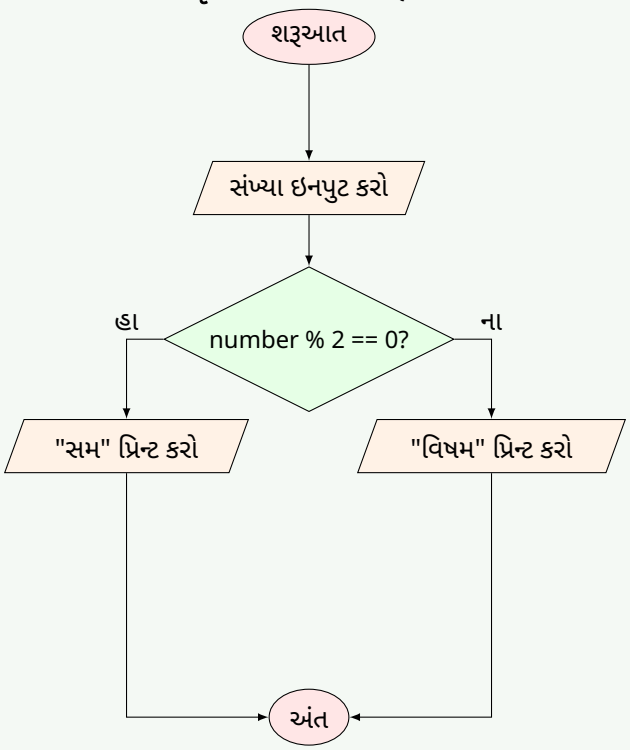
પ્રશ્ન 4(અ) [3 ગુણ]

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

જવાબ

```python
# સંખ્યા સમ છે કે વિષમ તે તપાસવાનો પ્રોગ્રામ
number = int(input("એક સંખ્યા દાખલ કરો: "))

if number % 2 == 0:
    print(f"{number} એક સમ સંખ્યા છે")
else:
    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]

ફંકશનની વ્યાખ્યા આપો. યુઝર ડિફાઇન ફંકશન યોગ્ય ઉદાહરણ આપી સમજાવો.

જવાબ

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

યુઝર-ડિફાઇન ફંકશનના ઘટકો:

- **def કીવર્ડ**: ફંકશન વ્યાખ્યાની શરૂઆત દર્શાવે છે
- **ફંકશન નામ**: ફંકશન માટે ઓળખકર્તા
- **પેરામીટર્સ**: ઇનપુટ વેલ્યુઝ (વૈકલ્પિક)
- **ડોકસ્ટ્રિંગ**: ફંકશનનું વર્ણન (વૈકલ્પિક)
- **ફંકશન બોડી**: એક્ઝિક્યુટ થનાર કોડ
- **રિટર્ન સ્ટેટમેન્ટ**: આઉટપુટ વેલ્યુ (વૈકલ્પિક)

ઉદાહરણ:

```
# લંબચોરસનું ક્ષેત્રફળ ગણવા માટેનું યુઝરડફિાઇન- ફંકશન
def calculate_area(length, width):
    """
    લંબચોરસનું ક્ષેત્રફળ ગણે છે
    """
    area = length * width
    return area

# ફંકશન કૉલ કરો
result = calculate_area(5, 3)
print(f"લંબચોરસનું ક્ષેત્રફળ: {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 એક્સટ્રેક્ટ કરવો
Methods	બિલ્ટ-ઇન ફંકશન્સ (len, upper, lower, વગેરે)
Membership Testing	સ્ટ્રિંગમાં સબસ્ટ્રિંગ છે કે નહીં તે તપાસવું
Formatting	ફોર્મેટેડ સ્ટ્રિંગ્સ બનાવવી
Escape Sequences	\થી શરૂ થતા સ્પેશિયલ કેરેક્ટર્સ

ઉદાહરણ:

1. સ્ટ્રિંગ Concatenation:

```
first_name = "John"
last_name = "Doe"
full_name = first_name + " " + last_name
print(full_name)  # આઉટપુટ: John Doe
```

2. સ્ટ્રિંગ Slicing:

```
message = "Python Programming"
print(message[0:6])   # આઉટપુટ: Python
print(message[7:])    # આઉટપુટ: Programming
print(message[-11:])  # આઉટપુટ: Programming
```

3. સ્ટ્રિંગ Methods:

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

મેમરી ટ્રીક

"CSM" (Concatenate strings, Slice portions, Manipulate with methods)

OR

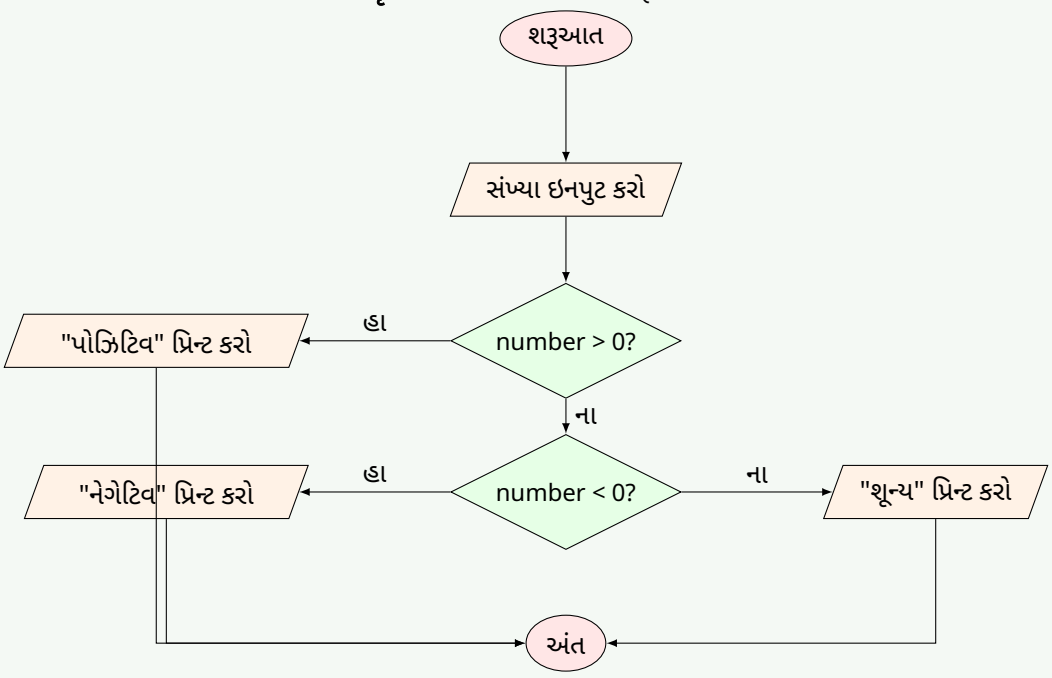
પ્રશ્ન 4(અ) [3 ગુણ]

પોઝિટિવ અને નેગેટિવ નંબર તપાસવા પાયથોન કોડ બનાવો.

જવાબ

```python
# સંખ્યા પોઝિટિવ છે કે નેગેટિવ તે તપાસવાનો પ્રોગ્રામ
number = float(input("એક સંખ્યા દાખલ કરો: "))

if number > 0:
    print(f"{number} એક પોઝિટિવ સંખ્યા છે")
elif number < 0:
    print(f"{number} એક નેગેટિવ સંખ્યા છે")
else:
    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	ફંકશનની અંદર વ્યાખ્યાયિત અને માત્ર તે ફંકશનની અંદર જ એક્સેસિબલ
Global Variable	ફંકશનની બહાર વ્યાખ્યાયિત અને પ્રોગ્રામના તમામ ભાગમાં એક્સેસિબલ

ઉદાહરણ:

```
# Global વેરએિબલ
count = 0

def update_count():
    # Local વેરએિબલ
    local_var = 5

    # ફંકશનની અંદર Global વેરએિબલ એક્સેસ કરવો
    global count
    count += 1

    print(f"Local: {local_var}")
    print(f"Global (inside): {count}")

# ફંકશન કૉલ કરો
update_count()

# ફંકશનની બહાર વેરએિબલ એક્સેસ કરવા
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 ત્રણ ઉદાહરણનો ઉપયોગ કરીને સમજાવો.

જવાબ

પાયથોનમાં લિસ્ટ ઓપરેશન્સ:

કોષ્ટક 7. લિસ્ટ ઓપરેશન્સ

ઓપરેશન	વર્ણન
લિસ્ટ બનાવવી	સ્ક્વેર બ્રેકેટ્સ [] નો ઉપયોગ
ઇન્ડેક્સિંગ	પોઝિશન દ્વારા એલિમેન્ટ એક્સેસ કરવા
સ્લાઇસિંગ	લિસ્ટના ભાગો એક્સટ્રેક્ટ કરવા
એપેન્ડ	છેલ્લે એલિમેન્ટ ઉમેરવા
ઇન્સર્ટ	ચોક્કસ પોઝિશન પર એલિમેન્ટ ઉમેરવા
રિમૂવ	ચોક્કસ એલિમેન્ટ દૂર કરવા
પોપ	એલિમેન્ટ દૂર કરવું અને પાછું મેળવવું
સોર્ટ	લિસ્ટ એલિમેન્ટ્સ ઓર્ડર કરવા
રિવર્સ	લિસ્ટનો ક્રમ ઊલટાવવો
એક્સ્ટેન્ડ	લિસ્ટ્સ જોડવી

ઉદાહરણ:

1. લિસ્ટ ઇન્ડેક્સિંગ અને સ્લાઇસિંગ:

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

2. લિસ્ટ મેથડ્સ (append, insert, remove):

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

3. લિસ્ટ કોમ્પ્રિહેન્શન્સ:

```
# સ્ક્વેર્સની 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)

પ્રશ્ન 5(અ) [3 ગુણ]

લિસ્ટમાં આપેલ બે એલિમેન્ટ્સને સ્વેપ કરવા પાયથોન કોડ લખો.

જવાબ

```
# લિસ્ટમાં બે એલમિન્ટ્સને સ્વેપ કરવાનો પ્રોગ્રામ
def swap_elements(my_list, pos1, pos2):
    # પોઝિશન્સ માન્ય છે કે નહીં તે તપાસો
    if 0 <= pos1 < len(my_list) and 0 <= pos2 < len(my_list):
        # એલમિન્ટ્સ સ્વેપ કરો
        my_list[pos1], my_list[pos2] = my_list[pos2], my_list[pos1]
        return True
    else:
        return False
```

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

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 ફંકશન્સ 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 ફ્લોટ
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]

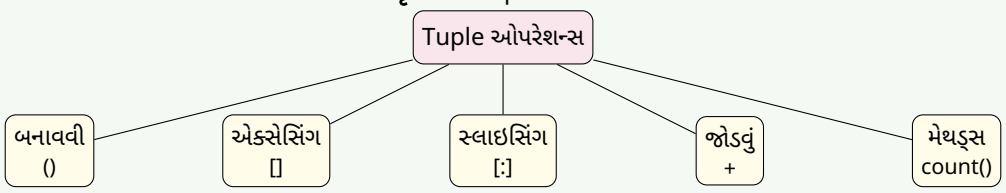
Tuple ફંકશન અને ઓપરેશન દર્શાવવા પાયથોન કોડ લખો.

જવાબ

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

```python

# Tuple સ્લાઇસિંગ
print(mixed_tuple[1:3])  # આઉટપુટ: ("Hello", 3.14)

# Tuple જોડવા
tuple1 = (1, 2)
tuple2 = (3, 4)
print(tuple1 + tuple2)  # આઉટપુટ: (1, 2, 3, 4)

# Tuple મેથડ્સ
numbers = (1, 2, 2)
print(numbers.count(2))  # આઉટપુટ: 2

# મેમ્બરશિપ ટેસ્ટિંગ
print(1 in numbers)  # આઉટપુટ: True
```

આકૃતિ 11. Tuple ઓપરેશન્સ



મેમરી ટ્રીક

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

OR

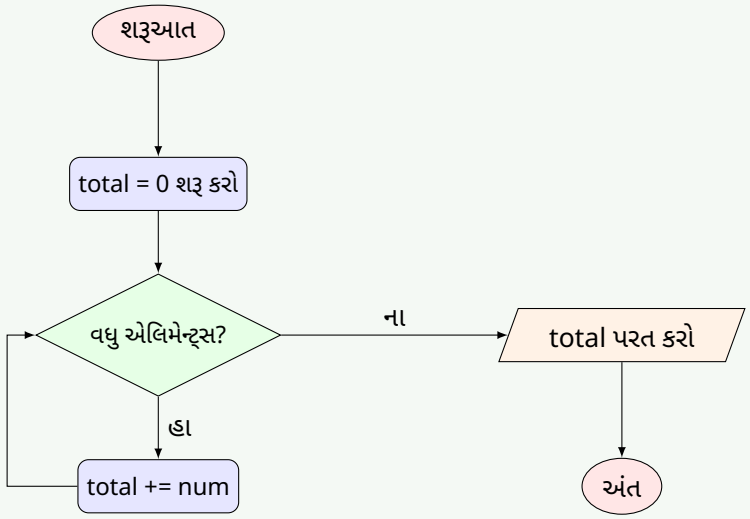
પ્રશ્ન 5(અ) [3 ગુણ]

લિસ્ટમાં સામેલ એલિમેંટનો સરવાળો કરવા પાયથોન કોડ લખો.

જવાબ

```python
# લિસ્ટના એલમિન્ટ્સનો સરવાળો કરવા માટેનો પ્રોગ્રામ
def sum_of_elements(numbers):
    total = 0
    for num in numbers:
        total += num
    return total

# ઉદાહરણ
my_list = [10, 20, 30, 40, 50]
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 ફંકશન્સ

ફંકશન	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 પાસેથી ઇનપુટ વાંચે છે	input("Naam:")

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

મેમરી ટ્રીક

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

OR

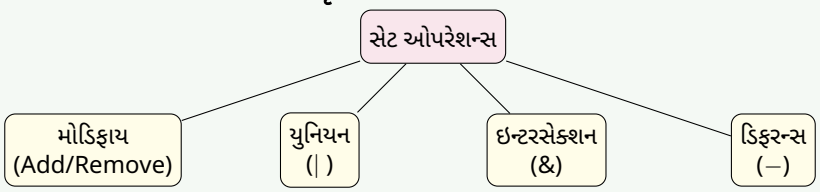
પ્રશ્ન 5(ક) [7 ગુણ]

સેટ ફંકશન અને ઓપરેશન દર્શાવવા પાયથોન કોડ લખો.

જવાબ

```
# સેટ બનાવવા
numbers = {1, 2, 3}

# એલમેિન્ટ ઉમેરવા
numbers.add(4)

# અપડેટ
numbers.update([5, 6])

# દૂર કરવા
numbers.remove(3)

# સેટ ઓપરેશન્સ
set1 = {1, 2, 3}
set2 = {3, 4, 5}

# યુનયિન
print(set1 | set2)     # આઉટપુટ: {1, 2, 3, 4, 5}

# ઇન્ટરસેક્શન
print(set1 & set2)     # આઉટપુટ: {3}

# ડફિરન્સ
print(set1 - set2)     # આઉટપુટ: {1, 2}
```

આકૃતિ 13. સેટ ઓપરેશન્સ



મેમરી ટ્રીક

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