

# પાયથોન પ્રોગ્રામિંગ (1323203) - શિયાળુ 2023 સોલ્યુશન

Milav Dabgar

20 જાન્યુઆરી, 2024

## પ્રશ્ન 1(a) [3 ગુણા]

આપેલ નંબર પોઝિટિવ છે કે નેગેટિવ તે તપાસવા માટે સ્થૂડો કોડ લખો

### જવાબ

Listing 1. Pseudocode for Number Check

```
1 BEGIN
2   Input number
3   IF number > 0 THEN
4     Display "Number is positive"
5   ELSE IF number < 0 THEN
6     Display "Number is negative"
7   ELSE
8     Display "Number is zero"
9   END IF
10  END
```

### મેમરી ટ્રીક

“શૂન્ય સાથે સરખાવો”

## પ્રશ્ન 1(b) [4 ગુણા]

એલ્ગોરિધમ વ્યાખ્યાયિત કરો અને ત્રણ નંબર માંથી મહત્તમ નંબર શોધવાનો એલ્ગોરિધમ બનાવો.

### જવાબ

**Algorithm વ્યાખ્યા:** એલ્ગોરિધમ એટલે ચોક્કસ સમસ્યાને ઉકેલવા માટે અથવા ગણતરી કરવા માટે બનાવેલ સ્ટેપ-બાય-સ્ટેપ પ્રક્રિયા અથવા નિયમોનો સેટ.

ત્રણ નંબરમાંથી મહત્તમ શોધવાનો એલ્ગોરિધમ:

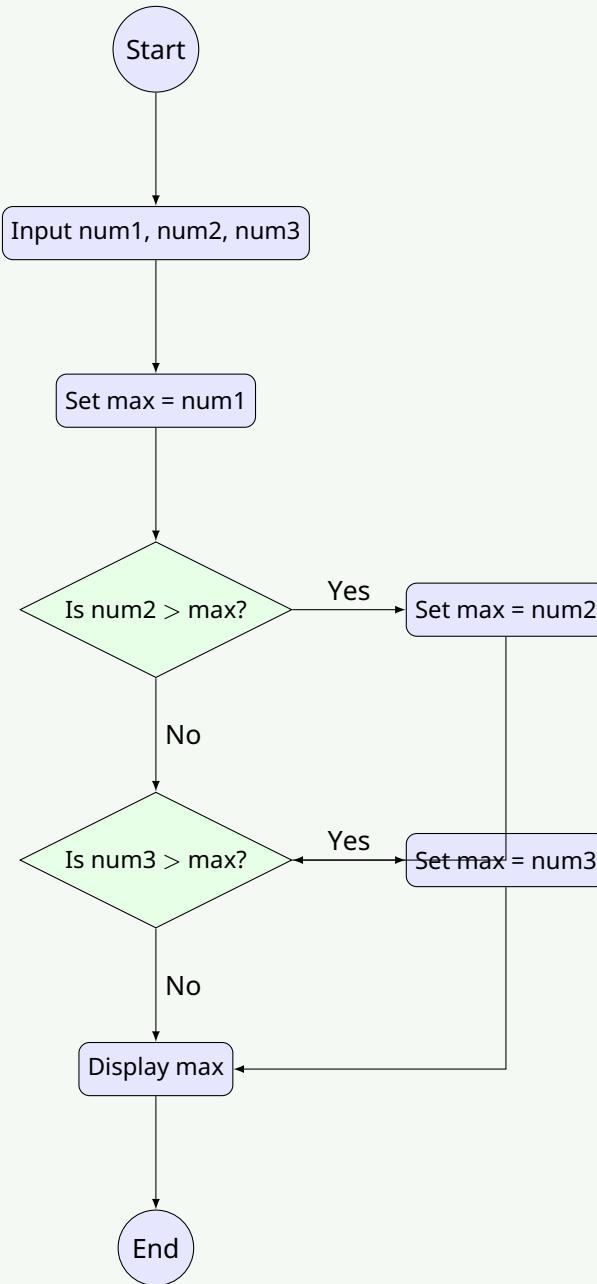
Listing 2. Algorithm for Maximum of Three Numbers

```
1 BEGIN
2   Input num1, num2, num3
3   Set max = num1
4   IF num2 > max THEN
5     Set max = num2
6   END IF
7   IF num3 > max THEN
8     Set max = num3
```

```

9   END IF
10  Display max
11  END

```



આકૃતિ 1. Flowchart to Find Maximum of Three Numbers

## મેમરી ટ્રીક

“સરખામણી અને બદલો”

## પ્રશ્ન 1(c) [7 ગુણ]

તાપમાન ના સોલિસ્યસ ને ફેરનહાઇટ માં કન્વર્ટ કરવાનો પાયથોન કોડ લખો.

## જવાબ

Listing 3. Celsius to Fahrenheit Conversion

```

1 # સેલ્સિયસથી ફેરનહાઇટ રૂપાંતરનો પ્રોગ્રામ
2
3 # યુઝર પાસેથી સેલ્સિયસ તાપમાન મેળવો
4 celsius = float(input("સેલ્સિયસમાં તાપમાન દાખલ કરો: "))
5
6 # સૂત્ર વાપરીને ફેરનહાઇટમાં રૂપાંતરતી કરો: F = (C * 9/5) + 32
7 fahrenheit = (celsius * 9/5) + 32
8
9 # પરિણામ દર્શાવો
10 print(f"\{celsius}\u00B0C એ {fahrenheit}\u00B0F ની બરાબર છે")

```

કોષ્ટક 1. તાપમાન રૂપાંતરણ

ધર્તક	વર્ણન
ઇનપુટ	સેલ્સિયસમાં તાપમાન
સૂત્ર	$F = (C \times 9/5) + 32$
આઉટપુટ	ફેરનહાઇટમાં તાપમાન

## મેમરી ટ્રીક

“9થી ગુણાકાર, 5થી ભાગાકાર, 32 બરાબરો”

## પ્રશ્ન 1(c OR) [7 ગુણ]

કંપેરિઝન ઓપરેટર નું લિસ્ટ આપો અને દરેકને પાયથોન કોડના ઉદાહરણ સાથે સમજાવો.

## જવાબ

કોષ્ટક 2. પાયથોન કંપેરિઝન ઓપરેટર્સ

ઓપરેટર	વર્ણન	ઉદાહરણ	પરિણામ
$==$	બરાબર છે	$5 == 5$	True
$!=$	બરાબર નથી	$5 != 6$	True
$>$	કરતાં મોટું	$6 > 3$	True
$<$	કરતાં નાનું	$3 < 6$	True
$\geq$	કરતાં મોટું અથવા બરાબર	$5 \geq 5$	True
$\leq$	કરતાં નાનું અથવા બરાબર	$5 \leq 5$	True

### Code Example:

Listing 4. Comparison Operators Example

```

1 # પાયથોન કંપેરિઝન ઓપરેટર્સ ઉદાહરણ
2 a = 10
3 b = 5
4
5 # બરાબર છે
6 print(f"\{a} == \{b}: {a == b}\") # False
7
8 # બરાબર નથી
9 print(f"\{a} != \{b}: {a != b}\") # True

```

```

10 # કરતાં મોટું
11 print(f"{a} > {b}: {a > b}") # True
12
13 # કરતાં નાનું
14 print(f"{a} < {b}: {a < b}") # False
15
16 # કરતાં મોટું અથવા બરાબર
17 print(f"{a} >= {b}: {a >= b}") # True
18
19 # કરતાં નાનું અથવા બરાબર
20 print(f"{a} <= {b}: {a <= b}") # False
21

```

### મેમરી ટ્રીક

“સરખાવો”

## પ્રશ્ન 2(a) [3 ગુણા]

પાયથોન ના ડેટા ટાઇપ સમજાવો.

### જવાબ

#### કોષ્ટક 3. પાયથોન ડેટા ટાઇપ્સ

ડેટા ટાઇપ	વર્ણન	ઉદાહરણ
<b>int</b>	પૂર્ણાંક મૂલ્યો	x = 10
<b>float</b>	દશાંશ બિંદુ મૂલ્યો	y = 10.5
<b>str</b>	ટેક્સ્ટ અથવા અક્ષર મૂલ્યો	name = "Python"
<b>bool</b>	તાર્કિક મૂલ્યો (True/False)	is_valid = True
<b>list</b>	ક્રમબદ્ધ, બદલી શકાય તેવો સંગ્રહ	nums = [1, 2, 3]
<b>tuple</b>	ક્રમબદ્ધ, ન બદલી શકાય તેવો સંગ્રહ	point = (5, 10)
<b>dict</b>	કી-વેલ્યુ જોડી	student = {"name": "John"}

### મેમરી ટ્રીક

“NIFTY SLD: નંબર્સ, ઇન્ટીજર્સ, ફ્લોટ્સ, ટેક્સ્ટ, યસ/નો, સીકવ-સીસ, લિસ્ટ્સ, ડિક્શનરીઝ”

## પ્રશ્ન 2(b) [4 ગુણા]

Nested If પાયથોન કોડ ના ઉદાહરણ સાથે સમજાવો.

### જવાબ

**Nested if:** એક conditional statement ની અંદર બીજું conditional statement લખવાને nested if કહેવામાં આવે છે. તે ઘણી શરતોને ક્રમમાં તપાસવાની મંજૂરી આપે છે.

#### Listing 5. Nested If Example

```

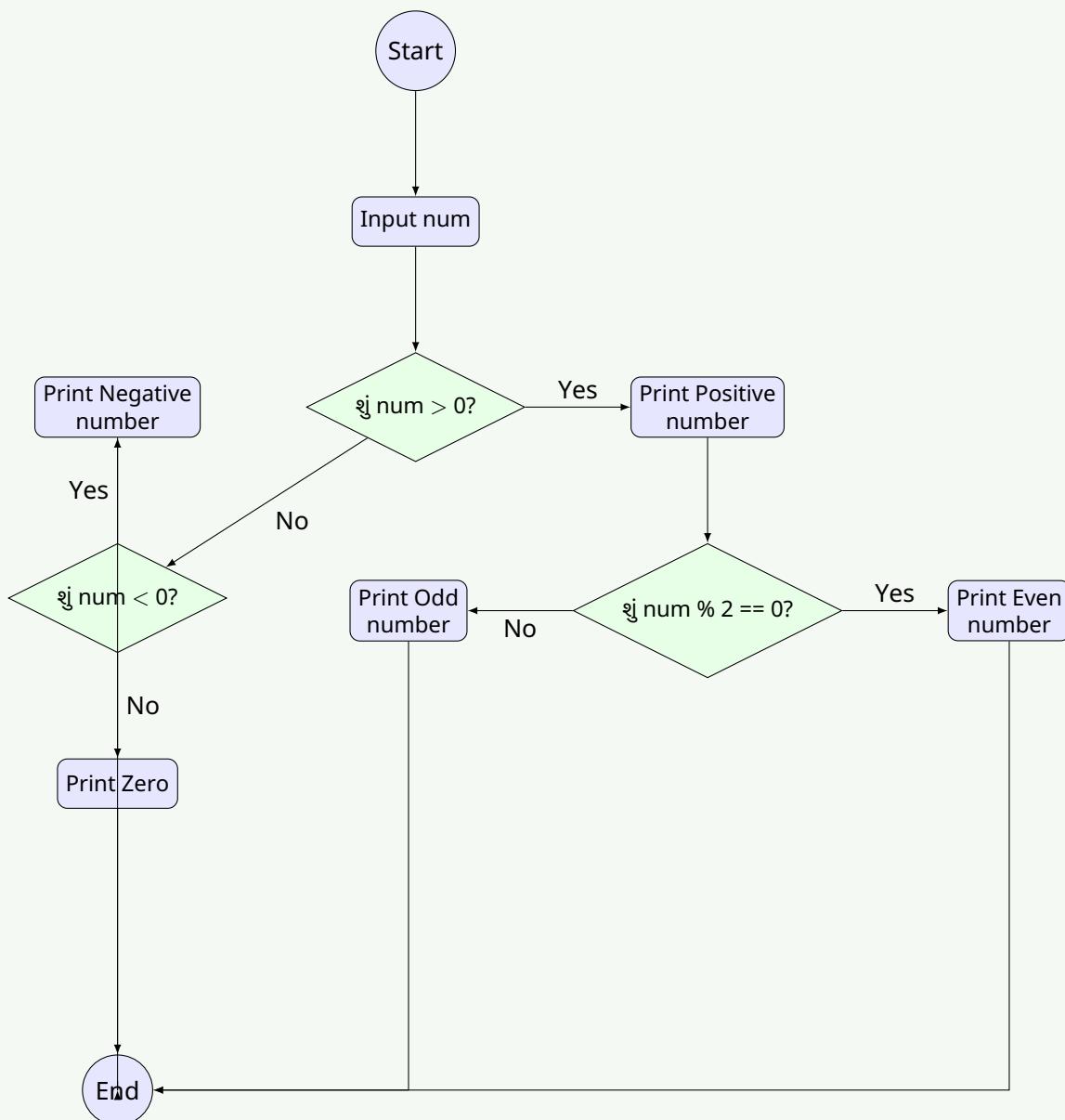
1 # નંબર પોઝિટિવિ, નેગેટિવિ કે શૂન્ય છે તે ચકાસવા માટેનો nested if ઉદાહરણ
2 # અને જો પોઝિટિવિ હોય, તો તે સમ છે કે વધિમ તે ચકાસો

```

```

3 num = int(input("ਏਕ ਨੰਬਰ ਦਾਖਲ ਕਰੋ: "))
4
5 if num > 0:
6     print("ਪੋਝਟਿਵ ਨੰਬਰ")
7     # nested if ਜੇ ਚਕਾਸੇ ਛੇ ਕੇ ਪੋਝਟਿਵ ਨੰਬਰ ਸਮ ਛੇ ਕੇ ਵਖਿਮ
8     if num % 2 == 0:
9         print("ਸਮ ਨੰਬਰ")
10    else:
11        print("ਵਖਿਮ ਨੰਬਰ")
12 elif num < 0:
13     print("ਨੇਗਟਿਵ ਨੰਬਰ")
14 else:
15     print("ਸ਼ੂਨ੍ਧ")
16

```



ਆਕਤਿ 2. Nested If Flowchart

## મેમરી ટ્રીક

"ચેક અંદર ચેક"

## પ્રક્રિયા 2(c) [7 ગુણ]

ઉદાહરણ સાથે વિવિધ પ્રકારના પસંદગી/નિષ્ણય લેવાના ફલો-ઓફ-કંટ્રોલ સ્ટ્રક્ચર ઉપયોગ સમજાવો.

## જવાબ

## કોષ્ટક 4. પાયથોનમાં સિલેક્શન કંટ્રોલ સ્ટ્રક્ચર્સ

સ્ટ્રક્ચર	હેતુ	વપરાશ
if	શરત સાચી હોય ત્યારે કોડ ચલાવવા	સરળ શરત ચકાસણી
if-else	સાચી શરત માટે એક કોડ, ખોટી માટે બીજો	દ્વિ નિષ્ણય લેવા
if-elif-else	ઘણી શરતો ચકાસવી	ઘણા સંભવિત પરિણામો
Nested if	શરત અંદર બીજી શરત	જટિલ શ્રોણીબદ્ધ નિર્ણયો
Ternary operator	એક લાઇન if-else	સરળ શરતી નિયુક્તિ

## Code Example:

## Listing 6. Selection Structures Example

```

1 # વિવિધ સલિક્ષણ સ્ટ્રક્ચર્સનું ઉદાહરણ
2 score = int(input("તમારો સ્કોર દાખલ કરો: "))
3
4 # સાંદુરું if
5 if score >= 90:
6     print("ઉત્તુતમા!")
7
8 # if-else
9 if score >= 60:
10    print("તમે પાસ થયા છો.")
11 else:
12    print("તમે નાપાસ થયા છો.")
13
14 # if-elif-else
15 if score >= 90:
16     grade = "A"
17 elif score >= 80:
18     grade = "B"
19 elif score >= 70:
20     grade = "C"
21 elif score >= 60:
22     grade = "D"
23 else:
24     grade = "F"
25 print(f"તમારો ગ્રેડ {grade} છે")
26
27 # Ternary operator
28 result = "પાસ" if score >= 60 else "નાપાસ"
29 print(result)

```

## મેમરી ટ્રીક

“SCENE: સિમ્પલ if, કન્ડિશન્સ વિથ else, Elif ફોર માટિયલ, Nested ફોર કોમ્પ્લેક્સ, એક્સપ્રેસ વિથ ટર્નરી”

## પ્રશ્ન 2(a OR) [3 ગુણ]

વેરિએબલ વ્યાખ્યાપિત કરવાના નિયમો લિસ્ટ કરો.

### જવાબ

કોષ્ટક 5. પાયથોનમાં વેરિએબલ્સ વ્યાખ્યાપિત કરવાના નિયમો

નિયમ	વર્ણન	ઉદાહરણ
અક્ષર અથવા અન્ડરસ્કોરથી શરૂ કરો	પ્રથમ અક્ષર એક લેટર અથવા અન્ડરસ્કોર હોવો જોઈએ	name = "John", _count = 10
કોઈ ખાસ અક્ષરો નહીં	માત્ર અક્ષરો, અંકો અને અન્ડરસ્કોર માન્ય	user_name (માન્ય), user-name (અમાન્ય)
કેસ રોન્સિટિવ	મોટા અક્ષરો અને નાના અક્ષરો અલગ	age અને Age અલગ વેરિએબલ્સ છે
રિઝર્વ કીવર્ડ્સ નહીં	પાયથોન કીવર્ડ્સને વેરિએબલ નામ તરીકે ઉપયોગ ન કરી શકાય	if, for, while, વગેરે
સ્પેસ નહીં	સ્પેસને બદલે અન્ડરસ્કોર વાપરો	first_name (first name નહીં)

### મેમરી ટ્રીક

"SILKS: શરૂઆત ચોગ્ય રીતે, ઇચ્છાર સ્પેશિયલ કેરેક્ટર, લૂક એટ કેસ, કીવર્ડ્સ અવોઇડ, સ્પેસ નોટ અલાઉન્ડ"

## પ્રશ્ન 2(b OR) [4 ગુણ]

ફોર લૂપ ને જરૂરી ઉદાહરણ સાથે સમજાવો.

### જવાબ

પાયથોનમાં For Loop: for લૂપનો ઉપયોગ કોઈ sequence (લિસ્ટ, ટ્પલ, સ્ટ્રોંગ) અથવા અન્ય iterable ઓફજેક્ટ પર પુનરાવર્તન કરવા માટે થાય છે. તે sequence ના દરેક આઇટમ માટે કોડનો એક બ્લોક ચલાવે છે.

Listing 7. For Loop Example

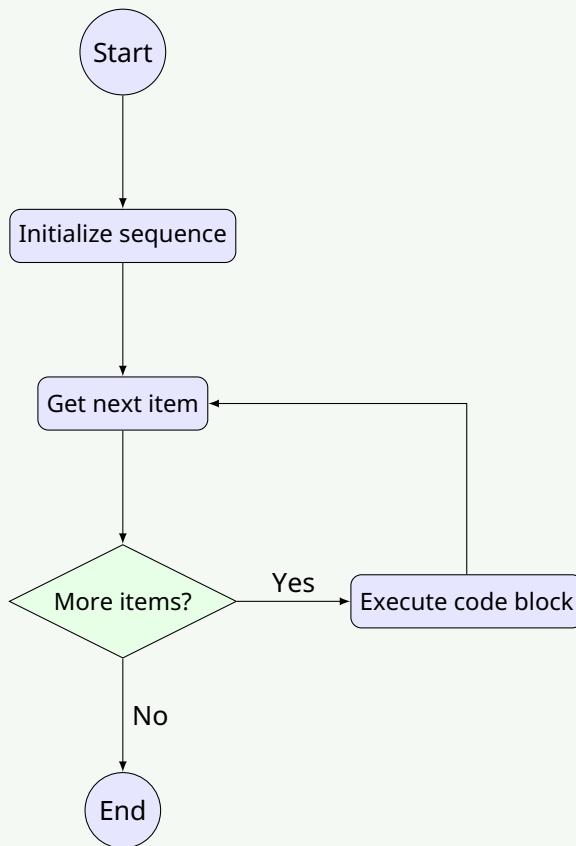
```

1 # પાયથોનમાં for લૂપનો ઉદાહરણ
2 # લસ્ટના દરેક એલમેન્ટને પ્રનિટ કરવા
3 fruits = ["apple", "banana", "cherry"]
4 for fruit in fruits:
5     print(fruit)

6
7 # range ફંક્શનનો for લૂપ સાથે ઉપયોગ
8 print("1 થી 5 સુધીના નંબર:")
9 for i in range(1, 6):
10    print(i)

11
12 # સ્ટ્રાંગ સાથે for લૂપનો ઉપયોગ
13 name = "Python"
14 for char in name:
15    print(char)

```



આકૃતિ 3. For Loop Flowchart

## મેમરી ટ્રીક

“ITEM: દે઱ે સભ્ય પર પુનરાવર્તન કરો”

## પ્રશ્ન 2(c OR) [7 ગુણ]

Break અને continue સ્ટેટમેન્ટને સંક્ષિપ્તમાં સમજાવો.

## જવાબ

## કોષ્ટક 6. Break અને Continue સ્ટેટમેન્ટ્સ

ટેટમેન્ટ	હેતુ	અસર
<b>break</b>	લૂપમાંથી તરત જ બહાર નીકળો	વર્તમાન લૂપને અટકાવે છે અને લૂપ પછીના સ્ટેટમેન્ટ પર કંટ્રોલ ટ્રાન્સફર કરે છે
<b>continue</b>	વર્તમાન પુનરાવર્તન છોડી દો	લૂપના આગલા પુનરાવર્તન પર જાય છે, continue સ્ટેટમેન્ટ પછીના કોઈપણ કોડને છોડી દે છે

## Code Example:

## Listing 8. Break and Continue Example

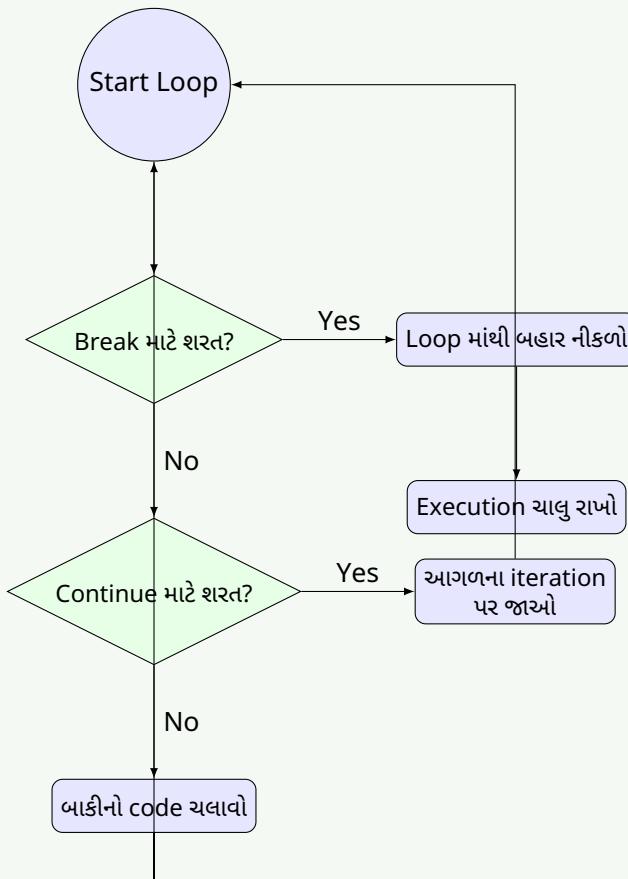
```

1 # Break સ્ટેટમેન્ટ ઉદાહરણ
2 print("Break ઉદાહરણ:")
3 for i in range(1, 11):
4     if i == 6:
  
```

```

5     print("i =", i, "પર લૂપ તોડીએ છીએ")
6     break
7     print(i, end=" ")
8     print("\nલૂપન સમાપ્ત થઈ")
9
10    # Continue સ્ટેટમેન્ટ ઉદાહરણ
11    print("\nContinue ઉદાહરણ:")
12    for i in range(1, 11):
13        if i % 2 == 0:
14            continue
15        print(i, end=" ")
16        print("\nમાત્રન વધિમ નંબરો પૂરનીટ થયા")

```



આકૃતિ 4. Break and Continue Flowchart

**મેમરી ટ્રીક****``EXIT SKIP: EXIT with break, SKIP with continue''****પ્રશ્ન 3(a) [3 ગુણ]**

1 થી 10 નંબર ને લૂપથી પ્રિન્ટ કરવા માટેનો પાયથન કોડ બનાવો.

## જવાબ

**Listing 9.** Printing 1 to 10

```

1 # 1 થી 10 સુધીના નંબર પ્રણિટ કરવા for લૂપનો ઉપયોગ
2 print("for લૂપનો ઉપયોગ કરીને:")
3 for i in range(1, 11):
4     print(i, end=" ")
5
6 print("\n\nwhile લૂપનો ઉપયોગ કરીને:")
7 # 1 થી 10 સુધીના નંબર પ્રણિટ કરવા while લૂપનો ઉપયોગ
8 counter = 1
9 while counter <= 10:
10     print(counter, end=" ")
11     counter += 1

```

**કોષ્ટક 7.** લૂપ અભિગમ

અભિગમ	ફાયદો
range સાથે For લૂપ	સરળ, સંક્ષિપ્ત, આપોઆપ કાઉન્ટર મેનેજ કરે છે
While લૂપ	જટિલ શરતો માટે વધુ લવચીક

## મેમરી ટ્રીક

“COUNT UP: Counter દરેક પુનરાવર્તનમાં અપડેટ થાય છે”

## પ્રશ્ન 3(b) [4 ગુણ]

નીચેની પેટર્ન પ્રિન્ટ કરવા માટેનો પાયથન કોડ લખો:

\*  
\*\*  
\*\*\*  
\*\*\*\*  
\*\*\*\*\*

## જવાબ

**Listing 10.** Star Pattern Program

```

1 # for લૂપનો ઉપયોગ કરીને સ્ટાર પેટર્ન પ્રણિટ કરો
2 rows = 5
3
4 for i in range(1, rows + 1):
5     # દરેક રો માં i જેટલા સ્ટાર પ્રણિટ કરો
6     print("*" * i)

```

વૈકલ્પિક ઉકેલ નેસ્ટેડ લૂપસ સાથે:

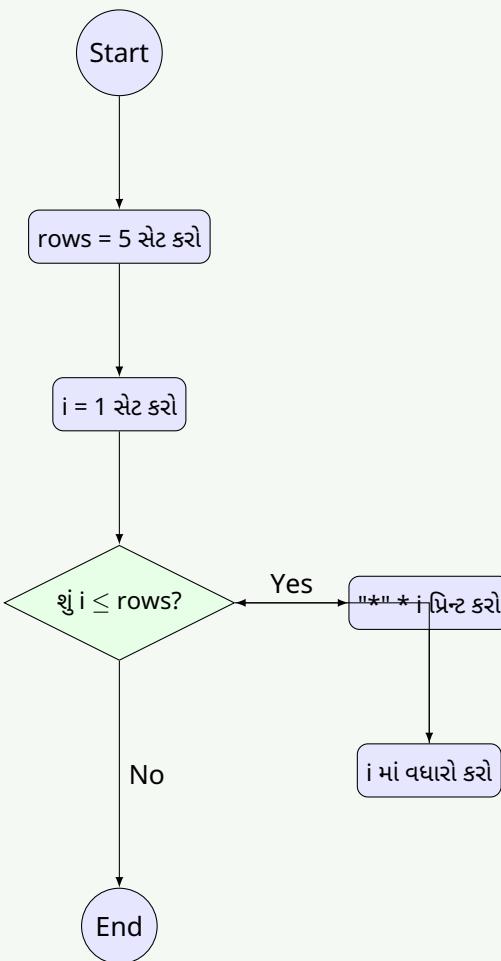
**Listing 11.** Star Pattern Nested Loop

```

1 # નેસ્ટેડ લૂપસનો ઉપયોગ કરીને સ્ટાર પેટર્ન પરણિટ કરો
2 rows = 5
3
4 for i in range(1, rows + 1):
5     for j in range(1, i + 1):
6         print("*", end="")

```

7 print() # દરેક રો પછી ન્યુલ લાઇન



આકૃતિ 5. Pattern Printing Flowchart

### મેમરી ટ્રીક

“RISE UP: રો વધે છે, સ્ટાર ઉપર તરફ વિસ્તરે છે”

## પ્રશ્ન 3(c) [7 ગુણ]

આપેલા નંબર નો factorial શોધવા માટેનું યુઝર ડિફાઈન ફંક્શન બનાવો.

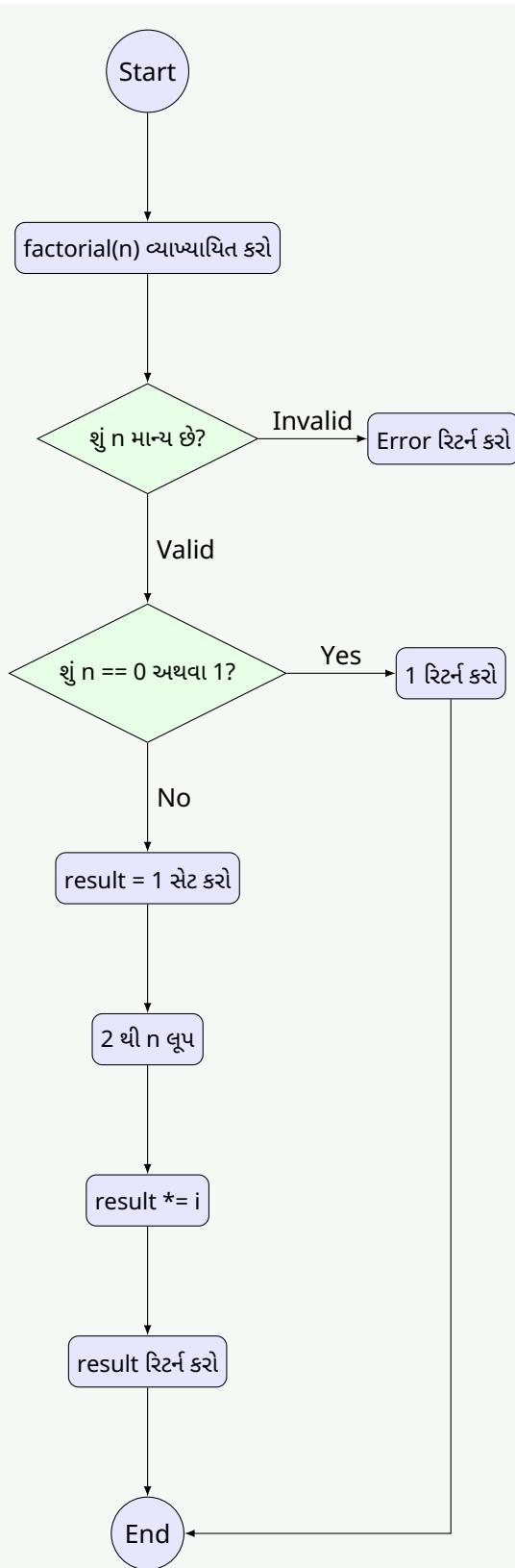
### જવાબ

Listing 12. Factorial Function

```

1 # આપેલા નંબરનો ફેક્ટરિયલ શોધવા માટેનું ફંક્શન
2 def factorial(n):
3     # ઇનપુટ માન્ય છે કે નહીં તે ચકાસો
4     if not isinstance(n, int) or n < 0:
5         return "અમાન્ય ઇનપુટ. કૃપા કરીને નોનનેગેટિવિટી ઇન્ટીજર દાખલ કરો."
6
7     # બેઝ કેસ: 0 અથવા 1 નો ફેક્ટરિયલ 1 છે
  
```

```
8     if n == 0 or n == 1:
9         return 1
10
11     # ઇટેશન વાપરીને ફેક્ટોરિયલ ગણતરી
12     result = 1
13     for i in range(2, n + 1):
14         result *= i
15
16     return result
17
18     # ફુલ્ફુલ ટેસ્ટ કરો
19     number = int(input("ફેક્ટોરિયલ શોધવા માટે એક નંબર દાખલ કરો: "))
20     print(f"{number} નો ફેક્ટોરિયલ {factorial(number)} છે!")
```



આકૃતિ 6. Factorial Function Flowchart

કોષ્ટક 8. ફેક્ટોરિયલ ઉદાહરણો

નંબર	ગણતરી	ફેક્ટોરિયલ
0	$0! = 1$	1
1	$1! = 1$	1
3	$3! = 3 \times 2 \times 1$	6
5	$5! = 5 \times 4 \times 3 \times 2 \times 1$	120

## મેમરી ટ્રીક

“1 સુધી ગુણાકાર કરો; બધા આંકડાને 1 સુધી ગુણાકાર કરો”

## પ્રશ્ન 3(a OR) [3 ગુણ]

1 થી N માંથી odd અને even નંબર શોધવાનો પાયથન કોડ બનાવો.

## જવાબ

Listing 13. Odd and Even Numbers Loop

```

1 # 1 થી N સુધીના odd અને even નંબર શોધવાનો પ્રોગ્રામ
2
3 # યુઝર પાસેથી ઇનપુટ લો
4 N = int(input("N ની કમિત દાખલ કરો: "))
5
6 print("1 થી", N, "સુધીના even નંબર છે:")
7 for i in range(1, N + 1):
8     if i % 2 == 0:
9         print(i, end=" ")
10
11 print("\n1 થી", N, "સુધીના odd નંબર છે:")
12 for i in range(1, N + 1):
13     if i % 2 != 0:
14         print(i, end=" ")

```

## કોષ્ટક 9. Even અને Odd ચેક

નંબર	ચેક	પ્રકાર
Even નંબર	number % 2 == 0	2, 4, 6, ...
Odd નંબર	number % 2 != 0	1, 3, 5, ...

## મેમરી ટ્રીક

“MOD-2: Modulo 2 જે even કે odd નક્કી કરે છે”

## પ્રશ્ન 3(b OR) [4 ગુણ]

Nested લિસ્ટ અને તેના એલિમેન્ટ ડિસ્પ્લે કરવા માટેનો પાયથન કોડ બનાવો.

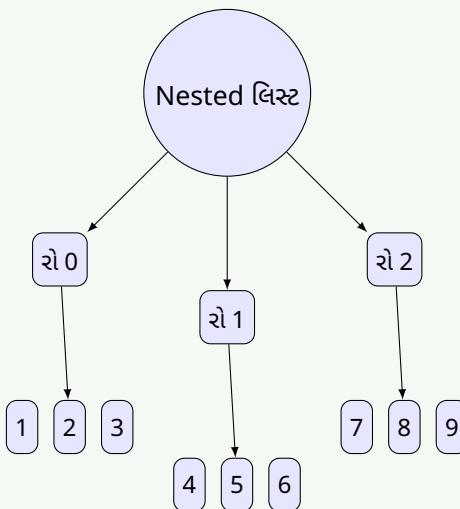
## જવાબ

**Listing 14.** Nested List Example

```

1 # Nested લસ્ટ બનાવવા અને ડસ્ટ્રિબ્યુટ કરવાનો પ્રોગ્રામ
2
3 # Nested લસ્ટ બનાવો
4 nested_list = [
5     [1, 2, 3],
6     [4, 5, 6],
7     [7, 8, 9]
8 ]
9
10 # Nested લસ્ટ ડસ્ટ્રિબ્યુટ કરો
11 print("Nested લસ્ટઃ", nested_list)
12
13 # Nested લૂપ્સનો ઉપયોગ કરીને દરેક એલમીન્ટ ડસ્ટ્રિબ્યુટ કરો
14 print("\nNested લસ્ટના એલમીન્ટ્સઃ")
15 for i in range(len(nested_list)):
16     for j in range(len(nested_list[i])):
17         print(f"nested_list[{i}][{j}] = {nested_list[i][j]}")
18
19 # enumerate નો ઉપયોગ કરીને વૈકલ્પિક રીત
20 print("\nenumerate નો ઉપયોગ કરીને:")
21 for i, inner_list in enumerate(nested_list):
22     for j, value in enumerate(inner_list):
23         print(f"પોઝિશિન ({i}, {j}): {value}")

```



**આકૃતિ 7.** Nested List Structure

## મેમરી ટ્રીક

“ROWS COLS: રો અને કોલમ માળખું બનાવે છે”

## પ્રશ્ન 3(c OR) [7 ગુણ]

Local અને Global વેરિયેબલ ઉદાહરણ સાથે સમજાવો.

## જવાબ

### કોષ્ટક 10. Local vs Global વેરિએબલ્સ

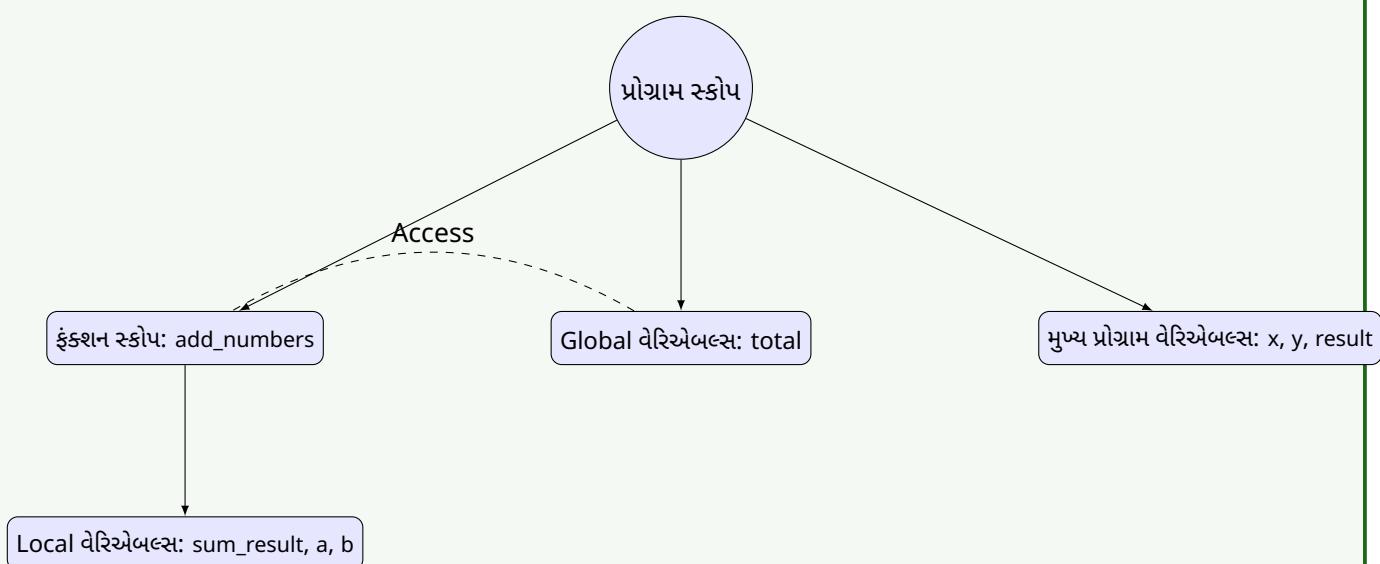
પ્રકાર	સ્કોપ	એક્સેસિબિલિટી	ઘોખણા
Local વેરિએબલ્સ	માત્ર જે ફંક્શનમાં ઘોખિત થયા છે ત્યાં	માત્ર ઘોખિત કરનાર ફંક્શનની અંદર	ફંક્શનની અંદર
Global વેરિએબલ્સ	સમગ્ર પ્રોગ્રામમાં	બધા ફંક્શન એક્સેસ કરી શકે	કોઈપણ ફંક્શનની બહાર

### Listing 15. Global vs Local Variables

```

1 # Global વેરિએબલ
2 total = 0
3
4 def add_numbers(a, b):
5     # Local વેરિએબલ્સ
6     sum_result = a + b
7     print(f"Local વેરિએબલ sum_result: {sum_result}")
8
9     # Global વેરિએબલ એક્સેસ કરવું
10    print(f"Global વેરિએબલ total મોડફિકેશન પહેલાં: {total}")
11
12    # ફંક્શનની અંદર Global વેરિએબલ મોડફિકેશન કરવા
13    global total
14    total = sum_result
15    print(f"Global વેરિએબલ total મોડફિકેશન પછી: {total}")
16
17    return sum_result
18
19 # મુખ્ય પ્રોગ્રામ
20 x = 5 # મુખ્ય પ્રોગ્રામમાં Local
21 y = 10 # મુખ્ય પ્રોગ્રામમાં Local
22
23 result = add_numbers(x, y)
24 print(f"પરિણામ: {result}")
25 print(f"અપડેટેડ global total: {total}")

```



### આકૃતિ 8. Variable Scoops

## મેમરી ટ્રીક

``GLOBAL SEES ALL: Global વેરિએબલ્સ બધે જોઈ શકે છે''

## પ્રશ્ન 4(a) [3 ગુણા]

પાયથન ની સ્ટાન્ડર્ડ લાઇબ્રેરી ના મેથેમેટિકલ ફંક્શન લિસ્ટ કરો.

## જવાબ

કોષ્ટક 11. પાયથોન Math મોડ્યુલ ફંક્શન્સ

ફંક્શન	વર્ણન	ઉદાહરણ
<b>abs()</b>	અબ્સોલ્યુટ વેલ્યુ આપે છે	<code>abs(-5) → 5</code>
<b>pow()</b>	$x$ ને $y$ ની ઘાત આપે છે	<code>pow(2, 3) → 8</code>
<b>max()</b>	સૌથી મોટી વેલ્યુ આપે છે	<code>max(5, 10, 15) → 15</code>
<b>min()</b>	સૌથી નાની વેલ્યુ આપે છે	<code>min(5, 10, 15) → 5</code>
<b>round()</b>	નજીકના પૂર્ણાંક સુધી રાઉન્ડ કરે છે	<code>round(4.6) → 5</code>
<b>math.sqrt()</b>	વર્ગમૂળ	<code>math.sqrt(16) → 4.0</code>
<b>math.sin()</b>	સાઇન ફંક્શન	<code>math.sin(math.pi/2) → 1.0</code>

## મેમરી ટ્રીક

``PEARS Math: Power, Exponents, Arithmetic, Roots, Sine functions in Math''

## પ્રશ્ન 4(b) [4 ગુણા]

પાયથન મોડ્યુલ કોડ સાથે સમજાવો.

## જવાબ

મોડ્યુલ: પાયથોનમાં મોડ્યુલ એટલે પાયથોન વ્યાખ્યાઓ અને સ્ટેટમેન્ટ્સ ધરાવતી ફાઇલ. ફાઇલનું નામ .py સફિક્સ સાથેનું મોડ્યુલનું નામ છે.

Listing 16. Module Usage Example

```

1 # math મોડ્યુલના ઉપયોગનું ઉદાહરણ
2 import math
3
4 # math મોડ્યુલમાંથી ગાણિતિક ફંક્શન્સનો ઉપયોગ
5 radius = 5
6 area = math.pi * math.pow(radius, 2)
7 print(f"ત્રાણિકા {radius} વાળા વર્તુળનું ક્ષેત્રફળ {area:.2f} છે")
8
9 # વિધિ import ટેકનાફિસનો ઉપયોગ
10 from math import sqrt, sin
11 angle = math.pi / 4
12 print(f"25 નું વર્ગમૂળ {sqrt(25)} છે")
13 print(f"{angle} રેન્ડિન્સનો સાઇન {sin(angle):.4f} છે")
14
15 # alias સાથે import
16 import random as rnd
17 random_number = rnd.randint(1, 100)
18 print(f"1 અને 100 વચ્ચેના રેન્ડમ નંબર: {random_number}")

```

### કોષ્ટક 12. મોડ્યુલ Import ટેકનિક્સ

પદ્ધતિ	સિન્ટેક્સ	ઉદાહરણ
આખો મોડ્યુલ import કરો	import module_name	import math
ચોક્કસ આઈટમ્સ import કરો	from module_name import item1, item2	from math import sqrt, sin
alias સાથે import કરો	import module_name as alias	import random as rnd

#### મેમરી ટ્રીક

“CODE-LIB: Code Libraries for reuse - ફરીથી ઉપયોગ માટે કોડ લાઇબ્રેરીઓ”

## પ્રશ્ન 4(c) [7 ગુણા]

એક પાયથન પ્રોગ્રામ લખો જે નિર્ધારિત કરે છે કે આપેલ નંબર 'આર્મસ્ટ્રોંગ નંબર' છે કે વપરાશકર્તા-વ્યાખ્યાયિત કાર્યનો ઉપયોગ કરીને પેલિન્ડ્રોમ છે.

#### જવાબ

**Listing 17. Armstrong and Palindrome Check**

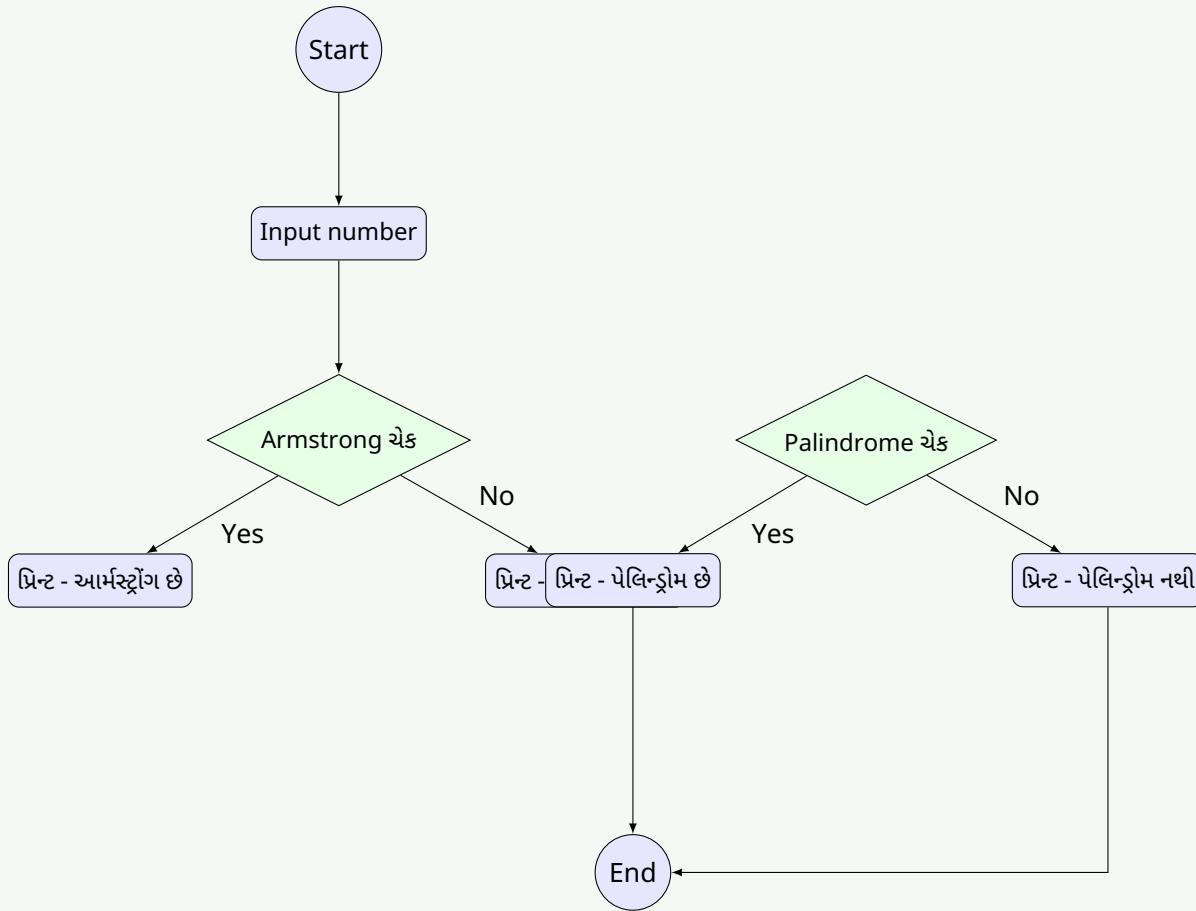
```

1 # નંબર આર્મસ્ટ્રોંગ છે કે નહી તે ચેક કરવા માટેનું ફૂક્શન
2 def is_armstrong(num):
3     # અંકોની સંખ્યા ગણવા માટે નંબરને સ્ટ્રિંગમાં રૂપાંતરિત કરો
4     num_str = str(num)
5     n = len(num_str)
6
7     # દરેક અંકની સંખ્યાની ઘાત સુધી ઉગામેલા સરવાળાની ગણતરી
8     armstrong_sum = 0
9     for digit in num_str:
10         armstrong_sum += int(digit) ** n
11
12     # ચેક કરો કે સરવાળો મૂળ નંબર સાથે મેળ ખાય છે
13     return armstrong_sum == num
14
15 # નંબર પેલનિડ્રોમ છે કે નહી તે ચેક કરવા માટેનું ફૂક્શન
16 def is_palindrome(num):
17     # નંબરને સ્ટ્રિંગમાં રૂપાંતરિત કરો અને ચેક કરો કે તે આગળથી અને પાછળથી એક સરખો વંચાય છે
18     num_str = str(num)
19     return num_str == num_str[::-1]
20
21 # મુખ્ય પ્રોગ્રામ
22 number = int(input("એક નંબર દાખલ કરો: "))
23
24 # ચેક કરો કે નંબર આર્મસ્ટ્રોંગ છે કે નહી
25 if is_armstrong(number):
26     print(f"{number} એક આર્મસ્ટ્રોંગ નંબર છે")
27 else:
28     print(f"{number} આર્મસ્ટ્રોંગ નંબર નથી")
29
30 # ચેક કરો કે નંબર પેલનિડ્રોમ છે કે નહી
31 if is_palindrome(number):
32     print(f"{number} એક પેલનિડ્રોમ છે")
33 else:
34     print(f"{number} પેલનિડ્રોમ નથી")

```

કોષ્ટક 13. ઉદાહરણો

નંબર	આર્મસ્ટ્રોગ ચેક	પેલિન્ડ્રોમ ચેક
153	$1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153 \checkmark$	$153 \neq 351 \times$
121	$1^3 + 2^3 + 1^3 = 1 + 8 + 1 = 10 \neq 121 \times$	$121 = 121 \checkmark$
1634	$1^4 + 6^4 + 3^4 + 4^4 = 1 + 1296 + 81 + 256 = 1634 \checkmark$	$1634 \neq 4361 \times$



આકૃતિ 9. Armstrong Palindrome Flowchart

## મેમરી ટ્રીક

“SAME SUM: SAME આગળ-પાછળ પેલિન્ડ્રોમ માટે, SUM ઘાતના અંકોનો આર્મસ્ટ્રોગ માટે”

## પ્રશ્ન 4(a OR) [3 ગુણ]

પાયથોનમાં બિલ્ટ ઇન ફંક્શન સમજાવો.

## જવાબ

**Built-in Functions:** આ ફંક્શન્સ પાયથોનના સ્ટાન્ડર્ડ લાઇબ્રેરીનો ભાગ છે અને કોઈપણ મોડ્યુલ import કર્યો વિના ઉપલબ્ધ છે.

કોષ્ટક 14. સામાન્ય પાયથોન Built-in Functions

ફંક્શન	હેતુ	ઉદાહરણ
<b>print()</b>	આઉટપુટ ડિસ્પ્લે	print("Hello")
<b>input()</b>	ચુઝર ઇનપુટ લે	name = input("Name: ")
<b>len()</b>	ઓફ્જેક્ટની લંબાઈ આપે	len([1, 2, 3]) → 3
<b>type()</b>	ઓફ્જેક્ટનો પ્રકાર આપે	type(5) → <class 'int'>
<b>int(), float(), str()</b>	ચોક્કસ પ્રકારમાં રૂપાંતર	int("5") → 5
<b>range()</b>	રીકવન્સ જનરેટ કરે	list(range(3)) → [0, 1, 2]
<b>sum()</b>	સરવાળો ગાળો	sum([1, 2, 3]) → 6

### મેમરી ટ્રીક

“PITS LCR: Print, Input, Type, Sum, Len, Convert, Range”

## પ્રશ્ન 4(b OR) [4 ગુણ]

એક પાયથોન કોડનું ઉદાહરણ આપીને પાયથોન મેથ મોડ્યુલનું વર્ણન કરો.

### જવાબ

પાયથોન Math મોડ્યુલ: math મોડ્યુલ C સ્ટાન્ડર્ડ દ્વારા વ્યાખ્યાયિત ગાણિતિક ફંક્શન્સની એક્સેસ પ્રદાન કરે છે.

Listing 18. Math Module Example

```

1 # math મોડ્યુલનો ઉપયોગ કરતો ઉદાહરણ
2 import math
3
4 # મૂળભૂત સ્થારિકો
5 print(f"pi ની કમિન્ટ: {math.pi}")
6 print(f"e ની કમિન્ટ: {math.e}")
7
8 # ત્રિક્લોશમતિફિલ્ફશન્સ આરગ્યુમેન્ટ (રેફિન્ન્સમાં)
9 angle = math.pi / 3 # 60 ડિગ્રી
10 print(f"{angle:.2f} રેફિન્ન્સનો સાઇન: {math.sin(angle):.4f}")
11 print(f"{angle:.2f} રેફિન્ન્સનો કોસાઇન: {math.cos(angle):.4f}")
12 print(f"{angle:.2f} રેફિન્ન્સનો ટેન્જન્ટ: {math.tan(angle):.4f}")
13
14 # લોગરધિમિક અને એક્સપોનેન્શાયિલ ફંક્શન્સ
15 x = 10
16 print(f"{x} નો નેચરલ લોગરધિમ: {math.log(x):.4f}")
17 print(f"{x} નો લોગરધિમ બેઝ 10: {math.log10(x):.4f}")
18 print(f"e ની {x} ઘાત: {math.exp(x):.4f}")
19
20 # અન્ય ફંક્શન્સ
21 print(f"25 નું વર્ગમૂળ: {math.sqrt(25)}")
22 print(f"4.3 નો રીલિંગ: {math.ceil(4.3)}")
23 print(f"4.7 નો ફ્લોર: {math.floor(4.7)})


```

કોષ્ટક 15. Math મોડ્યુલ કેટેગરીઝ

કેટેગરી	ફંક્શન્સ
સ્થારિકો	math.pi, math.e
નિકોશમિતિ	sin(), cos(), tan()
લોગરધિમિક	log(), log10(), exp()
ન્યુમેરિક	sqrt(), ceil(), floor()

## મેમરી ટ્રીક

"PENT: Pi/constants, Exponents, Numbers, Trigonometry"

## પ્રક્રિયા 4(c OR) [7 ગુણ]

પાયથોનમાં વેરિએબલના અવકાશનો કોન્સેપ્ટ સમજાવો અને પાયથોન પ્રોગ્રામમાં વૈશ્વિક અને સ્થાનિક વેરિએબલ કોન્સેપ્ટ લાગુ કરો.

## જવાબ

પાયથોનમાં વેરિએબલનો સ્કોપ: વેરિએબલનો સ્કોપ નક્કી કરે છે કે કે પ્રોગ્રામમાં ક્યાં વેરિએબલ એક્સેસિબલ કે દેખાય છે.

કોષ્ટક 16. વેરિએબલ સ્કોપના પ્રકારો

સ્કોપ	વર્ણન	એક્સેસ
Local	ફંક્શનની અંદર વ્યાખ્યાયિત વેરિએબલ્સ	માત્ર ફંક્શનની અંદર
Global	ટોપ લેવલ પર વ્યાખ્યાયિત વેરિએબલ્સ	સમગ્ર પ્રોગ્રામમાં
Enclosing	નેટ્વર્ક ફંક્શન-સના બાહ્ય ફંક્શનના વેરિએબલ્સ	બાહ્ય અને અંદરના ફંક્શનમાં
Built-in	પાયથોનમાં પહેલેથી વ્યાખ્યાયિત વેરિએબલ્સ	સમગ્ર પ્રોગ્રામમાં

Listing 19. Variable Scope Demonstration

```

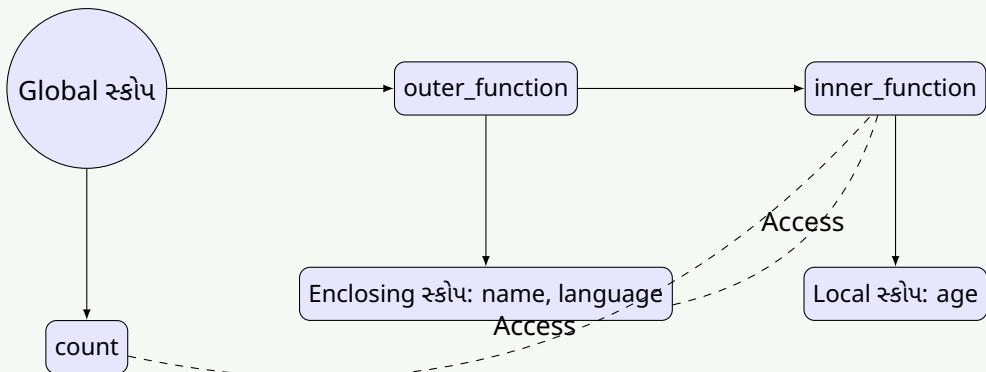
1 # વેરિએબલ સ્કોપ ડેમોન્સ્ટ્રેશન
2
3 # Global વેરિએબલ
4 count = 0
5
6 def outer_function():
7     # Enclosing સ્કોપ વેરિએબલ
8     name = "Python"
9
10    def inner_function():
11        # Local વેરિએબલ
12        age = 30
13        # Global વેરિએબલ એક્સેસ કરું
14        global count
15        count += 1
16        # Enclosing વેરિએબલ એક્સેસ કરું
17        print(f"inner_function ની અંદર: name is {name}")
18        print(f"inner_function ની અંદર: age is {age}")
19        print(f"inner_function ની અંદર: count is {count}")
20
21    # outer_function માટે Local વેરિએબલ
22    language = "Programming"
23    print(f"outer function ની અંદર: name is {name}")
24    print(f"outer function ની અંદર: language is {language}")
25    print(f"outer function ની અંદર: count is {count}")
26
27    # ઇનર ફંક્શન કોલ કરો
28    inner_function()
29
30    # આ ભૂલ આપરો - age એ inner_function માટે Local છે
31    # print(age)
32
33    # મુખ્ય પ્રોગ્રામ
34    print(f"Global સ્કોપ: count is {count}")
35    outer_function()
36    print(f"ફંક્શન કોલ પછી Global સ્કોપ: count is {count}")

```

```

37 # આ ભૂલ આપશે - તેઓ ફંક્શન્સ માટે Local છે
38 # print(name)
39 # print(language)
40

```



આકૃતિ 10. Variable Scope Hierarchy

## મેમરી ટ્રીક

“LEGB: Local, Enclosing, Global, Built-in - સ્કોપ લુકઅપનો ક્રમ”

## પ્રશ્ન 5(a) [3 ગુણ]

આપેલ સૂચિમાં બે ઘટકોને સ્વેપ કરવા માટે પાયથોન પ્રોગ્રામ બનાવો.

## જવાબ

Listing 20. Swapping List Elements

```

1 # લસ્ટમાં બે એલમિન્ટ્સ સ્વેપ કરવાનો પ્રોગ્રામ
2
3 # એક લસ્ટ બનાવો
4 my_list = [10, 20, 30, 40, 50]
5 print("મૂળ લસ્ટ:", my_list)
6
7 # સ્વેપ કરવા માટેની પોઝિશન મેળવો
8 pos1 = int(input("પ્રથમ પોઝિશન દાખલ કરો ઇન્ડેક્સ( 0 થી શરૂ થાય છે): "))
9 pos2 = int(input("બીજી પોઝિશન દાખલ કરો ઇન્ડેક્સ( 0 થી શરૂ થાય છે): "))
10
11 # ટેમ્પરરી વેરાયેબલનો ઉપયોગ કરીને એલમિન્ટ્સ સ્વેપ કરો
12 if 0 <= pos1 < len(my_list) and 0 <= pos2 < len(my_list):
13     # સ્વેપગ્યા
14     temp = my_list[pos1]
15     my_list[pos1] = my_list[pos2]
16     my_list[pos2] = temp
17
18     print(f"પોઝિશન {pos1} અને {pos2} પર એલમિન્ટ્સ સ્વેપ કર્યા પછી લસ્ટ:", my_list)
19 else:
20     print("અમન્ય પોઝિશન! પોઝિશન લસ્ટની રેન્જની અંદર હોવી જોઈએ.")

```

## વૈકલ્પિક પદ્ધતિ:

### Listing 21. Pythonic Swap

```

1 # પાયથોનની tuple અનપેકિનો ઉપયોગ કરીને સ્વેપ વધુ(પાયથોનકિ)
2 if 0 <= pos1 < len(my_list) and 0 <= pos2 < len(my_list):
3     my_list[pos1], my_list[pos2] = my_list[pos2], my_list[pos1]
4     print(f"પોઝિશન {pos1} અને {pos2} પર એલમિન્ટ્સ સ્વેપ કર્યા પછી લિસ્ટ: ", my_list)

```

### કોષ્ટક 17. સ્વેપિંગ પદ્ધતિઓ

પદ્ધતિ	કોડ
ટૈમ્પ વેરિએબલનો ઉપયોગ	temp = a; a = b; b = temp
પાયથોન ટપલ અનપેકિંગ	a, b = b, a

### મેમરી ટ્રીક

“TEMP SWAP: Temporary variable helps safe swapping”

## પ્રશ્ન 5(b) [4 ગુણ]

ઉદાહરણ આપીને નેસ્ટેડ લિસ્ટ સમજાવો.

### જવાબ

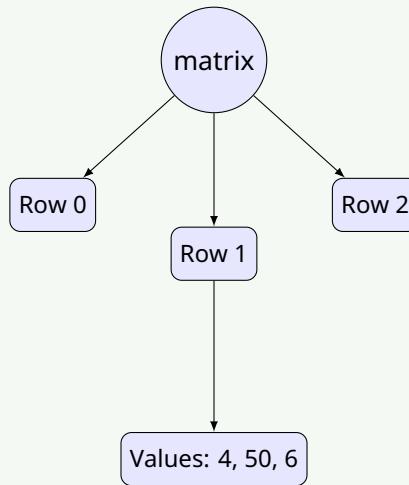
**Nested List:** નેસ્ટેડ લિસ્ટ એ એક લિસ્ટ છે જે તેના ઘટકો તરીકે અન્ય લિસ્ટ્સ ધરાવે છે, જે મલ્ટી-ડાયમેન્શનલ ડેટા સ્ટ્રક્ચર બનાવે છે.

### Listing 22. Nested List Operations

```

1 # નેસ્ટેડ લાસ્ટ બનાવવું (3x3 મેટ્રિક્સ)
2 matrix = [
3     [1, 2, 3],
4     [4, 5, 6],
5     [7, 8, 9]
6 ]
7
8 # એલમિન્ટ્સ એક્સેસ કરવા
9 print("સંપૂર્ણ મેટ્રિક્સ:", matrix)
10 print("પ્રથમ રૌન્ડ:", matrix[0])
11 print("રો 1, કોલમ 2 પર એલમિન્ટ:", matrix[0][1]) # આર્ટિપુર: 2
12
13 # એલમિન્ટ્સ મોડફિય કરવા
14 matrix[1][1] = 50
15 print("મોડફિકેશન પછી મેટ્રિક્સ:", matrix)
16
17 # નેસ્ટેડ લાસ્ટ માં લૂપ ફેરવવું
18 print("મેટ્રિક્સની પ્રાન્તી કરવું:")
19 for row in matrix:
20     for element in row:
21         print(element, end=" ")
22     print() # દરેક રો પછી નવી લાઇન

```



આકૃતિ 11. Nested List Structure

કોષ્ટક 18. Nested List Operations

ઓપરેશન	સિન્ક્રેસ	ઉદાહરણ
એલિમેન્ટ એક્સેસ	list[row][col]	matrix[0][1]
એલિમેન્ટ મોડિફિકેશન	list[row][col] = new_value	matrix[1][1] = 50
નવી રો ઉમેરવી	list.append([...])	matrix.append([10, 11, 12])

### મેમરી ટ્રીક

“MARS: Matrix Access with Row and column Structure - મેટ્રિક્સ એક્સેસ રો અને કોલમ માળખા સાથે”

## પ્રશ્ન 5(c) [7 ગુણા]

ઉદાહરણો સાથે સ્ટ્રિંગ ઓપરેશન્સ સમજાવો.

### જવાબ

કોષ્ટક 19. પાયથોનમાં સ્ટ્રિંગ ઓપરેશન્સ

ઓપરેશન	વર્ણન	ઉદાહરણ
Concatenation	સ્ટ્રિંગ્સ જોડવા	"Hello" + " World" → "Hello World"
Repetition	સ્ટ્રિંગ્સ રિપીટ કરવા	"Python" * 3 → "PythonPythonPython"
Slicing	સબસ્ટ્રિંગ એક્સ્ટ્રેક્ટ કરવું	"Python"[1:4] → "yth"
Indexing	કેરેક્ટર એક્સેસ કરવું	"Python"[0] → "P"
Length	કેરેક્ટર્સ ગણવા	len("Python") → 6
Membership	હાજરી ચેક કરવી	"P" in "Python" → True
Comparison	સ્ટ્રિંગ્સ સરખાવવા	"apple" < "banana" → True

Listing 23. String Operations Example

```

1 # સૂટરંગિ ઓપરેશન્સ ડેમોનસૂટરેશન
2 text = "Python Programming"
3
4 # Indexing
5 print("રૂ પ્રથમ કેરેક્ટર:", text[0])
  
```

```

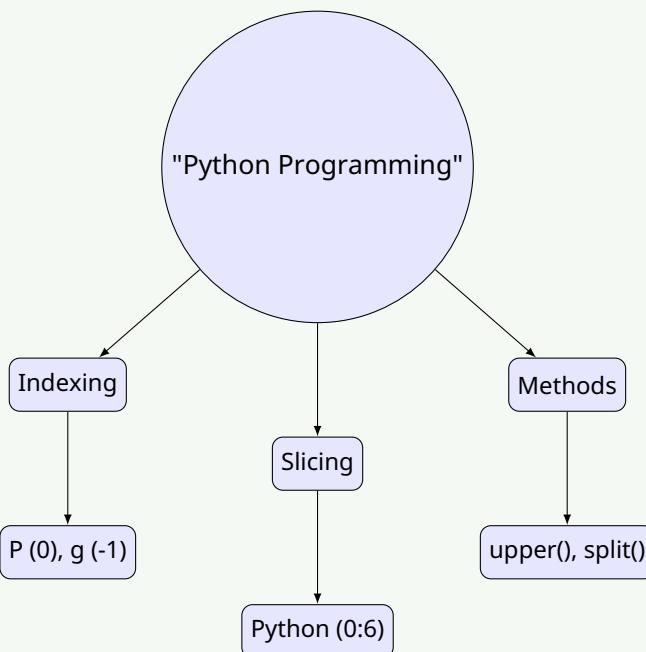
6 print("છેલ્લો કેરેક્ટરાં:", text[-1])

7
8 # Slicing
9 print("પૂર્વમ શબ્દાં:", text[:6])
10 print("બીજો શબ્દાં:", text[7:])
11 print("મધ્યા કેરેક્ટરસાં:", text[3:10])
12 print("ઉલ્ટું (Reverse):", text[::-1])

13
14 # સ્ટ્રિંગ મેથ્ડ્સ
15 print("અપરકેરાં:", text.upper())
16 print("લોઓકેસરાં:", text.lower())
17 print("‘P’ ને ‘J’ સાથે બદલોં:", text.replace("P", "J"))
18 print("સ્પેસ દ્વારા સ્પ્લાઇટાં:", text.split())
19 print("‘m’ ગણોં:", text.count('m'))
20 print("‘gram’ શોધાં:", text.find("gram"))

21
22 # રેક ઓપરેશન્સ
23 print("શું આલફા-નિયુમેરિક છે?", text.isalnum())
24 print("શું ‘Py’ થી શરૂ થાય છે?", text.startswith("Py"))
25 print("શું ‘ing’ થી સમાપ્ત થાય છે?", text.endswith("ing"))

```



આકૃતિ 12. String Operations

## મેમરી ટ્રીક

“SCREAM: Slice, Concat, Replace, Extract, Access, Methods”

## પ્રશ્ન 5(a OR) [3 ગુણ]

આપેલ સૂચિમાં તમામ ઘટકોનો સરવાળો શોધવા માટે પાયથોન પ્રોગ્રામ બનાવો

## જવાબ

**Listing 24.** Sum of List Elements

```

1 # લિસ્ટમાં રહેલા તમામ એલમીન્ટ્સનો સરવાળો શોધવાનો પ્રોગ્રામ
2
3 # પદ્ધતિ1: બલિટિન- sum() ફંક્શનનો ઉપયોગ કરીને
4 def sum_list_builtin(numbers):
5     return sum(numbers)
6
7 # પદ્ધતિ2: લૂપનો ઉપયોગ કરીને
8 def sum_list_loop(numbers):
9     total = 0
10    for num in numbers:
11        total += num
12    return total
13
14 # એક સેમ્પલ લિસ્ટ બનાવો
15 my_list = [10, 20, 30, 40, 50]
16 print("લિસ્ટ:", my_list)
17
18 # બલિટિન- ફંક્શનનો ઉપયોગ કરીને સરવાળો ગણો
19 print("બલિટિન- ફંક્શનનો ઉપયોગ કરીને સરવાળો:", sum_list_builtin(my_list))
20
21 # લૂપનો ઉપયોગ કરીને સરવાળો ગણો
22 print("લૂપનો ઉપયોગ કરીને સરવાળો:", sum_list_loop(my_list))

```

**કોષ્ટક 20.** સરવાળા પદ્ધતિઓની તુલના

પદ્ધતિ	ફાયદો
Built-in <code>sum()</code>	સરળ, કાર્યક્ષમ, ઝડપી
Loop approach	કસ્ટમ સરવાળા લોજિક માટે કામ કરે છે

## મેમરી ટ્રીક

“ADD ALL: દરેક એલિમેન્ટને ઉમેરો”

## પ્રશ્ન 5(b OR) [4 ગુણ]

પાયથોન લિસ્ટમાં ઈન્ડેક્સિંગ અને સ્લાઇસિંગ ઓપરેશન્સ સમજાવો

## જવાબ

**કોષ્ટક 21.** ઈન્ડેક્સિંગ અને સ્લાઇસિંગ ઓપરેશન્સ

ઓપરેશન	સિન્ટેક્સ	વર્ણન	ઉદાહરણ
Positive Indexing	<code>list[i]</code>	ઠાં પોઝિશન પર આઈટમ એક્સેસ કરો	<code>fruits[0]</code>
Negative Indexing	<code>list[-i]</code>	છેલ્લેથી આઈટમ એક્સેસ કરો	<code>fruits[-1]</code>
Basic Slicing	<code>list[s:e]</code>	start થી end-1 સુધીની આઈટમ્સ	<code>fruits[1:3]</code>
Slice with Step	<code>list[s:e:st]</code>	રટેપના અંતરાલ સાથે આઈટમ્સ	<code>nums[1:6:2]</code>
Reverse	<code>list[::-1]</code>	લિસ્ટને ઉલ્ટું કરો	<code>fruits[::-1]</code>

**Listing 25.** Indexing and Slicing Demo

```

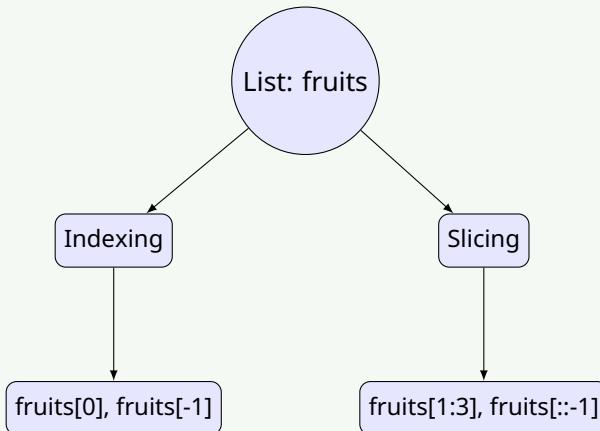
1 # ઈન્ડેક્સિંગ અને સ્લાઇસિંગ ડેમોન્સ્ટ્રેશન

```

```

2 fruits = ["apple", "banana", "cherry", "date", "elderberry", "fig"]
3 print("મૂળ લાસ્ટ:", fruits)
4
5 # Indexing
6 print("ઇન્ડેક્સિંગ ઉદાહરણો:")
7 print("પ્રથમ આઈટમ:", fruits[0]) # apple
8 print("છેલ્લી આઈટમ:", fruits[-1]) # fig
9
10 # Slicing
11 print("સ્લાઇસિંગ ઉદાહરણો:")
12 print("પ્રથમ ત્રણા આઈટમ્સ:", fruits[:3])
13 print("છેલ્લી ત્રણા આઈટમ્સ:", fruits[-3:])
14 print("મધ્ય આઈટમ્સ:", fruits[2:4])
15 print("દ્વારા બીજી આઈટમ્સ:", fruits[::2])
16 print("ઉલ્લંઘન લાસ્ટ:", fruits[::-1])

```



આકૃતિ 13. Indexing and Slicing

**મેમરી ટ્રીક**

``START-END-STEP: સ્લાઇસિંગ સિન્ટેક્સ: [start:end:step]''

**પ્રક્રિયા 5(c OR) [7 ગુણ]**

જરૂરી ઉદાહરણ સાથે tuple ને ટૂંકમાં સમજાવો.

**જવાબ**

Tuple: Tuple એ એલિમેન્ટ્સનું ઓર્ડર્ડ, અપરિવર્તનીય (immutable) કલેક્શન છે. એકવાર બનાવ્યા પછી, એલિમેન્ટ્સ બદલી શકતા નથી.

કોષ્ટક 22. Tuple vs List

ક્રીયા	Tuple	List
સિન્ટેક્સ	(item1, item2)	[item1, item2]
ભ્યુટેબિલિટી	Immutable (બદલી શકતું નથી)	Mutable (બદલી શકાય છે)
પરફોર્મન્સ	ઝડપી	ધીમું
ઉપયોગ	ફિક્સ્ડ ડેટા, કીજ	ડાયનેમિક ડેટા

Listing 26. Tuple Example

```

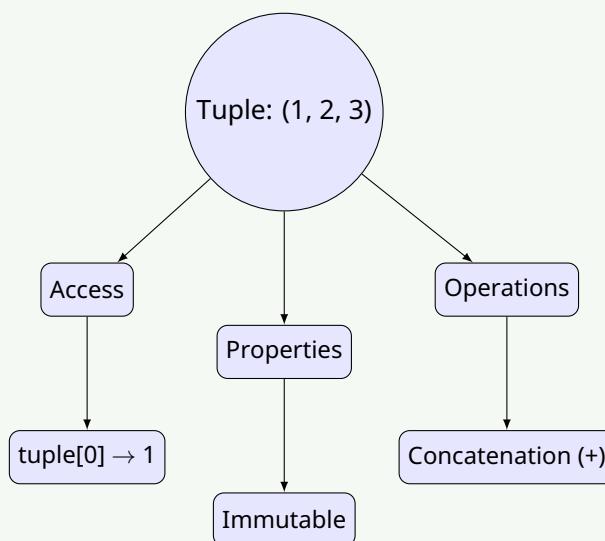
1 # Tuples બનાવવા
2 empty_tuple = ()
3 single_item_tuple = (1,) # એક આઈટમ માટે અલ્પવરિએમ જરૂરી છે
4 mixed_tuple = (1, "Hello", 3.14, True)
5 nested_tuple = (1, 2, (3, 4), 5)

6
7 # Tuple એલમેન્ટ્સ એક્સેસ કરવા
8 print("પ્રથમ આઈટમ:", mixed_tuple[0]) # 1
9 print("નેસ્ટેડ ટ્પલ એલમેન્ટ:", nested_tuple[2][0]) # 3

10
11 # Tuple ઓપરેશન્સ
12 combined_tuple = mixed_tuple + nested_tuple
13 print("સંયુક્ત ટ્પલ:", combined_tuple)

14
15 # આ નૂલ આપશે કારણ કે tuples અપરાઇટનીય છે
16 # mixed_tuple[0] = 100 # TypeError

```



આકૃતિ 14. Tuple Concepts

## મેમરી ટ્રીક

“IPAC: Immutable, Parentheses, Access only, Cannot modify - અપરિવર્તનીય, કૌંસ, માત્ર એક્સેસ, મોડિફાય ન કરી શકાય”