પાયથોન પ્રોગ્રામિંગ (1323203) - શિયાળુ 2023 સોલ્યુશન

Milav Dabgar

20 જાન્યુઆરી, 2024

પ્રશ્ન 1(a) [3 ગુણ]

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

જવાબ

Listing 1. Pseudocode for Number Check

```
BEGIN
   Input number
   IF number > 0 THEN
      Display "Number is positive"
   ELSE IF number < 0 THEN
      Display "Number is negative"
   ELSE
      Display "Number is zero"
   END IF
END
```

મેમરી ટ્રીક

``શૂન્ય સાથે સરખાવો''

પ્રશ્ન 1(b) [4 ગુણ]

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

જવાબ

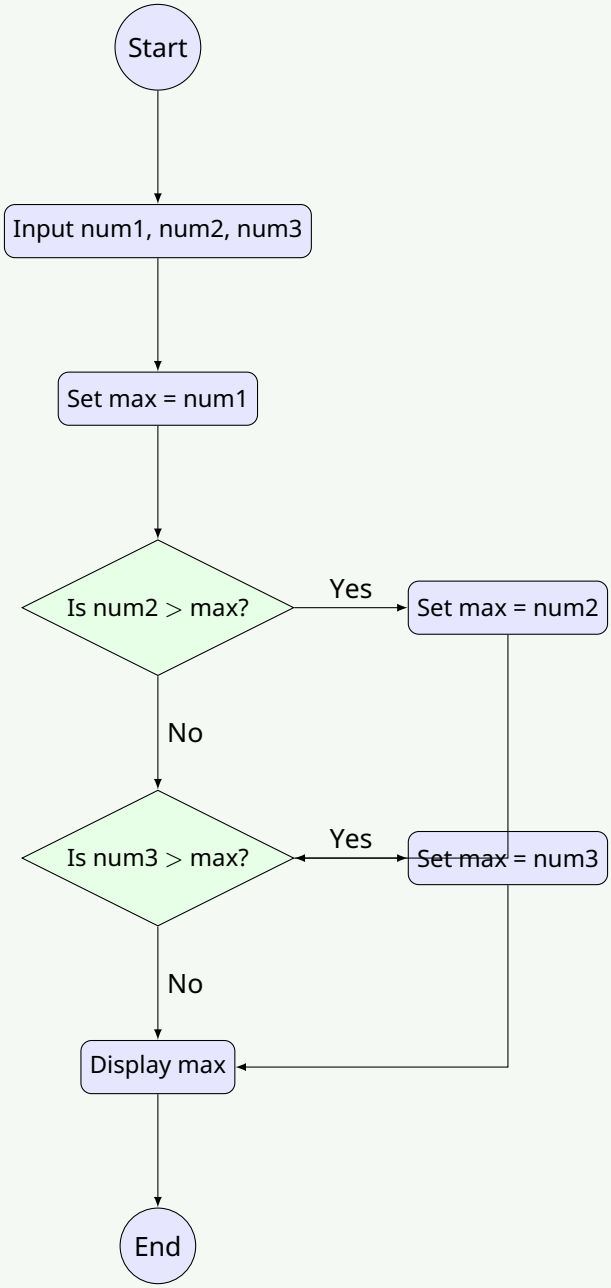
Algorithm વ્યાખ્યા: એલ્ગોરિધમ એટલે ચોક્કસ સમસ્યાને ઉકેલવા માટે અથવા ગણતરી કરવા માટે બનાવેલ સ્ટેપ-બાય-સ્ટેપ પ્રક્રિયા અથવા નિયમોનો સેટ.

ત્રણ નંબરમાંથી મહત્તમ શોધવાનો એલ્ગોરિધમ:

Listing 2. Algorithm for Maximum of Three Numbers

```
BEGIN
   Input num1, num2, num3
   Set max = num1
   IF num2 > max THEN
      Set max = num2
   END IF
   IF num3 > max THEN
      Set max = num3
```

```
    END IF
    Display max
END
```



આકૃતિ 1. Flowchart to Find Maximum of Three Numbers

મેમરી ટ્રીક

``સરખામણી અને બદલો''

પ્રશ્ન 1(c) [7 ગુણ]

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

જવાબ

Listing 3. Celsius to Fahrenheit Conversion

```python
# સેલ્સિયસથી ફેરનહાઇટ રૂપાંતરનો પ્રોગ્રામ

# યુઝર પાસેથી સેલ્સિયસ તાપમાન મેળવો
celsius = float(input("સેલ્સિયસમાં તાપમાન દાખલ કરો: "))

# સૂત્ર વાપરીને ફેરનહાઇટમાં રૂપાંતરિત કરો: F = (C * 9/5) + 32
fahrenheit = (celsius * 9/5) + 32

# પરિણામ દર્શાવો
print(f"{celsius}\u00B0C એ {fahrenheit}\u00B0F ની બરાબર છે")
```

કોષ્ટક 1. તાપમાન રૂપાંતરણ

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

મેમરી ટ્રીક

``9થી ગુણાકાર, 5થી ભાગાકાર, 32 ઉમેરો''

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
>=	કરતાં મોટું અથવા બરાબર	5 >= 5	True
<=	કરતાં નાનું અથવા બરાબર	5 <= 5	True

Code Example:

Listing 4. Comparison Operators Example

```python
# પાયથોન કંપેરઝિન ઓપરેટર્સ ઉદાહરણ
a = 10
b = 5

# બરાબર છે
print(f"{a} == {b}: {a == b}")  # False

# બરાબર નથી
print(f"{a} != {b}: {a != b}")  # True
```

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

મેમરી ટ્રીક

``સરખાવો''

પ્રશ્ન 2(a) [3 ગુણ]

પાયથોન ના ડેટા ટાઇપ સમજાવો.

જવાબ

કોષ્ટક 3. પાયથોન ડેટા ટાઇપ્સ

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

મેમરી ટ્રીક

``NIFTY SLD: નંબર્સ, ઇન્ટીજર્સ, ફ્લોટ્સ, ટેક્સ્ટ, યસ/નો, સીક્વન્સીસ, લિસ્ટ્સ, ડિક્શનરીઝ''

પ્રશ્ન 2(b) [4 ગુણ]

Nested If પાયથોન કોડ ના ઉદાહરણ સાથે સમજાવો.

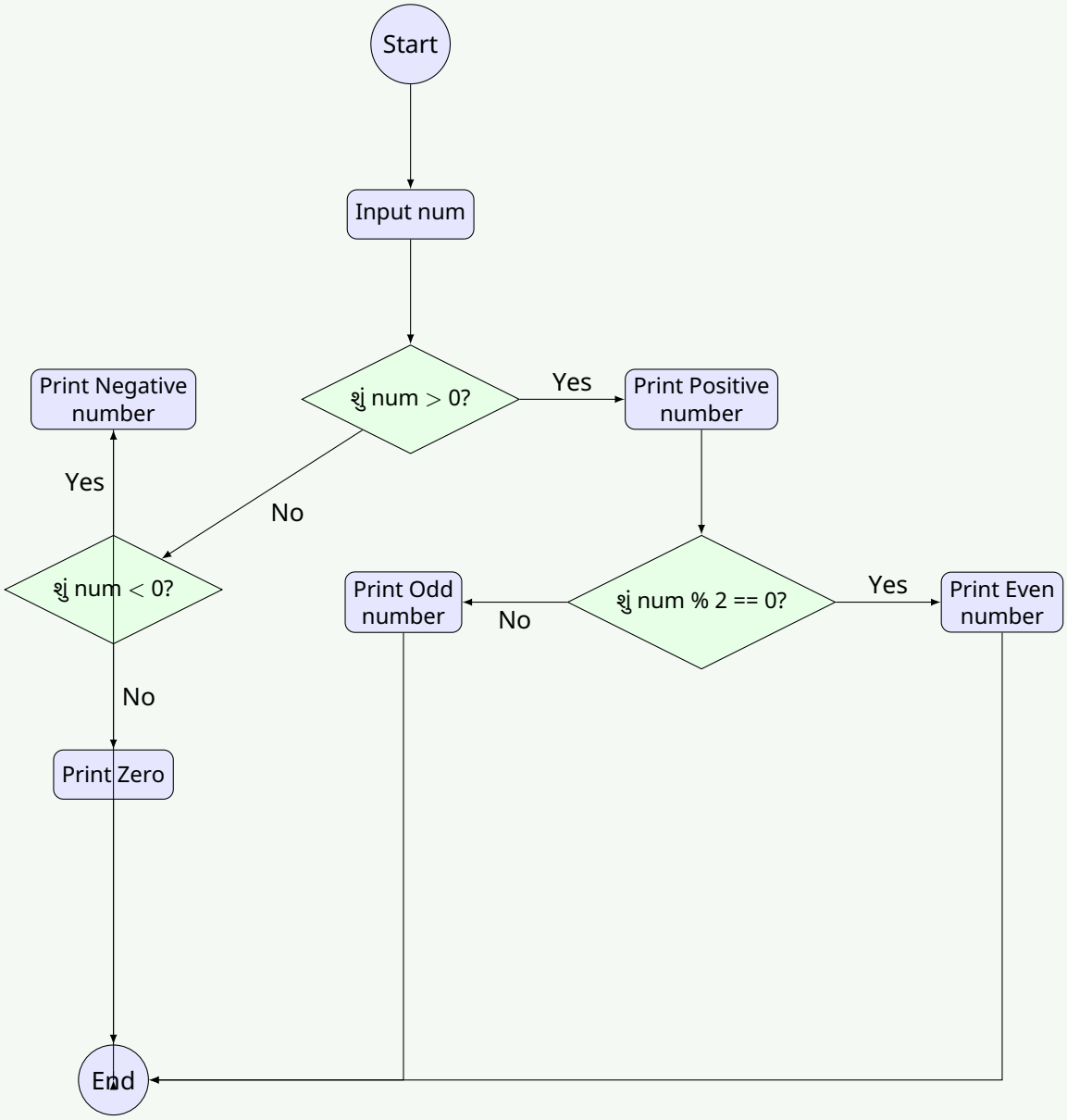
જવાબ

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 પોઝિટિવ, નેગેટિવ કે શૂન્ય છે તે ચકાસવા માટેનો nested if ઉદાહરણ
2  # અને જો પોઝિટિવ હોય, તો તે સમ છે કે વિષમ તે ચકાસો
```

```python

num = int(input("એક નંબર દાખલ કરો: "))

if num > 0:
    print("પોઝિટિવ નંબર")
    # nested if જે ચકાસે છે કે પોઝિટિવ નંબર સમ છે કે વિષમ
    if num % 2 == 0:
        print("સમ નંબર")
    else:
        print("વિષમ નંબર")
elif num < 0:
    print("નેગેટિવ નંબર")
else:
    print("શૂન્ય")
```



આકૃતિ 2. Nested If Flowchart

મેમરી ટ્રીક

``ચેક અંદર ચેક''

પ્રશ્ન 2(c) [7 ગુણ]

ઉદાહરણ સાથે વિવિધ પ્રકારના પસંદગી/નિર્ણય લેવાના ફ્લો-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 નિર્ણય લેવા
if-elif-else	ઘણી શરતો ચકાસવી	ઘણા સંભવિત પરિણામો
Nested if	શરત અંદર બીજી શરત	જટિલ શ્રેણીબદ્ધ નિર્ણયો
Ternary operator	એક લાઇન if-else	સરળ શરતી નિયુક્તિ

Code Example:

Listing 6. Selection Structures Example

```
# વિવિધ સિલેક્શન સ્ટ્રક્ચર્સનું ઉદાહરણ
score = int(input("તમારો સ્કોર દાખલ કરો: "))

# સાદું if
if score >= 90:
    print("ઉત્તમ!")

# if-else
if score >= 60:
    print("તમે પાસ થયા છો.")
else:
    print("તમે નાપાસ થયા છો.")

# if-elif-else
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "F"
print(f"તમારો ગ્રેડ {grade} છે")

# Ternary operator
result = "પાસ" if score >= 60 else "નાપાસ"
print(result)
```

મેમરી ટ્રીક

``SCENE: સિમ્પલ if, કન્ડિશન્સ વિથ else, Elif ફોર મલ્ટિપલ, Nested ફોર કોમ્પ્લેક્સ, એક્સપ્રેસ વિથ ટર્નરી''

પ્રશ્ન 2(a OR) [3 ગુણ]

વેરિયેબલ વ્યાખ્યાયિત કરવાના નિયમો લિસ્ટ કરો.

જવાબ

કોષ્ટક 5. પાયથોનમાં વેરિએબલ્સ વ્યાખ્યાયિત 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 સેન્સિટિવ	મોટા અક્ષરો અને નાના અક્ષરો અલગ	age અને Age અલગ વેરિએબલ્સ છે
રિઝર્વ્ડ 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 યોગ્ય રીતે, ઇગ્નોર સ્પેશિયલ કેરેક્ટર, લૂક એટ કેસ, કીવર્ડ્સ અવોઇડ, સ્પેસ નોટ અલાઉડ''

પ્રશ્ન 2(b OR) [4 ગુણ]

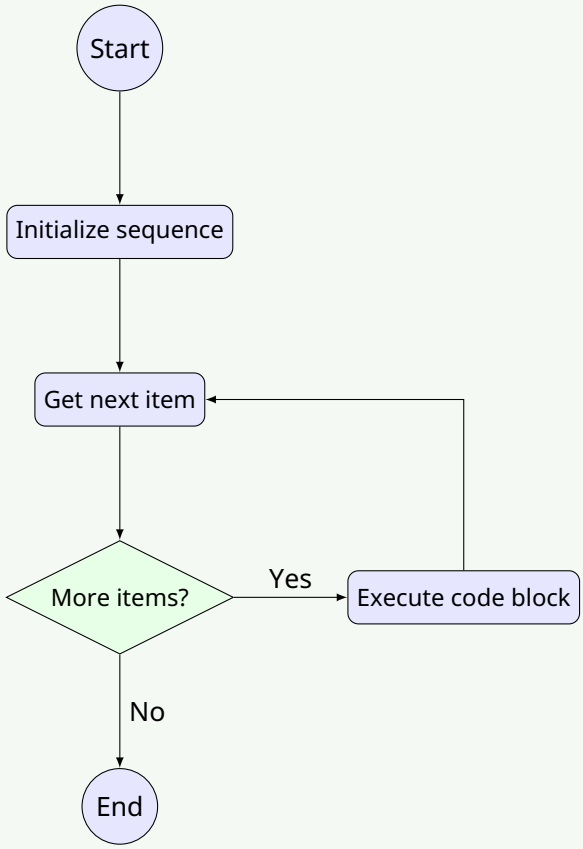
ફોર લૂપ ને જરૂરી ઉદાહરણ સાથે સમજાવો.

જવાબ

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

Listing 7. For Loop Example

```
# પાયથોનમાં for લૂપનો ઉદાહરણ
# લિસ્ટના દરેક એલિમેન્ટને પ્રિન્ટ કરવા
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)

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

# સ્ટ્રિંગ સાથે for લૂપનો ઉપયોગ
name = "Python"
for char in name:
    print(char)
```



આકૃતિ 3. For Loop Flowchart

મેમરી ટ્રીક

``ITEM: દરેક સભ્ય પર પુનરાવર્તન કરો''

પ્રશ્ન 2(c OR) [7 ગુણ]

Break અને continue સ્ટેટમેન્ટને સંક્ષિપ્તમાં સમજાવો.

જવાબ

કોષ્ટક 6. Break અને Continue સ્ટેટમેન્ટ્સ

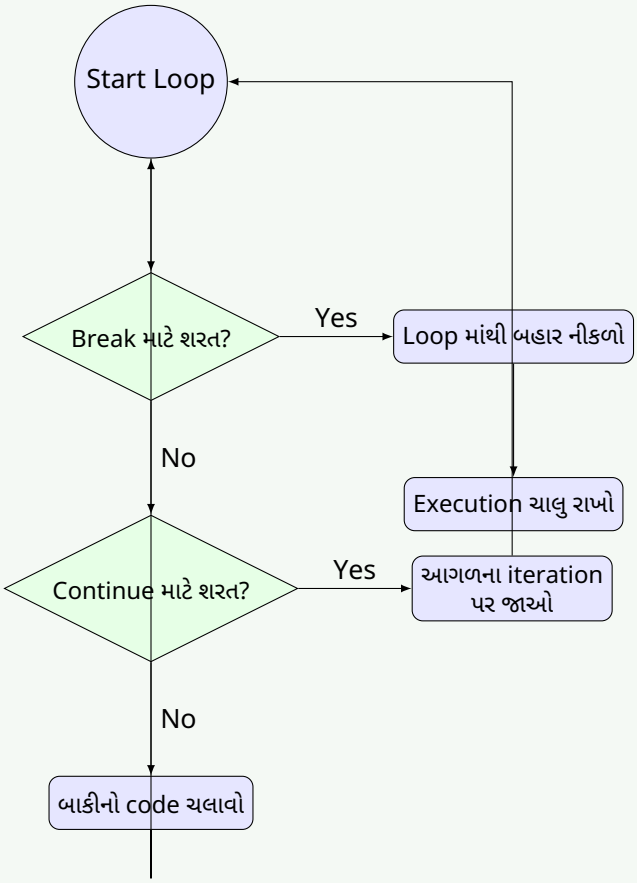
ટેટમેન્ટ	હેતુ	અસર
break	લૂપમાંથી તરત જ બહાર નીકળો	વર્તમાન લૂપને અટકાવે છે અને લૂપ પછીના સ્ટેટમેન્ટ પર કંટ્રોલ ટ્રાન્સફર કરે છે
continue	વર્તમાન પુનરાવર્તન છોડી દો	લૂપના આગલા પુનરાવર્તન પર જાય છે, continue સ્ટેટમેન્ટ પછીના કોઈપણ કોડને છોડી દે છે

Code Example:

Listing 8. Break and Continue Example

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

```
        print("i =", i, "પર લૂપ તોડીએ છીએ")
        break
    print(i, end=" ")
print("\nલૂપ સમાપ્ત થઈ")

# Continue સ્ટેટમેન્ટ ઉદાહરણ
print("\nContinue ઉદાહરણ:")
for i in range(1, 11):
    if i % 2 == 0:
        continue
    print(i, end=" ")
print("\nમાત્ર વિષમ નંબરો પ્રિન્ટ 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 થી 10 સુધીના નંબર પ્રિન્ટ કરવા for લૂપનો ઉપયોગ
print("for લૂપનો ઉપયોગ કરીને:")
for i in range(1, 11):
    print(i, end=" ")

print("\n\nwhile લૂપનો ઉપયોગ કરીને:")
# 1 થી 10 સુધીના નંબર પ્રિન્ટ કરવા while લૂપનો ઉપયોગ
counter = 1
while counter <= 10:
    print(counter, end=" ")
    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

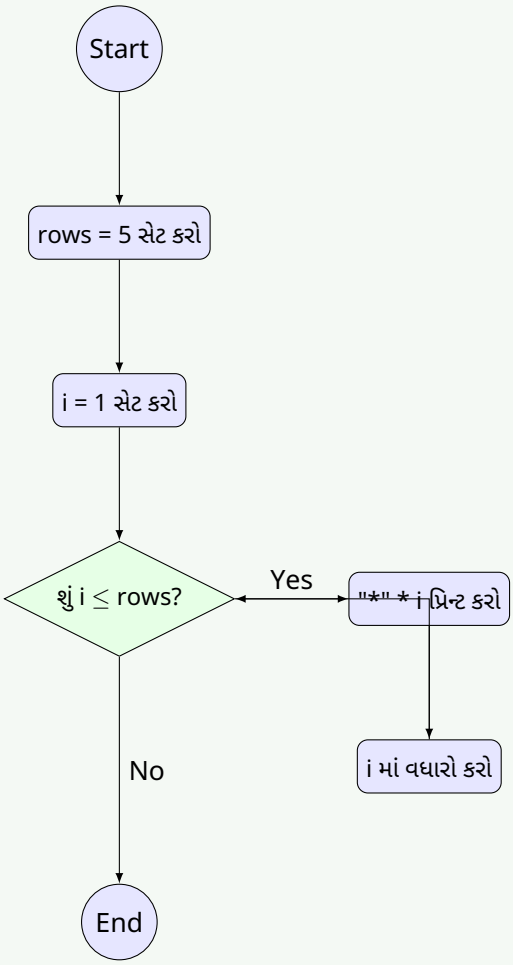
```
# for લૂપનો ઉપયોગ કરીને સ્ટાર પેટર્ન પ્રિન્ટ કરો
rows = 5

for i in range(1, rows + 1):
    # દરેક રો માં i જેટલા સ્ટાર પ્રિન્ટ કરો
    print("*" * i)
```

વૈકલ્પિક ઉકેલ નેસ્ટેડ લૂપ્સ સાથે:

Listing 11. Star Pattern Nested Loop

```
# નેસ્ટેડ લૂપ્સનો ઉપયોગ કરીને સ્ટાર પેટર્ન પ્રિન્ટ કરો
rows = 5

for i in range(1, rows + 1):
    for j in range(1, i + 1):
        print("*", end="")
```

```
    print()  # દરેક રો પછી ન્યુ લાઇન
```



આકૃતિ 5. Pattern Printing Flowchart

મેમરી ટ્રીક

``RISE UP: રો વધે છે, સ્ટાર ઉપર તરફ વિસ્તરે છે''

પ્રશ્ન 3(c) [7 ગુણ]

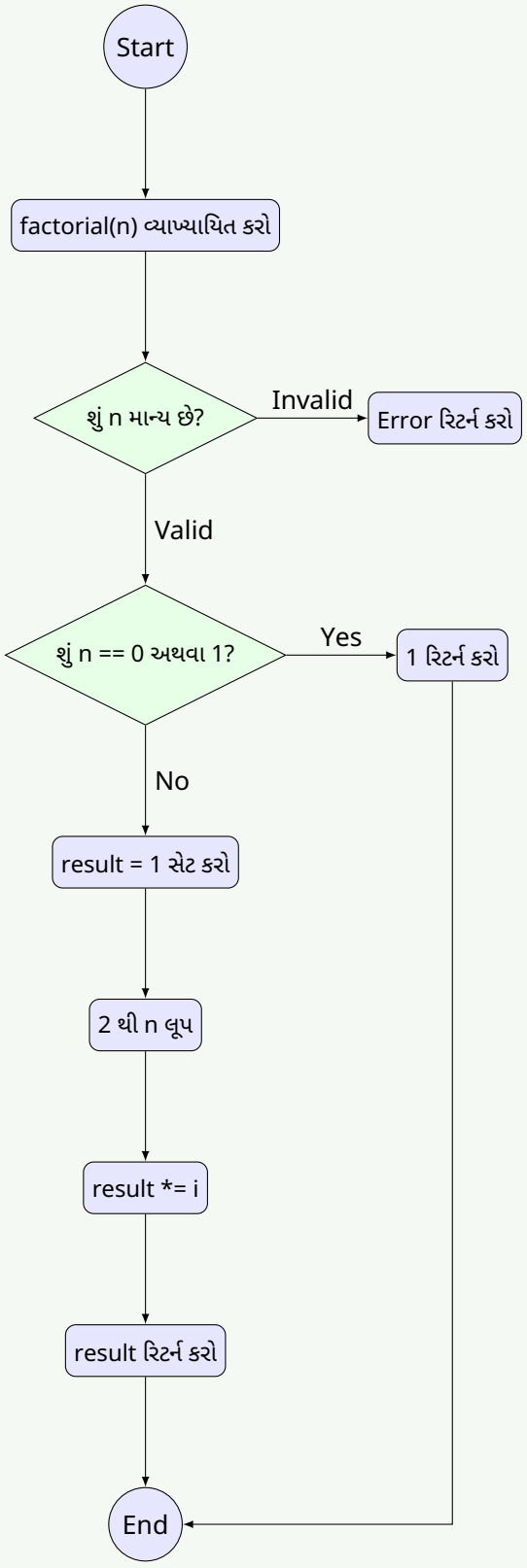
આપેલા નંબર નો factorial શોધવા માટેનું યુઝર ડિફાઇન ફંક્શન બનાવો.

જવાબ

Listing 12. Factorial Function

```
# આપેલા નંબરનો ફેક્ટોરિયલ શોધવા માટેનું ફંક્શન
def factorial(n):
    # ઇનપુટ માન્ય છે કે નહીં તે ચકાસો
    if not isinstance(n, int) or n < 0:
        return "અમાન્ય ઇનપુટ. કૃપા કરીને નોનનેગેટિવ- ઇન્ટીજર દાખલ કરો."

    # બેઝ કેસ: 0 અથવા 1 નો ફેક્ટોરિયલ 1 છે
```

```
    if n == 0 or n == 1:
        return 1

    # ઇટરેશન વાપરીને ફેક્ટોરિયલ ગણતરી
    result = 1
    for i in range(2, n + 1):
        result *= i

    return result

# ફંક્શન ટેસ્ટ કરો
number = int(input("ફેક્ટોરિયલ શોધવા માટે એક નંબર દાખલ કરો: "))
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 × 2 × 1	6
5	5! = 5 × 4 × 3 × 2 × 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 થી N સુધીના odd અને even નંબર શોધવાનો પ્રોગ્રામ

# યુઝર પાસેથી ઇનપુટ લો
N = int(input("N ની કમિત દાખલ કરો: "))

print("1 થી", N, "સુધીના even નંબર છે:")
for i in range(1, N + 1):
    if i % 2 == 0:
        print(i, end=" ")

print("\n1 થી", N, "સુધીના odd નંબર છે:")
for i in range(1, N + 1):
    if i % 2 != 0:
        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''

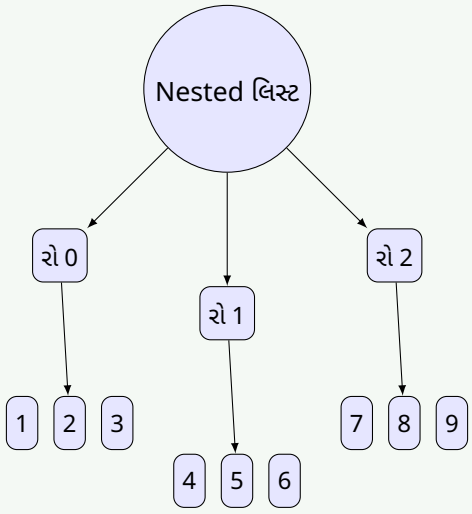
પ્રશ્ન 3(b OR) [4 ગુણ]

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

જવાબ

Listing 14. Nested List Example

```
# Nested લિસ્ટ બનાવવા અને ડિસ્પ્લે કરવાનો પ્રોગ્રામ

# Nested લિસ્ટ બનાવો
nested_list = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# Nested લિસ્ટ ડિસ્પ્લે કરો
print("Nested લિસ્ટ:", nested_list)

# Nested લૂપ્સનો ઉપયોગ કરીને દરેક એલિમેન્ટ ડિસ્પ્લે કરો
print("\nNested લિસ્ટના એલિમેન્ટ્સ:")
for i in range(len(nested_list)):
    for j in range(len(nested_list[i])):
        print(f"nested_list[{i}][{j}] = {nested_list[i][j]}")

# enumerate નો ઉપયોગ કરીને વૈકલ્પિક રીત
print("\nenumerate નો ઉપયોગ કરીને:")
for i, inner_list in enumerate(nested_list):
    for j, value in enumerate(inner_list):
        print(f"પોઝિશન ({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.

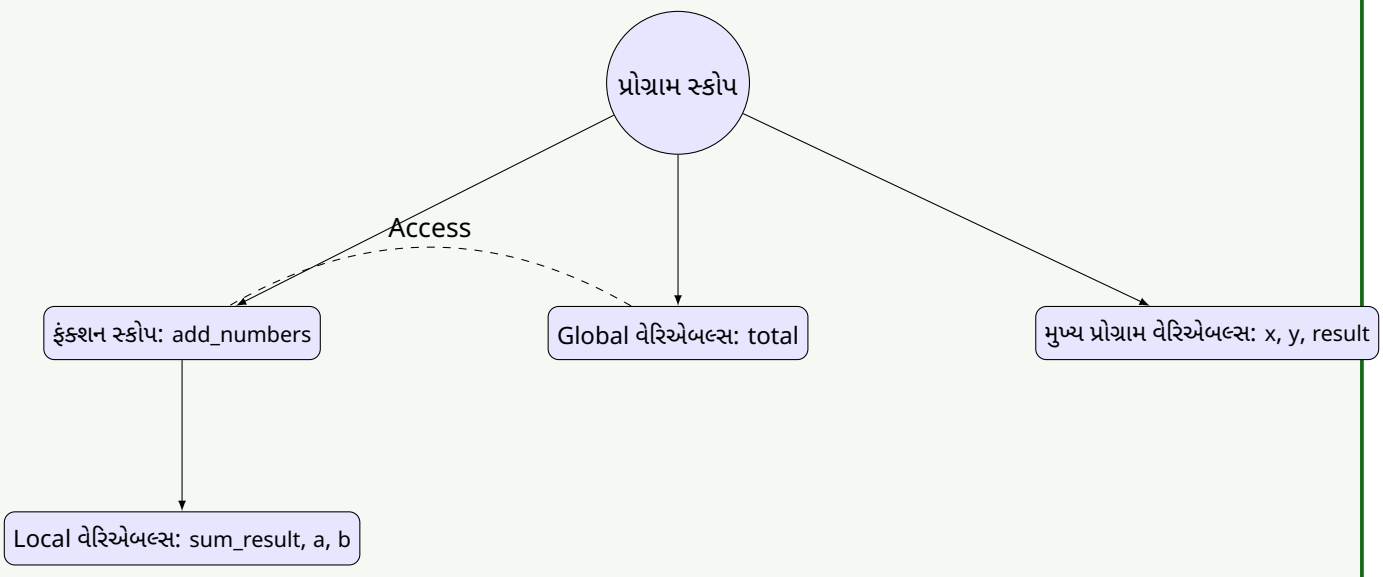
જવાબ

કોષ્ટક 10. Local vs Global વેરિએબલ્સ

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

Listing 15. Global vs Local Variables

```python
# Global વેરએિબલ
total = 0

def add_numbers(a, b):
    # Local વેરએિબલ્સ
    sum_result = a + b
    print(f"Local વેરએિબલ sum_result: {sum_result}")

    # Global વેરએિબલ એક્સેસ કરવું
    print(f"Global વેરએિબલ total મોડફિકિશન પહેલાં: {total}")

    # ફંક્શનની અંદર Global વેરએિબલ મોડફિાય કરવા
    global total
    total = sum_result
    print(f"Global વેરએિબલ total મોડફિકિશન પછી: {total}")

    return sum_result

# મુખ્ય પ્રોગ્રામ
x = 5  # મુખ્ય પ્રોગ્રામમાં Local
y = 10  # મુખ્ય પ્રોગ્રામમાં Local

result = add_numbers(x, y)
print(f"પરણિામ: {result}")
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 ગુણ]

પાયથન ની સ્ટાન્ડર્ડ લાઇબ્રેરી ના મેથેમેટિકલ ફંક્શન લિસ્ટ કરો.

જવાબ

કોષ્ટક 11. પાયથોન Math મોડ્યુલ ફંક્શન્સ

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

મેમરી ટ્રીક

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

પ્રશ્ન 4(b) [4 ગુણ]

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

```
# math મોડ્યુલના ઉપયોગનું ઉદાહરણ
import math

# math મોડ્યુલમાંથી ગાણતિકિ ફંક્શન્સનો ઉપયોગ
radius = 5
area = math.pi * math.pow(radius, 2)
print(f"ત્રજ્યિા {radius} વાળા વર્તુળનું ક્ષેત્રફળ {area:.2f} છે")

# વવિધિ import ટેકનક્િસનો ઉપયોગ
from math import sqrt, sin
angle = math.pi / 4
print(f"25 નું વર્ગમૂળ {sqrt(25)} છે")
print(f"{angle} રેડયિન્સનો સાઇન {sin(angle):.4f} છે")

# alias સાથે import
import random as rnd
random_number = rnd.randint(1, 100)
print(f"1 અને 100 વચ્ચેનો રેન્ડમ નંબર: {random_number}")
```

કોષ્ટક 12. મોડ્યુલ Import ટેકનિક્સ

પદ્ધતિ	સિન્ટેક્સ	ઉદાહરણ
આખો મોડ્યુલ import કરો	import module_name	import math
ચોક્કસ આઇટમ્સ 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 ગુણ]

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

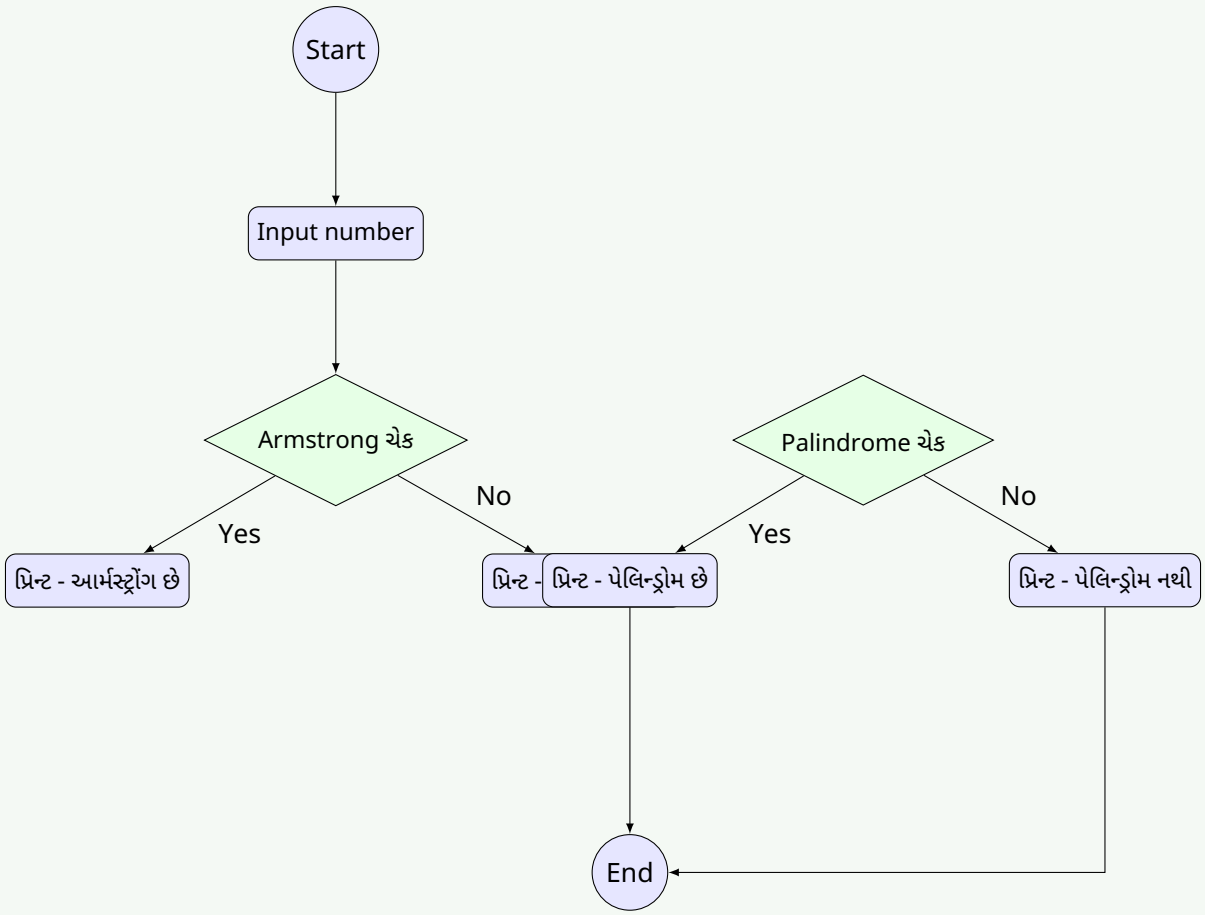
જવાબ

Listing 17. Armstrong and Palindrome Check

```
# નંબર આર્મસ્ટ્રોંગ છે કે નહીં તે ચેક કરવા માટેનું ફંક્શન
def is_armstrong(num):
    # અંકોની સંખ્યા ગણવા માટે નંબરને સ્ટ્રિંગમાં રૂપાંતરિત કરો
    num_str = str(num)
    n = len(num_str)

    # દરેક અંકને અંકોની સંખ્યાની ઘાત સુધી ઉગામેલા સરવાળાની ગણતરી
    armstrong_sum = 0
    for digit in num_str:
        armstrong_sum += int(digit) ** n

    # ચેક કરો કે સરવાળો મૂળ નંબર સાથે મેળ ખાય છે
    return armstrong_sum == num

# નંબર પેલિન્ડ્રોમ છે કે નહીં તે ચેક કરવા માટેનું ફંક્શન
def is_palindrome(num):
    # નંબરને સ્ટ્રિંગમાં રૂપાંતરિત કરો અને ચેક કરો કે તે આગળથી અને પાછળથી એક સરખો વંચાય છે
    num_str = str(num)
    return num_str == num_str[::-1]

# મુખ્ય પ્રોગ્રામ
number = int(input("એક નંબર દાખલ કરો: "))

# ચેક કરો કે નંબર આર્મસ્ટ્રોંગ છે કે નહીં
if is_armstrong(number):
    print(f"{number} એક આર્મસ્ટ્રોંગ નંબર છે")
else:
    print(f"{number} આર્મસ્ટ્રોંગ નંબર નથી")

# ચેક કરો કે નંબર પેલિન્ડ્રોમ છે કે નહીં
if is_palindrome(number):
    print(f"{number} એક પેલિન્ડ્રોમ છે")
else:
    print(f"{number} પેલિન્ડ્રોમ નથી")
```

કોષ્ટક 13. ઉદાહરણો

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



આકૃતિ 9. Armstrong Palindrome Flowchart

મેમરી ટ્રીક

``SAME SUM: SAME આગળ-પાછળ પેલિન્ડ્રોમ માટે, SUM ઘાતના અંકોનો આર્મસ્ટ્રોંગ 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 કર્યા વિના ઉપલબ્ધ છે.

કોષ્ટક 14. સામાન્ય પાયથોન Built-in Functions

ફંક્શન	હેતુ	ઉદાહરણ
print()	આઉટપુટ ડિસ્પ્લે	print("Hello")
input()	યુઝર ઇનપુટ લે	name = input("Name: ")
len()	ઓબ્જેક્ટની લંબાઈ આપે	len([1, 2, 3]) → 3
type()	ઓબ્જેક્ટનો પ્રકાર આપે	type(5) → <class 'int'>
int(), float(), str()	ચોક્કસ પ્રકારમાં રૂપાંતર	int("5") → 5
range()	સીક્વન્સ જનરેટ કરે	list(range(3)) → [0, 1, 2]
sum()	સરવાળો ગણે	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

```
# math મોડ્યુલનો ઉપયોગ કરતો ઉદાહરણ
import math

# મૂળભૂત સ્થિરાંકો
print(f"pi ની કિંમત: {math.pi}")
print(f"e ની કિંમત: {math.e}")

# ત્રિકોણમિતિ ફંક્શન્સ આર્ગ્યુમેન્ટ( રેડિયન્સમાં)
angle = math.pi / 3  # 60 ડિગ્રી
print(f"{angle:.2f} રેડિયન્સનો સાઇન: {math.sin(angle):.4f}")
print(f"{angle:.2f} રેડિયન્સનો કોસાઇન: {math.cos(angle):.4f}")
print(f"{angle:.2f} રેડિયન્સનો ટેન્જન્ટ: {math.tan(angle):.4f}")

# લોગરિધમિક અને એક્સપોનેન્શિયલ ફંક્શન્સ
x = 10
print(f"{x} નો નેચરલ લોગરિધમ: {math.log(x):.4f}")
print(f"{x} નો લોગરિધમ બેઝ 10: {math.log10(x):.4f}")
print(f"e ની {x} ઘાત: {math.exp(x):.4f}")

# અન્ય ફંક્શન્સ
print(f"25 નું વર્ગમૂળ: {math.sqrt(25)}")
print(f"4.3 નો સીલિંગ: {math.ceil(4.3)}")
print(f"4.7 નો ફ્લોર: {math.floor(4.7)}")
```

કોષ્ટક 15. Math મોડ્યુલ કેટેગરીઝ

કેટેગરી	ફંક્શન્સ
સ્થિરાંકો	math.pi, math.e
ત્રિકોણમિતિ	sin(), cos(), tan()
લોગરિધમિક	log(), log10(), exp()
ન્યુમેરિક	sqrt(), ceil(), floor()

મેમરી ટ્રીક

``PENT: Pi/constants, Exponents, Numbers, Trigonometry''

પ્રશ્ન 4(c OR) [7 ગુણ]

પાયથોનમાં વેરીએબલના અવકાશનો કોન્સેપ્ટ સમજાવો અને પાયથોન પ્રોગ્રામમાં વૈશ્વિક અને સ્થાનિક વેરીએબલ કોન્સેપ્ટ લાગુ કરો.

જવાબ

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

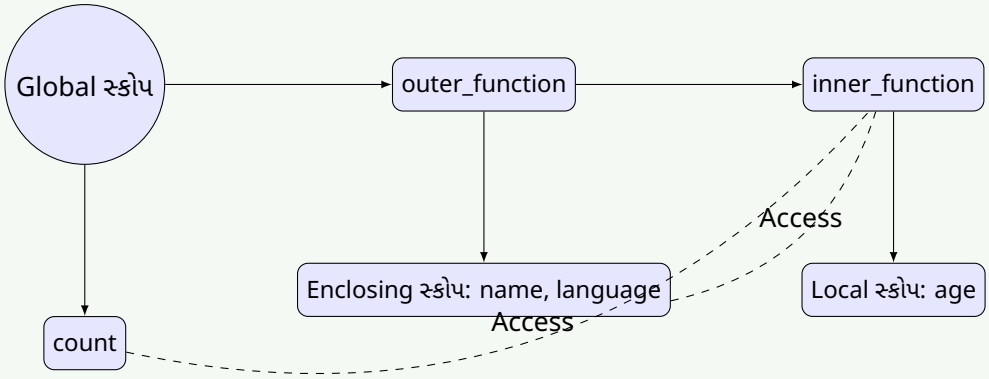
કોષ્ટક 16. વેરિએબલ સ્કોપના પ્રકારો

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

Listing 19. Variable Scope Demonstration

```
# વેરિએબલ સ્કોપ ડેમોન્સ્ટ્રેશન

# Global વેરિએબલ
count = 0

def outer_function():
    # Enclosing સ્કોપ વેરિએબલ
    name = "Python"

    def inner_function():
        # Local વેરિએબલ
        age = 30
        # Global વેરિએબલ એક્સેસ કરવું
        global count
        count += 1
        # Enclosing વેરિએબલ એક્સેસ કરવું
        print(f"inner_function ની અંદર: name is {name}")
        print(f"inner_function ની અંદર: age is {age}")
        print(f"inner_function ની અંદર: count is {count}")

    # outer_function માટે Local વેરિએબલ
    language = "Programming"
    print(f"outer_function ની અંદર: name is {name}")
    print(f"outer_function ની અંદર: language is {language}")
    print(f"outer_function ની અંદર: count is {count}")

    # ઇનર ફંક્શન કોલ કરો
    inner_function()

    # આ ભૂલ આપશે - age એ inner_function માટે Local છે
    # print(age)

# મુખ્ય પ્રોગ્રામ
print(f"Global સ્કોપ: count is {count}")
outer_function()
print(f"ફંક્શન કોલ પછી Global સ્કોપ: count is {count}")
```

```
# આ ભૂલ આપશે - તેઓ ફંક્શન્સ માટે Local છે
# print(name)
# print(language)
```



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

```
# લિસ્ટમાં બે એલિમેન્ટ્સ સ્વેપ કરવાનો પ્રોગ્રામ

# એક લિસ્ટ બનાવો
my_list = [10, 20, 30, 40, 50]
print("મૂળ લિસ્ટ:", my_list)

# સ્વેપ કરવા માટેની પોઝિશન મેળવો
pos1 = int(input("પ્રથમ પોઝિશન દાખલ કરો ઇન્ડેક્સ( 0 થી શરૂ થાય છે): "))
pos2 = int(input("બીજી પોઝિશન દાખલ કરો ઇન્ડેક્સ( 0 થી શરૂ થાય છે): "))

# ટેમ્પરરી વેરિએબલનો ઉપયોગ કરીને એલિમેન્ટ્સ સ્વેપ કરો
if 0 <= pos1 < len(my_list) and 0 <= pos2 < len(my_list):
    # સ્વેપિંગ
    temp = my_list[pos1]
    my_list[pos1] = my_list[pos2]
    my_list[pos2] = temp

    print(f"પોઝિશન {pos1} અને {pos2} પર એલિમેન્ટ્સ સ્વેપ કર્યા પછી લિસ્ટ:", my_list)
else:
    print("અમાન્ય પોઝિશન! પોઝિશન લિસ્ટની રેન્જની અંદર હોવી જોઈએ.")
```

વૈકલ્પિક પદ્ધતિ:

Listing 21. Pythonic Swap

```
# પાયથોનની tuple અનપેકિંગનો ઉપયોગ કરીને સ્વેપ વધુ પાયથોનિક)
if 0 <= pos1 < len(my_list) and 0 <= pos2 < len(my_list):
    my_list[pos1], my_list[pos2] = my_list[pos2], my_list[pos1]
    print(f"પોઝિશન {pos1} અને {pos2} પર એલિમેન્ટ્સ સ્વેપ કર્યા પછી લિસ્ટ:", my_list)
```

કોષ્ટક 17. સ્વેપિંગ પદ્ધતિઓ

પદ્ધતિ	કોડ
ટેમ્પ 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]

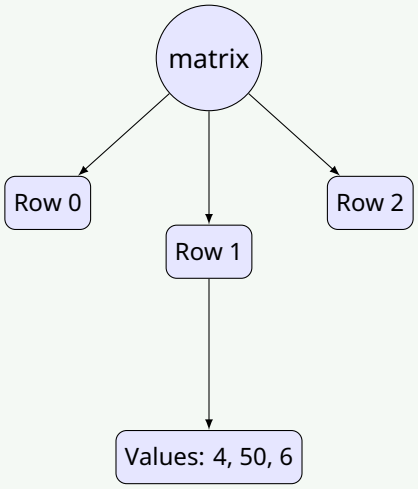
ઉદાહરણ આપીને નેસ્ટેડ લિસ્ટ સમજાવો.

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

```
# નેસ્ટેડ લિસ્ટ બનાવવું (3x3 મેટ્રિક્સ)
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# એલિમેન્ટ્સ એક્સેસ કરવા
print("સંપૂર્ણ મેટ્રિક્સ:", matrix)
print("પ્રથમ રો:", matrix[0])
print("રો 1, કોલમ 2 પર એલિમેન્ટ:", matrix[0][1])  # આઉટપુટ: 2

# એલિમેન્ટ્સ મોડિફાય કરવા
matrix[1][1] = 50
print("મોડિફિકેશન પછી મેટ્રિક્સ:", matrix)

# નેસ્ટેડ લિસ્ટ માં લૂપ ફેરવવું
print("\nમેટ્રિક્સn પ્રિન્ટ કરવું:")
for row in matrix:
    for element in row:
        print(element, end=" ")
    print()  # દરેક રો પછી નવી લાઇન
```



આકૃતિ 11. Nested List Structure

કોષ્ટક 18. Nested List Operations

ઓપરેશન	સિન્ટેક્સ	ઉદાહરણ
એલિમેન્ટ એક્સેસ	list[row][col]	matrix[0][1]
એલિમેન્ટ મોડિફાય	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 ગુણ]

ઉદાહરણો સાથે સ્ટ્રિંગ ઓપરેશન્સ સમજાવો.

જવાબ

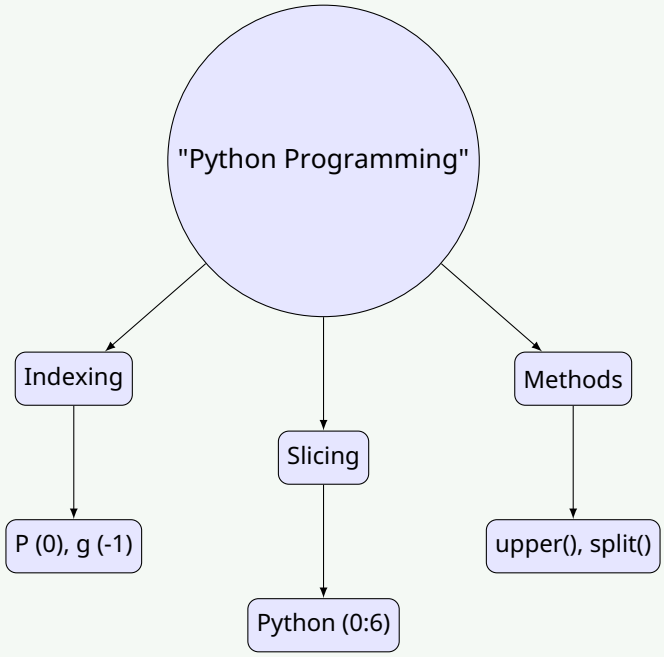
કોષ્ટક 19. પાયથોનમાં સ્ટ્રિંગ ઓપરેશન્સ

ઓપરેશન	વર્ણન	ઉદાહરણ
Concatenation	સ્ટ્રિંગ્સ જોડવા	"Hello" + " World" → "Hello World"
Repetition	સ્ટ્રિંગ્સ રિપીટ કરવા	"Python" * 3 → "PythonPythonPython"
Slicing	સબસ્ટ્રિંગ એક્સટ્રેક્ટ 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

```
# સ્ટ્રિંગ ઓપરેશન્સ ડેમોન્સ્ટ્રેશન
text = "Python Programming"

# Indexing
print("૨ પ્રથમ કેરેક્ટર:", text[0])
```

```
print("છેલ્લો કેરેક્ટર:", text[-1])

# Slicing
print("પ્રથમ શબ્દ:", text[:6])
print("બીજો શબ્દ:", text[7:])
print("મધ્ય કેરેક્ટર્સ:", text[3:10])
print("ઉલટું (Reverse):", text[::-1])

# સ્ટ્રિંગ મેથડ્સ
print("અપરકેસ:", text.upper())
print("લોઅરકેસ:", text.lower())
print("'P' ને 'J' સાથે બદલો:", text.replace("P", "J"))
print("સ્પેસ દ્વારા સ્પ્લિટ:", text.split())
print("'m' ગણો:", text.count('m'))
print("'gram' શોધો:", text.find("gram"))

# ચેક ઓપરેશન્સ
print("શું આલ્ફાન્યુમેરિક છે?", text.isalnum())
print("શું 'Py' થી શરૂ થાય છે?", text.startswith("Py"))
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

```
# લિસ્ટમાં રહેલા તમામ એલિમેન્ટ્સનો સરવાળો શોધવાનો પ્રોગ્રામ

# પદ્ધતિ 1: બિલ્ટઇન- sum() ફંક્શનનો ઉપયોગ કરીને
def sum_list_builtin(numbers):
    return sum(numbers)

# પદ્ધતિ 2: લૂપનો ઉપયોગ કરીને
def sum_list_loop(numbers):
    total = 0
    for num in numbers:
        total += num
    return total

# એક સેમ્પલ લિસ્ટ બનાવો
my_list = [10, 20, 30, 40, 50]
print("લિસ્ટ:", my_list)

# બિલ્ટઇન- ફંક્શનનો ઉપયોગ કરીને સરવાળો ગણો
print("બિલ્ટઇન- ફંક્શનનો ઉપયોગ કરીને સરવાળો:", sum_list_builtin(my_list))

# લૂપનો ઉપયોગ કરીને સરવાળો ગણો
print("લૂપનો ઉપયોગ કરીને સરવાળો:", sum_list_loop(my_list))
```

કોષ્ટક 20. સરવાળા પદ્ધતિઓની તુલના

પદ્ધતિ	ફાયદો
Built-in sum()	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 ઈન્ડેક્સીંગ અને સ્લાઇસિંગ ઓપરેશન્સ સમજાવો

જવાબ

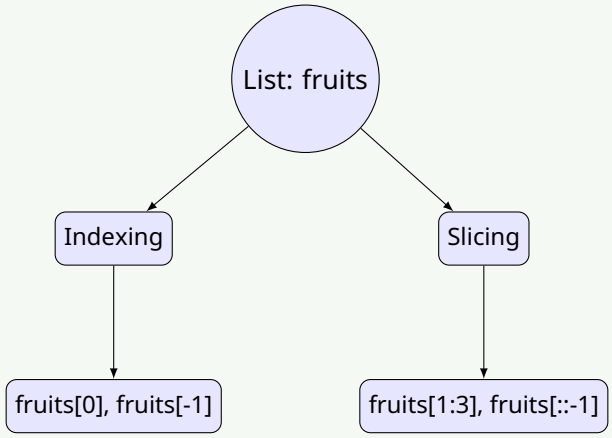
કોષ્ટક 21. ઇન્ડેક્સિંગ અને સ્લાઇસિંગ ઓપરેશન્સ

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

Listing 25. Indexing and Slicing Demo

```
# ઇન્ડેક્સિંગ અને સ્લાઇસિંગ ડેમોન્સ્ટ્રેશન
```

```
fruits = ["apple", "banana", "cherry", "date", "elderberry", "fig"]
print("મૂળ લિસ્ટ:", fruits)

# Indexing
print("\nઇન્ડેક્સિંગ ઉદાહરણો:")
print("પ્રથમ આઈટમ:", fruits[0])  # apple
print("છેલ્લી આઈટમ:", fruits[-1])  # fig

# Slicing
print("\nસ્લાઇસિંગ ઉદાહરણો:")
print("પ્રથમ ત્રણ આઈટમ્સ:", fruits[:3])
print("છેલ્લી ત્રણ આઈટમ્સ:", fruits[-3:])
print("મધ્ય આઈટમ્સ:", fruits[2:4])
print("દરેક બીજી આઈટમ:", fruits[::2])
print("ઉલટું લિસ્ટ:", fruits[::-1])
```



આકૃતિ 13. Indexing and Slicing

મેમરી ટ્રીક

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

પ્રશ્ન 5(c OR) [7 ગુણ]

જરૂરી ઉદાહરણ સાથે tuple ને ટૂંકમાં સમજાવો.

જવાબ

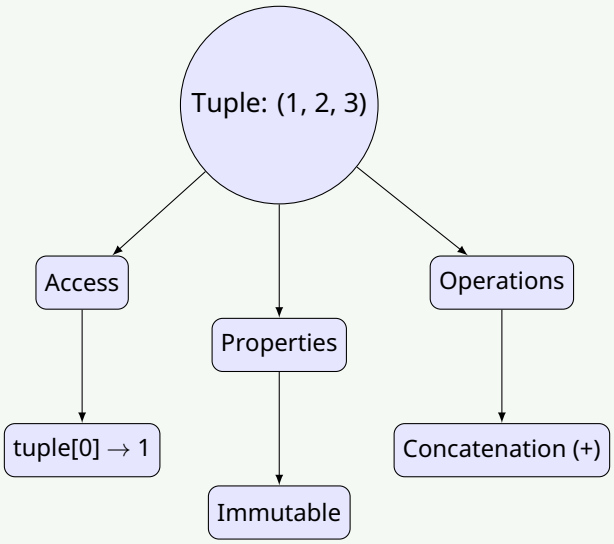
Tuple: Tuple એ એલિમેન્ટ્સનું ઓર્ડર્ડ, અપરિવર્તનીય (immutable) કલેક્શન છે. એકવાર બનાવ્યા પછી, એલિમેન્ટ્સ બદલી શકાતા નથી.

કોષ્ટક 22. Tuple vs List

ફીચર	Tuple	List
સિન્ટેક્સ	(item1, item2)	[item1, item2]
મ્યુટેબિલિટી	Immutable (બદલી શકાતું નથી)	Mutable (બદલી શકાય છે)
પરફોર્મન્સ	ઝડપી	ધીમું
ઉપયોગ	ફિક્સ્ડ ડેટા, કીઝ	ડાયનેમિક ડેટા

Listing 26. Tuple Example

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

# Tuple એલમેન્ટ્સ એક્સેસ કરવા
print("પ્રથમ આઈટમ:", mixed_tuple[0])  # 1
print("નેસ્ટેડ ટપલ એલમેન્ટ:", nested_tuple[2][0])  # 3

# Tuple ઓપરેશન્સ
combined_tuple = mixed_tuple + nested_tuple
print("સંયુક્ત ટપલ:", combined_tuple)

# આ ભૂલ આપશે કારણ કે tuples અપરિવર્તનીય છે
# 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''