

Subject Name (Gujarati)

1323203 -- Winter 2024

Semester 1 Study Material

Detailed Solutions and Explanations

પ્રશ્ન 1(a) [3 ગુણ]

ફ્લોચાર્ટને વ્યાખ્યાયિત કરો અને ફ્લોચાર્ટના કોઈપણ ચાર પ્રતીકોની સૂચિ બનાવો.

જવાબ

ફ્લોચાર્ટ એ એક પ્રક્રિયા, એલોરિધમ અથવા પ્રોગ્રામમાં પગલાંઓના કમને દર્શાવવા માટે માનક પ્રતીકોનો ઉપયોગ કરતું ચિત્રાત્મક પ્રતિનિધિત્વ છે.

સામાન્ય ફ્લોચાર્ટ પ્રતીકો:

પ્રતીક	નામ	હેતુ
લંબગોળ/ગોળાકાર આયત	Terminal/Start/End Process	પ્રક્રિયાની શરૂઆત અથવા અંત દર્શાવે છે ગણતરી અથવા ડેટા પ્રોસેસિંગનું પ્રતિનિધિત્વ કરે છે
હીરા આકાર	Decision	શરતી શાખાના બિંદુને દર્શાવે છે
સમાંતર ચતુર્ભુજોણ	Input/Output	ડેટા ઈનપુટ અથવા આઉટપુટનું પ્રતિનિધિત્વ કરે છે

મેમરી ટ્રીક

"TP-DI" (Terminal-Process-Decision-Input/Output)

પ્રશ્ન 1(b) [4 ગુણ]

પાયથોનમાં વિવિધ ડેટા પ્રકારોની યાદી બનાવો. કોઈપણ ત્રણ ડેટા પ્રકારો ઉદાહરણ સાથે સમજાવો.

જવાબ

પાયથોનના ડેટા પ્રકારો વિવિધ પ્રકારની ડેટા કિંમતોને વર્ગીકૃત કરે છે.

ડેટા પ્રકાર	વર્ણન	ઉદાહરણ
Integer	દર્શાંશ બિંદુઓ વિનાના સંપૂર્ણ સંખ્યાઓ	x = 10
Float	દર્શાંશ બિંદુઓ સાથેની સંખ્યાઓ	y = 3.14
String	અક્ષરોની શ્રેણી	name = "Python"
Boolean	સાચું અથવા ખોટું મૂલ્યો	is_valid = True
List	ક્રમબદ્ધ, પરિવર્તનશીલ સંગ્રહ	colors = ["red", "green"]
Tuple	ક્રમબદ્ધ, અપરિવર્તનીય સંગ્રહ	point = (5, 10)
Dictionary	કી-વેલ્યુ જોડીઓ	person = \{"name": "John"\}
Set	અવ્યવસ્થિત અનન્ય આઈટમોનો સંગ્રહ	unique = \{1, 2, 3\}

Integer: દરારાં બિંદુઓ વિનાની સંપૂર્ણ સંખ્યાઓનું પ્રતિનિધિત્વ કરે છે.

```
1 age = 25  
2 count = -10
```

String: અવતરણ ચિહ્નોમાં બંધ અક્ષરોના ક્રમનું પ્રતિનિધિત્વ કરે છે.

```
1 name = "Python"  
2 message = 'Hello World'
```

List: વિવિધ પ્રકારની વસ્તુઓનો ક્રમબદ્ધ, પરિવર્તનશીલ સંગ્રહ.

```
1 numbers = [1, 2, 3, 4]  
2 mixed = [1, "Python", True, 3.14]
```

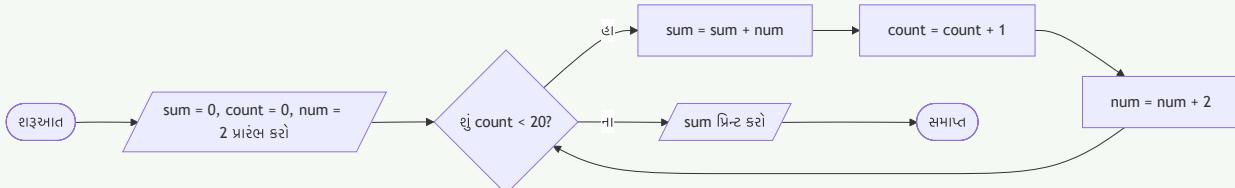
મેમરી ટ્રીક

“FIBS-LTDS” (Float-Integer-Boolean-String-List-Tuple-Dictionary-Set)

પ્રશ્ન 1(c) [7 ગુણ]

પ્રથમ વીસ સમાન પ્રાકૃતિક સંખ્યાઓના સરવાળાની ગણતરી કરવા માટે ફ્લોચાર્ટ ડિઝાઇન કરો.

જવાબ



સમજૂતી:

- ચલનો પ્રારંભ: sum=0, count=0 (મને સમ સંખ્યાઓને ટ્રેક કરવા માટે), num=2 (પ્રથમ સમ સંખ્યા)
- લૂપ શરત: 20 સમ સંખ્યાઓ મળે ત્યાં સુધી ચાલુ રાખો
- પ્રક્રિયા: વર્તમાન સમ સંખ્યાને સરવાળામાં ઉમેરો
- અપડેટ: કાઉન્ટર વધારો અને આગળની સમ સંખ્યા પર જાઓ
- આઉટપુટ: લૂપ પૂર્ણ થાય ત્યારે અંતિમ સરવાળો પ્રિન્ટ કરો

મેમરી ટ્રીક

“SCNL-20” (Sum-Count-Number-Loop until 20)

પ્રશ્ન 1(c) અથવા [7 ગુણ]

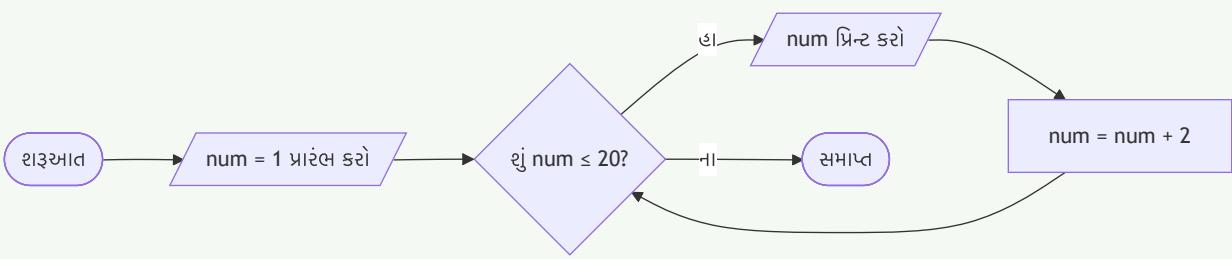
1 થી 20 ની વચ્ચેની વિષમ સંખ્યાઓ પ્રિન્ટ કરવા માટે એલોરિધમ બનાવો.

જવાબ

એલોરિધમ:

- ચલ નુંમ = 1 (પ્રથમ વિષમ સંખ્યાથી શરૂ કરીને) પ્રારંભ કરો
- જ્યાં સુધી નુંમ $\leq 20, 3 - 5$
- નુંમ ની કિમત પ્રિન્ટ કરો
- નુંમ ને 2 વધારો (આગળની વિષમ સંખ્યા મેળવવા માટે)
- પગલું 2 થી પુનરાવર્તન કરો
- સમાપ્ત

આફ્ટિટુડ:



કોડ અમલીકરણ:

```

1 # 1   20
2 num = 1
3 while num <= 20:
4     print(num)
5     num += 2

```

મેમરી ટ્રીક

“SOLO-20” (Start Odd Loop Output until 20)

પ્રશ્ન 2(a) [3 ગુણ]

પાયથોનના સભ્યપદ ઓપરેટર વિશે ચર્ચા કરો.

જવાબ

પાયથોનમાં સભ્યપદ ઓપરેટરનો ઉપયોગ કોઈ મૂલ્ય અથવા ચલ અનુક્રમમાં અસ્થિત્વમાં છે કે નહીં તેનું પરીક્ષણ કરવા માટે થાય છે.
સભ્યપદ ઓપરેટરની સારણી:

ઓપરેટર	વર્ણન	ઉદાહરણ	આઉટપુટ
in	જો મૂલ્ય અનુક્રમમાં અસ્થિત્વમાં હોય તો True પરત કરે છે	5 in [1,2,5]	True
not in	જો મૂલ્ય અસ્થિત્વમાં ન હોય તો True પરત કરે છે	4 not in [1,2,5]	True

સામાન્ય ઉપયોગ:

- લિસ્ટમાં તત્ત્વ અસ્થિત્વમાં છે કે નહીં તેની તપાસ કરવી: if item in my_list:
- શબ્દકોશમાં કી અસ્થિત્વમાં છે કે નહીં તેની તપાસ કરવી: if key in my_dict:
- સબસ્ટિંગ અસ્થિત્વમાં છે કે નહીં તેની તપાસ કરવી: if "py" in "python":

મેમરી ટ્રીક

“IM-NOT” (In Membership - NOT in Membership)

પ્રશ્ન 2(b) [4 ગુણ]

continue અને break સ્ટેમેન્ટ્ની જરૂરિયાત સમજાવો.

જવાબ

સ્ટેમેન્ટ	હેતુ	ઉપયોગ કેસ	ઉદાહરણ
break	લૂપને તાત્કાલિક સમાપ્ત કરે છે	જ્યારે શરત પૂરી થાય ત્યારે લૂપમાંથી બહાર નીકળો	તત્ત્વ શોધવું
continue	વર્તમાન પુનરાવર્તનને છોડી આગામી વધું	અમૃક મૂલ્યોને છોડી આગામી વધું	ફિલ્ટરિંગ મૂલ્યો

Break સ્ટેપેન્ટ:

- હેતુ: તાત્કાલિક લૂપમાંથી બહાર નીકળે છે
- ક્યારે ઉપયોગ કરવો: જ્યારે જરૂરી શરત હાંસલ થાય અને વધુ પ્રક્રિયાની જરૂર ન હોય
- ઉદાહરણ: લિસ્ટમાં ચોક્કસ તત્વ શોધવું

```

1 for num in range(1, 10):
2     if num == 5:
3         print("Found 5!")
4         break
5     print(num)

```

Continue સ્ટેપેન્ટ:

- હેતુ: વર્તમાન પુનરાવર્તનને છોડી આગળના પર જાય છે
- ક્યારે ઉપયોગ કરવો: જ્યારે અમૃક મૂલ્યોને છોડવાના હોય પરંતુ લૂપ ચાલુ રાખવાનો હોય
- ઉદાહરણ: લૂપમાં સમ સંખ્યાઓને છોડવી

```

1 for num in range(1, 10):
2     if num % 2 == 0:
3         continue
4     print(num) #

```

મેમરી ટ્રીક

“BS-CE” (Break Stops, Continue Expects)

પ્રશ્ન 2(c) [7 ગુણ]

યુઝર તરફથી ઇનપુટ તરીકે લેવામાં આવેલા ચાર વિષયના ગુણના આધારે કુલ અને સરેરાશ ગુણની ગણતરી કરવા માટે એક પ્રોગ્રામ બનાવો.

જવાબ

```

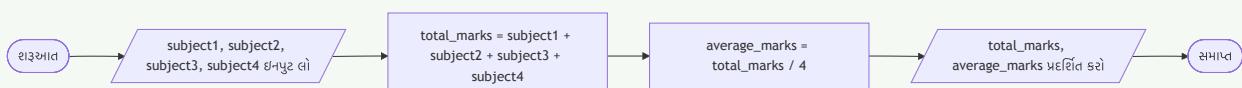
1 #
2 #
3 subject1 = float(input("    1      : "))
4 subject2 = float(input("    2      : "))
5 subject3 = float(input("    3      : "))
6 subject4 = float(input("    4      : "))

7 #
8 total_marks = subject1 + subject2 + subject3 + subject4
average_marks = total_marks / 4

10 #
11 print(f"    : {total_marks}")
12 print(f"    : {average_marks}")

```

આકૃતિ:



સમજૂતી:

- ઇનપુટ: યુઝર પાસેથી ચાર વિષયોના ગુણ મેળવો
- પ્રક્રિયા: બધા વિષયના ગુણને ઉમેરીને કુલ અને વિષયોની સંખ્યા વડે ભાગીને સરેરાશની ગણતરી કરો
- આઉટપુટ: કુલ અને સરેરાશ ગુણ પ્રદર્શિત કરો

મેમરી ટ્રીક

“IAPO” (Input-Add-Process-Output)

પ્રશ્ન 2(a) અથવા [3 ગુણ]

અસાઇનમેન્ટ ઓપરેટર પર ટૂંકી નોંધ લખો.

જવાબ

પાયથોનમાં અસાઇનમેન્ટ ઓપરેટરનો ઉપયોગ ચલોને મૂલ્યો સૌંપવા માટે થાય છે.

ઓપરેટર	નામ	વર્ણન	ઉદાહરણ
=	સરળ અસાઇનમેન્ટ	જમણા ઓપરન્ડ મૂલ્યને ડાબા ઓપરન્ડને સૌંપે છે	x = 10
+=	ઉમેરો અને સૌંપો	જમણા ઓપરન્ડને ડાબામાં ઉમેરે અને પરિણામ સૌંપે છે	x += 5 (x = x + 5 સમાન)
-=	બાદ કરો અને સૌંપો	જમણા ઓપરન્ડને ડાબામાંથી બાદ કરે અને સૌંપે છે	x -= 3 (x = x - 3 સમાન)
*=	ગુણાકાર અને સૌંપો	ડાબાને જમણા વડે ગુણાકાર કરે અને સૌંપે છે	x *= 2 (x = x * 2 સમાન)
/=	ભાગાકાર અને સૌંપો	ડાબાને જમણા વડે ભાગો અને સૌંપે છે	x /= 4 (x = x / 4 સમાન)

મિશ્રિત અસાઇનમેન્ટ ઓપરેટર અંકગણિતીય ઓપરેશન અને અસાઇનમેન્ટને જોડે છે, જેથી કોડ વધુ સંક્ષિપ્ત અને વાંચવા યોગ્ય બને છે.

મેમરી ટ્રીક

“SAME” (Simple Assignment Makes Easy)

પ્રશ્ન 2(b) અથવા [4 ગુણ]

for લૂપનો ઉપયોગ સિન્ટેક્સ, ફ્લોયાર્ટ અને ઉદાહરણ આપીને સમજાવો.

જવાબ

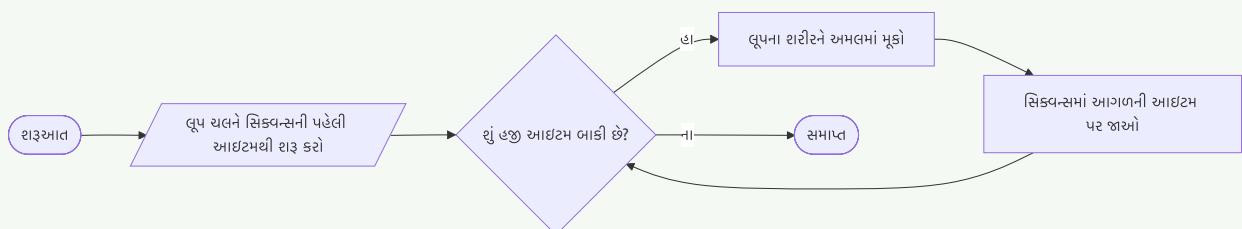
For લૂપનો સિન્ટેક્સ:

```

1 for variable in sequence:
2     #

```

ફ્લોયાર્ટ:



ઉદાહરણ:

```

1 # 1   5
2 for num in range(1, 6):
3     square = num ** 2
4     print(f"{num}      = {square}")

```

પાયથોનમાં for લૂપનો ઉપયોગ સિક્વન્સ (લિસ્ટ, ટપલ, સ્ટ્રિંગ, વગેરે) અથવા અન્ય ઇટરેબલ ઓફ્જેક્ટ્સ પર ચોક્કસ પુનરાવર્તન માટે થાય છે. તે ખાસ કરીને ત્યારે ઉપયોગી છે જ્યારે તમે પુનરાવર્તનોની સંખ્યા અગાઉથી જાણતા હો.

મેમરી ટ્રીક

“SIFE” (Sequence Iteration For Each item)

પ્રશ્ન 2(c) અથવા [7 ગુણ]

યુઝર દ્વારા આપેલ નંબરનો વર્ગ અને ઘન શોધવા માટે કોડ વિકસાવો.

જવાબ

```

1 # 
2 #
3 num = float(input("           : "))
4 #
5 square = num ** 2
6 cube = num ** 3
7 #
8 print(f"           : {num}")
9 print(f"{num}      : {square}")
10 print(f"{num}     : {cube}")

```

આફ્ટિં:



સમજૂતી:

- ઇનપુટ: યુઝર પાસેથી નંબર મેળવો
- પ્રક્રિયા: 2ની ઘાત પર ઉઠાવીને વર્ગ, 3ની ઘાત પર ઉઠાવીને ઘનની ગણતરી કરો
- આઉટપુટ: ઇનપુટ નંબર, તેનો વર્ગ અને ઘન પ્રદર્શિત કરો

મેમરી ટ્રીક

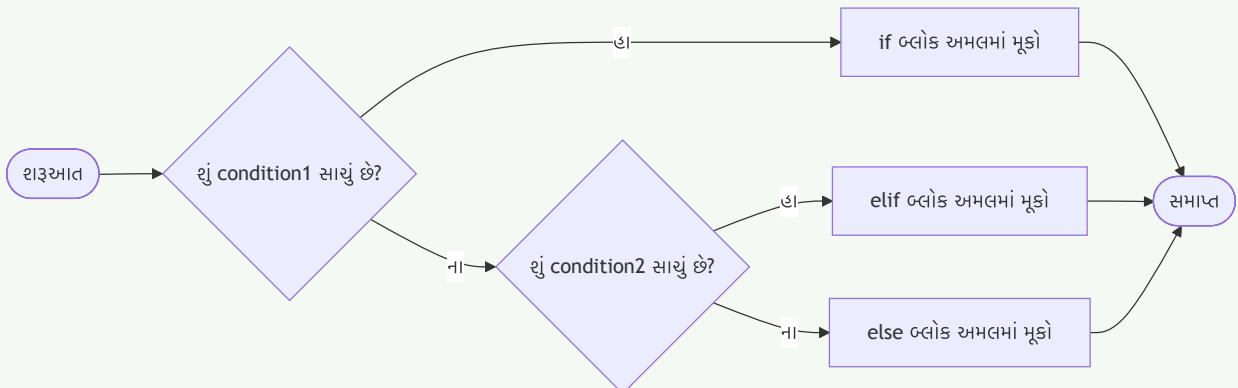
"ISCO" (Input-Square-Cube-Output)

પ્રશ્ન 3(a) [3 ગુણ]

if-elif-else સ્ટેટમેન્ટને ફ્લોચાર્ટ અને ચોંચ ઉદાહરણ સાથે સમજાવો.

જવાબ

પાયથોનમાં if-elif-else સ્ટેટમેન્ટ એ એવી શરતી ક્રિયા માટે છે જ્યાં ઘણા અભિવ્યક્તિઓનું મૂલ્યાંકન કરવામાં આવે છે.
ફ્લોચાર્ટ:



ઉદાહરણ:

```

1 #
2 marks = 75
3
4 if marks >= 90:
5     grade = "A"
6 elif marks >= 80:
7     grade = "B"

```

```

8 elif marks >= 70:
9     grade = "C"
10 elif marks >= 60:
11     grade = "D"
12 else:
13     grade = "F"
14
15 print(f"      : {grade}")

```

મેમરી ટ્રીક

“CITE” (Check If Then Else)

પ્રશ્ન 3(b) [4 ગુણ]

યુઝર ડિફાઇન ફંક્શન વ્યાખ્યાયિત કરો અને કેવી રીતે યુસર ડિફાઇન ફંક્શન કોલ કરતું તે યોગ્ય ઉદાહરણ આપીને સમજાવો.

જવાબ

ફંક્શન વ્યાખ્યા અને કોડિંગ:

પાસું	સિન્ટેક્સ	હેતુ
વ્યાખ્યા	def function_name(parameters):	પુનઃઉપયોગી કોડનો બ્લોક બનાવે છે
ફંક્શન બોડી	ઇન્ટેન્ડ કોડ બ્લોક	ફંક્શનનો લોજિક ધરાવે છે
રીટર્ન સ્ટેટમેન્ટ	return [expression]	કોલરને મૂલ્ય પાછું મોકલે છે
ફંક્શન કોલ	function_name(arguments)	ફંક્શન કોડ ચલાવે છે

ફંક્શન વ્યાખ્યાયિત અને કોલ કરવાનું ઉદાહરણ:

```

1 #
2 def calculate_area(length, width):
3     """
4         area = length * width
5     return area
6
7 #
8 result = calculate_area(5, 3)
9 print(f"      : {result}")

```

સમજૂતી:

- ફંક્શન વ્યાખ્યા: def કીવર્કનો ઉપયોગ કરીને ફંક્શન નામ અને પેરામીટર્સ સાથે
- ડોક્યુમેન્ટેશન: ફંક્શનનું વર્ણન કરતું વૈકલ્પિક ડોક્સિંગ
- ફંક્શન બોડી: કાર્ય કરતો કોડ
- રીટર્ન સ્ટેટમેન્ટ: કોલરને પરિણામ પાછું મોકલે છે
- ફંક્શન કોલ: ફંક્શન ચલાવવા માટે આર્થ્યુમેન્ટ્સ પસાર કરો

મેમરી ટ્રીક

“DBRCA” (Define-Body-Return-Call-Arguments)

પ્રશ્ન 3(c) [7 ગુણ]

આપેલ નંબરનો ફેક્ટોરીયલ શોધવા માટે કોડ વિકસાવો.

જવાબ

```

1 #
2 #
3 num = int(input("      : "))

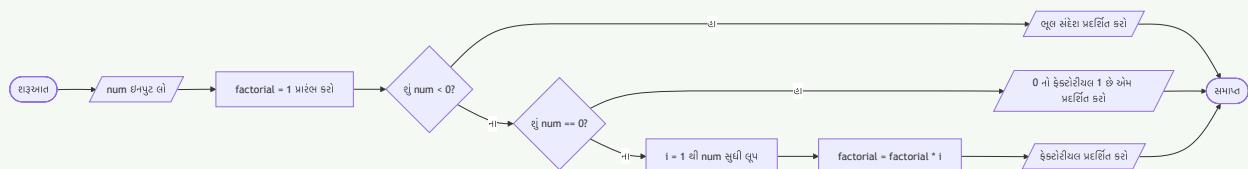
```

```

4
5  #
6 factorial = 1
7
8  #
9 if num < 0:
10    print("          ")
11 elif num == 0:
12    print("0      1  ")
13 else:
14    #
15    for i in range(1, num + 1):
16        factorial *= i
17    print(f"{num}      {factorial}  ")

```

આફ્ટરિટિંગ:



સમજૂતી:

- ઇનપુટ: ચુક્કર પાસેથી નંબર મેળવો
- ચકાસણી: તપાસો કે નંબર નકારાત્મક (ફેકટોરીયલ વ્યાખ્યાયિત નથી), શૂન્ય (ફેકટોરીયલ 1 છે), અથવા સકારાત્મક છે
- પ્રક્રિયા: સકારાત્મક નંબરો માટે, ફેકટોરીયલને 1 થી num સુધીના દરેક નંબર સાથે ગુણાકાર કરો
- આઉટપુટ: ફેકટોરીયલ પરિણામ પ્રદર્શિત કરો

મેમરી ટ્રીક

“MICE” (Multiply Incrementally, Check Edge-cases)

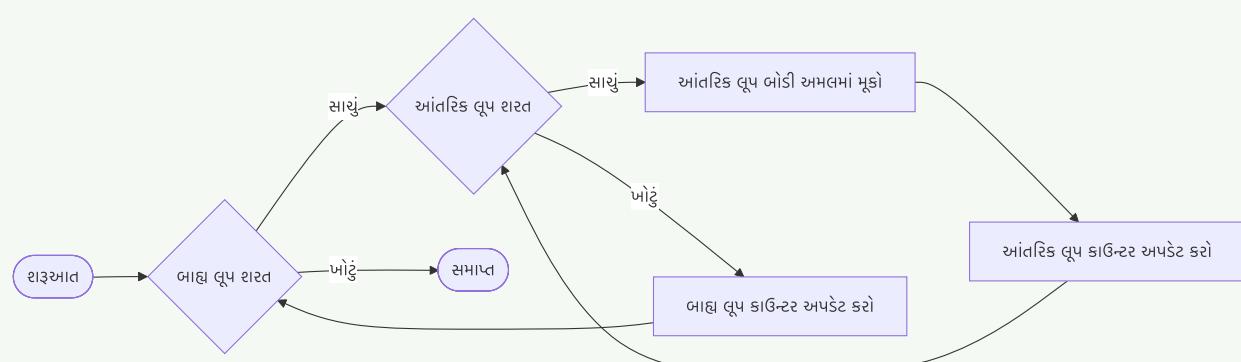
પ્રશ્ન 3(a) અથવા [3 ગુણ]

યોગ્ય ઉદાહરણનો ઉપયોગ કરીને નેર્ટેડ લૂપ સમજાવો.

જવાબ

નેર્ટેડ લૂપ એ એક લૂપની અંદર બીજું લૂપ છે. બાધ્ય લૂપના દરેક પુનરાવર્તન માટે આંતરિક લૂપ તેના બધા પુનરાવર્તનો પૂર્ણ કરે છે.

આફ્ટરિટિંગ:



ઉદાહરણ:

```

1 # 1   3
2 for i in range(1, 4): #      : 1   3
3     print(f"{i}           ")
4     for j in range(1, 6): #      : 1   5
5         print(f"{i} x {j} = {i*j}")
6     print() #

```

મેમરી ટ્રીક

“LOFI” (Loop Outside, Finish Inside)

પ્રશ્ન 3(b) અથવા [4 ગુણ]

ફૂકશન હેન્ડલિંગમાં રિટર્ન સ્ટેટમેન્ટ સમજાવો.

જવાબ

પાસું	વર્ણન	ઉદાહરણ
હેતુ	કોલરને મૂલ્ય પાછું મોકલો	return result
માલિટપલ રિટર્ન	ટપલ તરીકે ઘણા મૂલ્યો પાછા મોકલો	return x, y, z
અર્લી એક્ઝિટ	અંત પહેલા ફૂકશનમાંથી બહાર નીકળો	if error: return None
નો રિટર્ન	ફૂકશન મૂળભૂત રીતે None પાછું મોકલે છે	def show(): print("Hi")

પાયથોન ફૂકશનોમાં return સ્ટેટમેન્ટ:

1. ફૂકશન એક્ઝિક્યુશન સમાપ્ત કરે છે
2. ફૂકશન કોલરને મૂલ્ય પાછું મોકલે છે
3. ઘણા મૂલ્યો (ટપલ તરીકે) પાછા મોકલી શકે છે
4. વૈકલ્પિક છે (જો છોડવામાં આવે, તો ફૂકશન None પાછું મોકલે છે)

ઉદાહરણ:

```

1 def calculate_circle(radius):
2     """
3         ...
4     if radius < 0:
5         return None #
6
6     area = 3.14 * radius ** 2
7     circumference = 2 * 3.14 * radius
8
9     return area, circumference #
10
11 #
12 result = calculate_circle(5)
13 print(f"           : {result}")

```

મેમરી ટ્રીક

“TERM” (Terminate Execution, Return Multiple values)

પ્રશ્ન 3(c) અથવા [7 ગુણ]

લૂપ કોન્સેપ્ટનો ઉપયોગ કરીને નીચેની પેટર્ન દર્શાવવા માટે એક પ્રોગ્રામ બનાવો

```

1 A
2 AB
3 ABC
4 ABCD
5 ABCDE

```

જવાબ

```

1 #
2 #      : A   E
3
4 #      (1   5)
5 for i in range(1, 6):
6     #           , 'A'

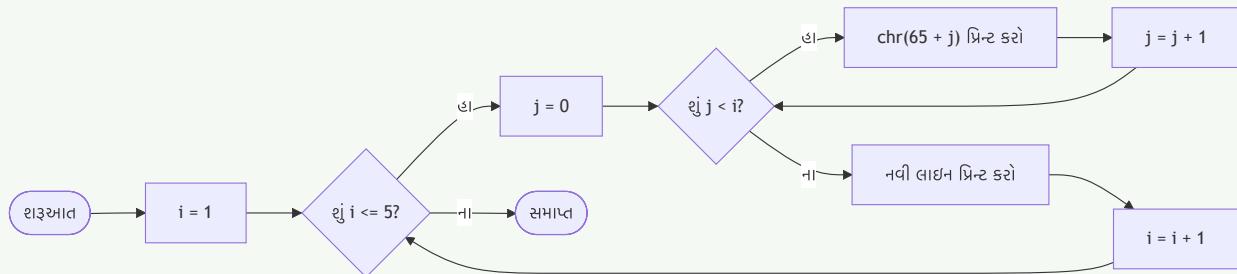
```

```

7   for j in range(i):
8       # 'A' ASCII 65 ,           j
9       print(chr(65 + j), end="")
10      #
11      print()

```

આફ્ટિનું:



સમજૂતીનું:

- બાહ્ય લૂપ: પંક્તિઓની સંખ્યા (1 થી 5) નિયંત્રિત કરે છે
- આંતરિક લૂપ: દરેક પંક્તિ માટે, 'A' થી શરૂ કરીને અક્ષરો પ્રિન્ટ કરે છે
- અક્ષર જનરેશન: ASCII મૂલ્ય રૂપાંતર (chr(65+j) 'A', 'B', વગેરે આપે છે)
- આઉટપુટ ફોર્મેટિંગ: દરેક પંક્તિ માટે end="" નો ઉપયોગ કરીને અક્ષરો એક જ લાઇનમાં પ્રિન્ટ કરવા

મેમરી ટ્રીક

“OICE” (Outer-Inner-Character-Endline)

પ્રશ્ન 4(a) [3 ગુણ]

નીચેના બિલ્ટ-ઈન ફંક્શનો યોગ્ય ઉદાહરણ સાથે વર્ણન કરો. i) max() ii) input() iii) pow()

જવાબ

ફંક્શન	હેતુ	સિન્કેસ	ઉદાહરણ
max()	ઇટરેબલમાં સૌથી મોટી વસ્તુ અથવા બે અથવા વધુ આંદ્રૂમેન્ટમાંથી સૌથી મોટી વસ્તુ પાછી મોકલે છે	max(iterable) અથવા max(arg1, arg2, ...)	max([1, 5, 3]) 5 પાછું મોકલે છે
input()	ઇનપુટમાંથી એક લાઈન વાંચે છે અને સિંગ્ટ્રે તરીકે પાછી મોકલે છે	input([prompt])	input(" " : ")
pow()	x ને y ની ઘાત પર ઉઠાવેલું પાછું મોકલે છે	pow(x, y)	pow(2, 3) 8 પાછું મોકલે છે

કોડમાં ઉદાહરણો:

```

1 # max()
2 numbers = [10, 5, 20, 15]
3 maximum = max(numbers)
4 print(f"      : {maximum}") #      : 20
5
6 # input()
7 name = input("      : ")
8 print(f"      , {name}!")
9
10 # pow()
11 result = pow(2, 4)
12 print(f"2  4      : {result}") #      : 2  4      : 16

```

પ્રશ્ન 4(b) [4 ગુણ]

યોગ્ય ઉદાહરણ આપીને સ્ટ્રિંગના સ્લાઇસિંગને સમજાવો.

જવાબ

પાયથોનમાં સ્ટ્રિંગ સ્લાઇસિંગનો ઉપયોગ સ્ટ્રિંગમાંથી સબસ્ટ્રિંગ બહાર કાઢવા માટે થાય છે.

સિન્ક્રેટન્સ: string[start:end:step]

પેરામીટર	વર્ણન	ડિફોલ્ટ	ઉદાહરણ
start	પ્રારંભિક ઇન્ડેક્સ (સમાવેશીત)	0	"Python"[1:] → "ython"
end	અંતિમ ઇન્ડેક્સ (અસમાવેશીત)	સ્ટ્રિંગની લંબાઈ	"Python"[:3] → "Pyt"
step	અક્ષરો વચ્ચે વધારો	1	"Python"[:, :2] → "Pto"

ઉદાહરણો:

```

1 text = "Python Programming"
2
3 #
4 print(text[0:6])      #    : "Python"
5 print(text[7:])        #    : "Programming"
6 print(text[:6])        #    : "Python"
7
8 #
9 print(text[::-2])      #    : "Pto rgamn"
0 print(text[0:10:-2])   #    : "Pto r"
1
2 #           ( )
3 print(text[-11:])     #    : "Programming"
4 print(text[:-12])     #    : "Python"
5
6 #
7 print(text[::-1])      #    : "gnimmargorP nohtyP"

```

મેમરી ટ્રીક

પ્રશ્ન 4(c) [7 ગુણ]

1 થી 7 ની વચ્ચેની તમામ વિષમ સંખ્યાઓના ક્યુબને પ્રિન્ટ કરતું ચુંગ ડિફાઇન ફંક્શન બનાવો.

જવાબ

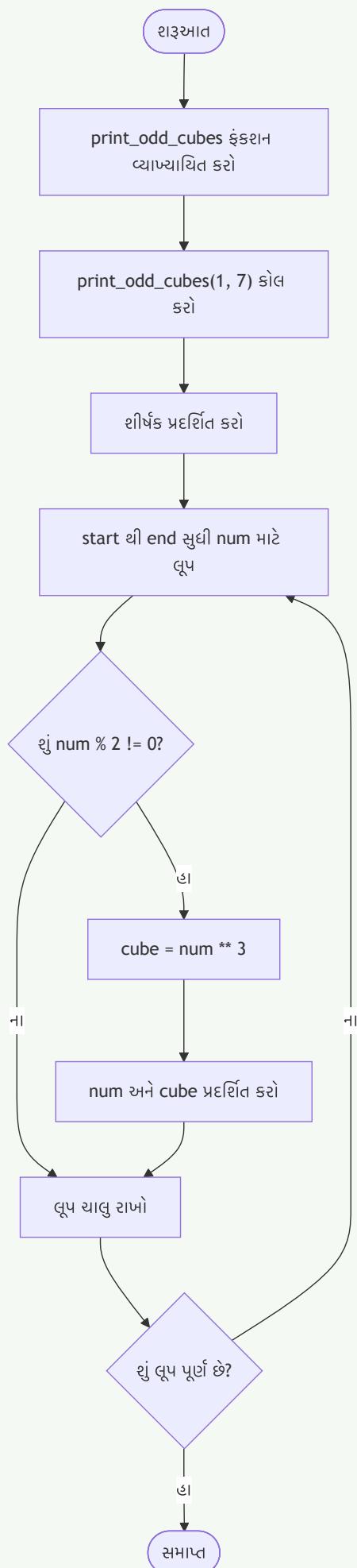
```

1 #
2 def print_odd_cubes(start, end):
3     """
4         ()
5     """
6     print(f"{start} {end} :")
7
8     #
9     for num in range(start, end + 1):
10         #
11         if num % 2 != 0:
12             #
13             cube = num ** 3

```

```
4     print(f"{num}      {cube}  ")
5
6 # 1    7
7 print_odd_cubes(1, 7)
```

આકૃતિ:



સમજૂતી:

- ફંક્શન વ્યાખ્યા: શ્રેણીમાં વિષમ સંખ્યાઓને પ્રોસેસ કરવા માટે ફંક્શન બનાવો
- લૂપ: શરૂઆતથી અંત સુધીના નંબરો પર પુનરાવર્તન કરો
- શરત: મોડ્યુલો ઓપરેટરનો ઉપયોગ કરીને તપારો કે નંબર વિષમ છે કે નહીં
- પ્રોસેસિંગ: વિષમ સંખ્યાઓના ક્યુબની ગણતરી કરો
- આઉટપુટ: દરેક વિષમ સંખ્યા અને તેનો ક્યુબ પ્રદાન કરો

મેમરી ટ્રીક

“FLOOP” (Function-Loop-Odd-Output-Power)

પ્રશ્ન 4(a) અથવા [3 ગુણ]

વિવિધ ફંક્શનો સાથે random મોડ્યુલ સમજવો.

જવાબ

પાયથોનમાં random મોડ્યુલ રેન્ડમ નંબર જનરેટ કરવા અને રેન્ડમ પસંદગીઓ કરવા માટે ફંક્શનો પ્રદાન કરે છે.

ફંક્શન	વર્ણન	ઉદાહરણ	પરિણામ
random()	0 અને 1 વચ્ચે રેન્ડમ ફલોટ પાછું મોકલે છે	random.random()	0.7134346335849448
randint(a, b)	a અને b (સમાવેશીત) વચ્ચે રેન્ડમ પૂણીક પાછું મોકલે છે	random.randint(1, 10)	7
choice(seq)	સિક્વન્સમાંથી રેન્ડમ તત્વ પાછું મોકલે છે	random.choice(['red', 'green', 'blue'])	'green'
shuffle(seq)	સિક્વન્સને ઇન-પ્લેસ શફ્ટ કરે છે	random.shuffle(my_list)	કોઈ રિટર્ન મૂલ્ય નહીં
sample(seq, k)	સિક્વન્સમાંથી k અન્ય રેન્ડમ તત્વો પાછા મોકલે છે	random.sample(range(1, 30), 5)	[3, 12, 21, 7, 25]

ઉદાહરણ:

```

1 import random
2
3 # 0 1
4 print(random.random())
5
6 # 1 10
7 print(random.randint(1, 10))
8
9 #
10 colors = ["red", "green", "blue", "yellow"]
11 print(random.choice(colors))
12
13 #
14 random.shuffle(colors)
15 print(colors)
16
17 #
18 print(random.sample(colors, 2))

```

મેમરી ટ્રીક

“RICES” (Random-Integer-Choice-Elements-Shuffle)

પ્રશ્ન 4(b) અથવા [4 ગુણ]

નીચેના લિસ્ટ ફંક્શનોની ચર્ચા કરો. i. len() ii. sum() iii. sort() iv. index()

જવાબ

ફંક્શન	હેતુ	સિન્ટેક્સ	ઉદાહરણ	આઉટપુટ
len()	લિસ્ટમાં આઇટમોની સંખ્યા પાછો મોકલે છે	len(list)	len([1, 2, 3])	3
sum()	લિસ્ટની બધી આઇટમોનો સરવાળો પાછો મોકલે છે	sum(list)	sum([1, 2, 3])	6
sort()	લિસ્ટને ઇન-પ્લેસ સોર્ટ કરે છે	list.sort()	[3, 1, 2].sort()	None (મૂળને સંશોધિત કરે છે)
index()	પ્રથમ ઘટનાનો ઇન્ડેક્સ પાછો મોકલે છે	list.index(value)	[10, 20, 30].index(20)	1

ઉદાહરણો:

```

1 # len()
2 numbers = [5, 10, 15, 20, 25]
3 print(f"      : {len(numbers)}") #      : 5
4
5 # sum()
6 print(f"      : {sum(numbers)}") #      : 75
7
8 # sort()
9 mixed = [3, 1, 4, 2]
10 mixed.sort() # -
11 print(f"      : {mixed}") #      : [1, 2, 3, 4]
12 mixed.sort(reverse=True)
13 print(f"      : {mixed}") #      : [4, 3, 2, 1]
14
15 # index()
16 fruits = ["apple", "banana", "cherry", "apple"]
17 print(f"'banana'      : {fruits.index('banana')}") #      : 1

```

મેમરી ટ્રીક

“LSSI” (Length-Sum-Sort-Index)

પ્રશ્ન 4(c) અથવા [7 ગુણ]

0 થી N સંખ્યાઓની ફિબોનાક્કી શ્રેણીને પ્રિન્ટ કરવા માટે યુઝર-ડિફાઇન ફંક્શન બનાવો. (જ્યાં N એક પૂર્ણાંક સંખ્યા છે અને આગ્રહીત તરીકે પસાર થાય છે)

જવાબ

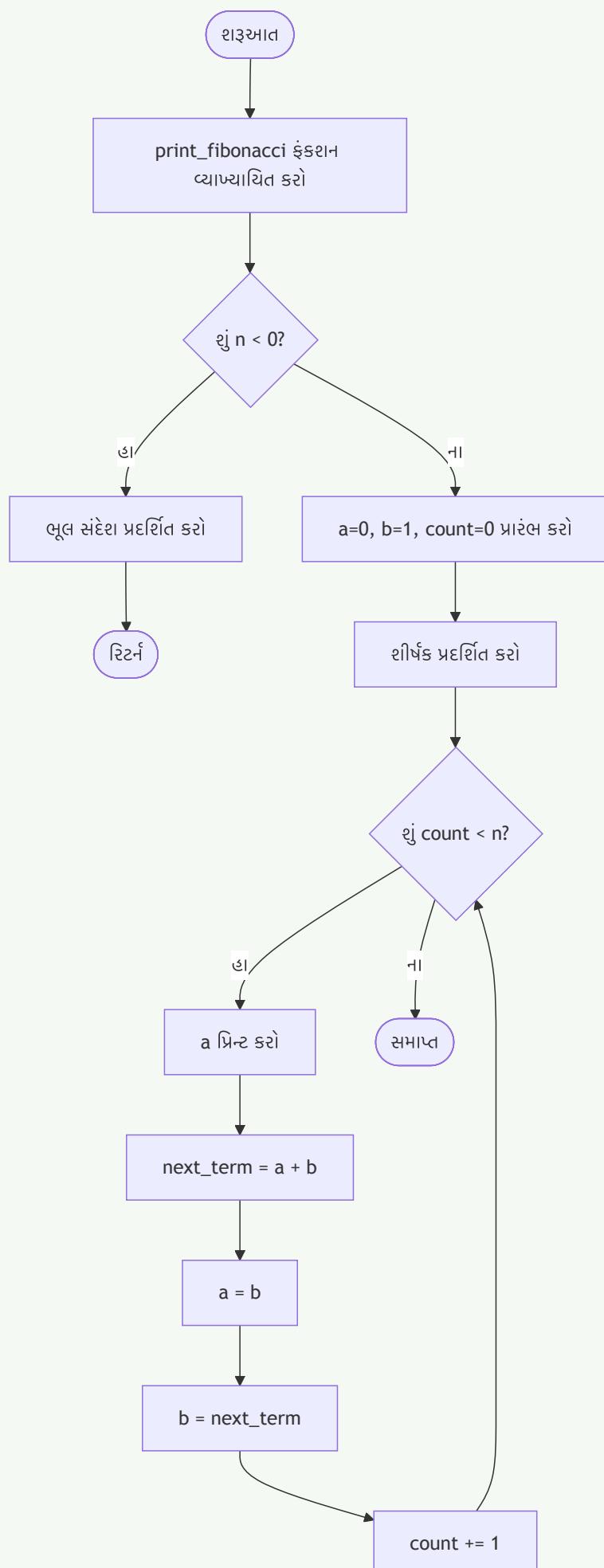
```

1 # N
2 def print_fibonacci(n):
3     """
4     n
5         0    0    1    1
6     """
7     #
8     if n < 0:
9         print("          ")
10        return
11
12     #
13     a, b = 0, 1
14     count = 0
15
16     print(f"{n}          ")
17
18     #

```

```
9  while count < n:
10     print(a, end=" ")
11     #
12     next_term = a + b
13     a = b
14     b = next_term
15     count += 1
```

આફ્રિતિ:



સમજૂતી:

- ઇનપુટ વેલિડેશન: તપાસો કે N એક માન્ય સકારાત્મક પૂર્ણાંક છે
- ચલો પ્રારંભ કરો: પ્રથમ બે ફિબોનાક્કી પદો સેટ કરો
- શ્રેણી પ્રિન્ટ કરો: ફિબોનાક્કી નંબરોને પ્રિન્ટ કરવા માટે લૂપ
- પદો અપડેટ કરો: આગળના પદની ગણતરી કરો અને આગળના પુનરાવર્તન માટે મૂલ્યો શિફ્ટ કરો
- સમાપ્તિ: જ્યારે કાઉન્ટ N સુધી પહોંચે ત્યારે અટકો

મેમરી ટ્રીક

“FIST” (Fibonacci-Initialize-Shift-Terminate)

પ્રશ્ન 5(a) [3 ગુણ]

આપેલ સ્ટ્રિંગ મેથડ્સ સમજાવો: i. count() ii. upper() iii. replace()

જવાબ

મેથડ	હેતુ	સિન્ટેક્સ	ઉદાહરણ	આઉટપુટ
count()	સબસ્ટ્રિંગની ઘટનાઓની ગણતરી કરે છે	str.count(substring)	"hello".count("l")	2
upper()	સ્ટ્રિંગને અપરકેસમાં રૂપાંતરિત કરે છે	str.upper()	"hello".upper()	"HELLO"
replace()	સબસ્ટ્રિંગની બધી ઘટનાઓને બદલે છે	str.replace(old, new)	"hello".replace("l", "r")	"herro"

ઉદાહરણો:

```

1 text = "Python programming is fun and Python is easy to learn"
2
3 # count()
4 print(f"'Python'      : {text.count('Python')}") #      : 2
5 print(f"'is'        : {text.count('is')}") #      : 2
6
7 # upper()
8 print(f"  : {text.upper()}") #  : PYTHON PROGRAMMING IS FUN AND PYTHON IS EASY TO LEARN"
9
0 # replace()
1 print(f"'Python'  'Java'      : {text.replace('Python', 'Java')}")
2 #      : "Java programming is fun and Java is easy to learn"

```

મેમરી ટ્રીક

“CUR” (Count-Uppper-Replace)

પ્રશ્ન 5(b) [4 ગુણ]

ટપલ ઓપરેશન ઉદાહરણ સાથે સમજાવો.

જવાબ

પાયથોનમાં ટપલ એ કમમાં રહેલા, અપરિવર્તનીય સંગહો છે જે કૌંસમાં બંધ થાય છે.

ઓપરેશન	વર્ણન	ઉદાહરણ	પરિણામ
સર્જન	મૂલ્યો સાથે ટપલ વ્યાખ્યાયિત કરો	t = (1, 2, 3)	3 આઈટમો સાથે ટપલ
ઇન્ડેક્સંગ	સ્થિતિ દ્વારા આઈટમને એક્સેસ કરો	t[0]	1

સ્લાઇસિંગ	ટપલનો ભાગ બહાર કાઢો	$t[1:3]$	(2, 3)
કેટેશન	બે ટપલસને જોડો	$t1 + t2$	સંયુક્ત ટપલ
રિપિટિશન	ટપલ તત્વોને પુનરાવર્તિત કરો	$t * 2$	કુલ્પિકેટ તત્વો

ઉદાહરણો:

```

1 #
2 fruits = ("apple", "banana", "cherry")
3 print(f"      : {fruits}")
4
5 #
6 print(f"      : {fruits[0]}") #      : "apple"
7 print(f"      : {fruits[-1]})" #      : "cherry"
8
9 #
10 print(f"      : {fruits[:2]}") #      : ("apple", "banana")
11
12 #
13 more_fruits = ("orange", "kiwi")
14 all_fruits = fruits + more_fruits
15 print(f"      : {all_fruits}") #      : ("apple", "banana", "cherry", "orange", "kiwi")
16
17 #
18 duplicated = fruits * 2
19 print(f"      : {duplicated}") #      : ("apple", "banana", "cherry", "apple", "banana", "cherry")
20
21 #
22 print(f"      : {len(fruits)}") #      : 3
23 print(f"      : {max(fruits)}") #      : "cherry"      ( )
24 print(f"      : {min(fruits)})" #      : "apple"      ( )

```

મેમરી ટ્રીક

“ICSM” (Immutable-Create-Slice-Merge)

પ્રશ્ન 5(c) [7 ગુણ]

બે સેટ બનાવવા અને આ બનાવેલા સેટ સાથે આપેલ ઓપરેશન કરવા માટે કોડ વિકસાવો: i) સેટ પર યુનિયન ઓપરેશન ii) સેટ પર ઇન્ટરસેક્શન ઓપરેશન
iii) સેટ પર ડિફરન્સ ઓપરેશન iv) બે સેટનો સિમેટ્રિક ડિફરન્સ

જવાબ

```

1 #
2
3 #
4 set_A = {1, 2, 3, 4, 5}
5 set_B = {4, 5, 6, 7, 8}
6
7 print(f" A: {set_A}")
8 print(f" B: {set_B}")
9
10 # i)      (A \cup B)
11 # A      B
12 union_result = set_A.union(set_B) #      set_A | set_B
13 print(f"\n{i} A      B      (A \cup B): {union_result}")
14
15 # ii)      (A \cap B)
16 # A      B
17 intersection_result = set_A.intersection(set_B) #      set_A & set_B
18 print(f"ii) A      B      (A \cap B): {intersection_result}")
19
20 # iii)      (A - B)
21 # A      B

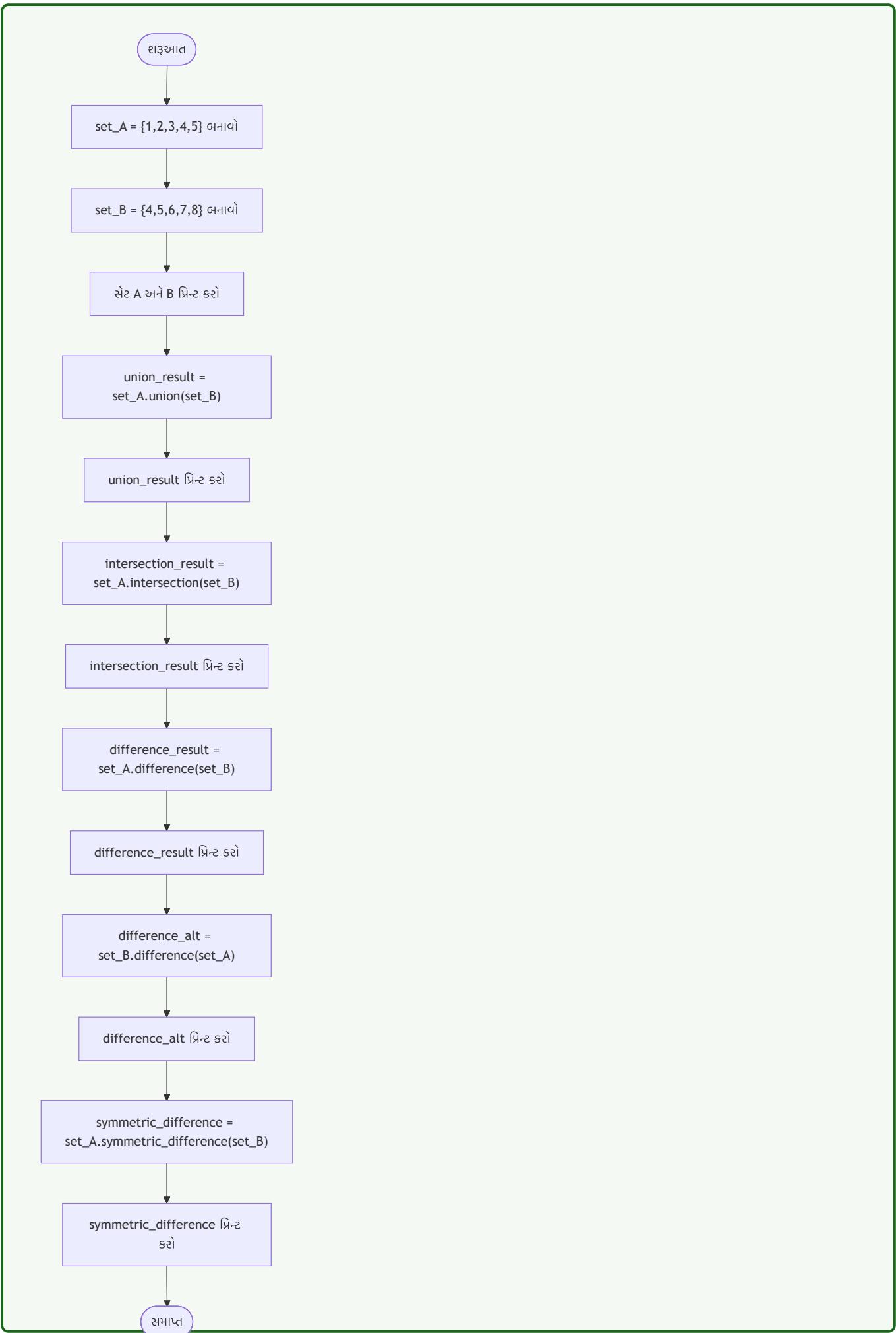
```

```

22 difference_result = set_A.difference(set_B) #      set_A - set_B
23 print(f"iii)      (A - B): {difference_result}")
24
25 #      (B - A)
26 difference_alt = set_B.difference(set_A) #      set_B - set_A
27 print(f"      (B - A): {difference_alt}")
28
29 # iv)      (A   B)
30 # A   B
31 symmetric_difference = set_A.symmetric_difference(set_B) #      set_A ^ set_B
32 print(f"iv)      (A   B): {symmetric_difference}")

```

આકૃતિ:



સમજૂતી:

- યુનિયન: ડુલિકેટ વિના બંને સેટના બધા તત્વો (1, 2, 3, 4, 5, 6, 7, 8)
- ઇન્ટરસેક્શન: બંને સેટમાં સામાન્ય તત્વો (4, 5)
- ડિફરન્સ (A-B): A માં પરંતુ B માં નહીં એવા તત્વો (1, 2, 3)
- ડિફરન્સ (B-A): B માં પરંતુ A માં નહીં એવા તત્વો (6, 7, 8)
- સિમેટ્રિક ડિફરન્સ: A અથવા B માં પરંતુ બંનેમાં નહીં એવા તત્વો (1, 2, 3, 6, 7, 8)

મેમરી ટ્રીક

“UIDS” (Union-Intersection-Difference-Symmetric)

પ્રશ્ન 5(a) અથવા [૩ ગુણ]

લિસ્ટને વ્યાખ્યાયિત કરો અને તે પાયથોનમાં કેવી રીતે બનાવવામાં આવે છે?

જવાબ

પાયથોનમાં લિસ્ટ એ કમબદ્ધ, પરિવર્તનશીલ વસ્તુઓનો સંગ્રહ છે જે વિવિધ ડેટા પ્રકારોના હોઈ શકે છે, જે ચોરસ કૌંસમાં બંધ હોય છે.
લિસ્ટ સર્જન પદ્ધતિઓની સારણી:

પદ્ધતિ	વર્ણન	ઉદાહરણ
લિટરલ	ચોરસ કૌંસનો ઉપયોગ કરીને બનાવો	my_list = [1, 2, 3]
કન્સ્ટ્રક્ટર	list() ફંક્શનનો ઉપયોગ કરીને બનાવો	my_list = list((1, 2, 3))
કોમ્પ્લિક્ષન્શન	એક લાઇન એક્સપ્રેશનનો ઉપયોગ કરીને બનાવો	my_list = [x for x in range(5)]
ઇટરેબલથી	અન્ય ઇટરેબલ્સને લિસ્ટમાં રૂપાંતરિત કરો	my_list = list("abc")
ખાલી લિસ્ટ	ખાલી લિસ્ટ બનાવો અને પછીથી ઉમેરો	my_list = []

ઉદાહરણો:

```

1  #
2  numbers = [1, 2, 3, 4, 5]
3  mixed = [1, "hello", 3.14, True]
4
5  # list()
6  tuple_to_list = list((10, 20, 30))
7  string_to_list = list("Python")
8
9  #
10 squares = [x**2 for x in range(1, 6)]
11
12 #
13 empty_list = []
14 empty_list.append("first")
15 empty_list.append("second")
16
17 print(f" : {numbers}")
18 print(f" : {mixed}")
19 print(f" : {tuple_to_list}")
20 print(f" : {string_to_list}")
21 print(f" : {squares}")
22 print(f" : {empty_list}")

```

મેમરી ટ્રીક

“LCMIE” (Literal-Constructor-Mixed-Iterable-Empty)

પ્રશ્ન 5(b) અથવા [4 ગુણ]

ડિક્શનરી બિલ્ટ-ઇન ફંક્શન અને મેથ્ડ્સ સમજાવો.

જવાબ

ડિક્શનરી એ કર્લી પ્રેસિઝ માં બંધ કી-વેલ્યુ જોડીઓનો સંગ્રહ છે.

ફંક્શન/મેથ્ડ	વર્ણન	ઉદાહરણ	પરિણામ
dict()	ડિક્શનરી બનાવે છે	dict(name='John', age=25)	\{'name': 'John', 'age': 25\}
len()	આઈટમોની સંખ્યા પાછી મોકલે છે	len(my_dict)	પૂર્ણાંક ગણતરી
keys()	બધી કીનું વ્યૂ પાછું મોકલે છે	my_dict.keys()	ડિક્શનરી વ્યૂ ઓફ્જેક્ટ
values()	બધા મૂલ્યોનું વ્યૂ પાછું મોકલે છે	my_dict.values()	ડિક્શનરી વ્યૂ ઓફ્જેક્ટ
items()	(કી, મૂલ્ય) જોડીઓનું વ્યૂ પાછું મોકલે છે	my_dict.items()	ડિક્શનરી વ્યૂ ઓફ્જેક્ટ
get()	કી માટે મૂલ્ય, અથવા ડિફોલ્ટ પાછું મોકલે છે	my_dict.get('key', 'default')	મૂલ્ય અથવા ડિફોલ્ટ
update()	બીજા ડિક્શનરીથી કી/મૂલ્યો સાથે ડિક્શનરી અપડેટ કરે છે	my_dict.update(other_dict)	None (ઇન-પ્લેસ અપડેટ કરે છે)
pop()	કી સાથેની આઇટમ દૂર કરે છે અને મૂલ્ય પાછું મોકલે છે	my_dict.pop('key')	દૂર કરેલી આઇટમનું મૂલ્ય

ઉદાહરણ:

```

1 # 
2 student = {
3     'name': 'John',
4     'age': 20,
5     'courses': ['Math', 'Science']
6 }
7
8 # - 
9 print(f" : {len(student)}") # : 3
10 #
11 print(f" : {student.keys()}")
12 print(f" : {student.values()}")
13 print(f" : {student.items()}")
14
15 #      get
16 print(f" ( ) : {student.get('grade', 'N/A')}")
17
18 #
19 student.update({'grade': 'A', 'age': 21})
20 print(f" : {student}")
21
22 #
23 removed_item = student.pop('age')
24 print(f" : {removed_item}")
25 print(f" : {student}")

```

મેમરી ટ્રીક

“LKVIGUP” (Length-Keys-Values-Items-Get-Update-Pop)

પ્રશ્ન 5(c) અથવા [7 ગુણ]

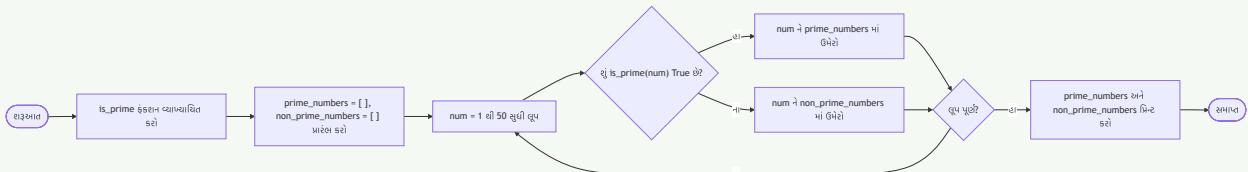
1 થી 50 શ્રેણીમાં અવિભાજ્ય અને સંચુક્ત સંખ્યાઓની સૂચિ બનાવવા માટે પાયથોન કોડ વિકસાવો.

```

1 # 1    50
2
3 def is_prime(num):
4     """
5         True,      False
6     """
7
8     # 1
9     if num <= 1:
10        return False
11
12     # 2
13     if num == 2:
14        return True
15
16     # 2
17     if num % 2 == 0:
18        return False
19
20     # num
21     (:      sqrt(num)           )
22     for i in range(3, int(num**0.5) + 1, 2):
23
24     if num %
25
26         i == 0:
27
28             return False
29
30     return True
31
32 #
33 prime_numbers = []
34 non_prime_numbers = []
35
36 # 1    50
37 for num in range(1, 51):
38     if is_prime(num):
39         prime_numbers.append(num)
40     else:
41         non_prime_numbers.append(num)
42
43 #
44 print(f"1    50      : {prime_numbers}")
45 print(f"1    50      : {non_prime_numbers}")

```

આફ્ટિન:



સમજૂતી:

- હેલ્પર ફંક્શન: `is_prime()` કાર્યક્રમ રીતે તપાસે છે કે સંખ્યા અવિભાજ્ય છે કે નહીં
- ઓફિચિયલ વર્ગમૂળ સુધી જ વિભાજ્યતા તપાસે છે
- વર્ગીકરણ: સંખ્યાઓને અવિભાજ્ય અથવા સંયુક્ત સૂચિમાં વર્ગીકૃત કરે છે
- આઉટપુટ: અંતે બંને સૂચિઓ પ્રદર્શિત કરે છે

અવિભાજ્ય સંખ્યાઓ (1 થી 50): 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47
સંયુક્ત સંખ્યાઓ (1 થી 50): 1, 4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 25, 26, 27, 28, 30, 32, 33, 34, 35, 36, 38, 39, 40, 42, 44, 45, 46, 48, 49, 50

