

Python Programming (4311601) - Summer 2023 Solution (Gujarati)

Milav Dabgar

August 09, 2023

પ્રશ્ન 1(a) [3 ગુણ]

પ્રોબ્લેમ સોલ્વિંગમાં સામેલ પગલાં સમજાવો.

જવાબ

કોષ્ટક 1. પ્રોબ્લેમ સોલ્વિંગના પગલાં

પગલું	વર્ણન
સમસ્યા સમજવી	સમસ્યાને સ્પષ્ટ રીતે વાંચો અને સમજો
વિશ્લેષણ	સમસ્યાને નાના ભાગોમાં વિભાજિત કરો
અલ્ગોરિધમ ડિઝાઇન	પગલાંવાર ઉકેલનો અભિગમ બનાવો
અમલીકરણ	પ્રોગ્રામિંગ લેન્વેજનો ઉપયોગ કરીને કોડ કરો
ટેસ્ટિંગ	વિવિધ ટેસ્ટ કેસ સાથે સોલ્યુશન ચકાસો
ડૉક્યુમેન્ટેશન	ભવિષ્યના ઉપયોગ માટે સોલ્યુશન દસ્તાવેજુકરણ કરો

મુખ્ય મુદ્દાઓ:

- સમસ્યા વ્યાખ્યા: શું હલ કરવાની જરૂર છે તે સ્પષ્ટ રીતે ઓળખો
- ઇનપુટ/આઉટપુટ: જરૂરી ઇનપુટ અને અપેક્ષિત આઉટપુટ નક્કી કરો
- લોઝિક બિલ્ડિંગ: સોલ્યુશનનો તાર્કિક પ્રવાહ બનાવો

મેરી ટ્રીક

“લોકો હંમેશા ડિઝાઇન કરીને અમલીકરણ ટેસ્ટ કરે છે દરરોજ”

પ્રશ્ન 1(b) [4 ગુણ]

Python ના ફીચર્સ લખો.

જવાબ

કોષ્ટક 2. Python ફીચર્સ

ફીચર	વર્ણન
સરળ સિન્કેસ	કોડ વાંચવામાં અને લખવામાં સરળ
ઇન્ટરપ્રિટેડ	કોમ્પાઇલેશનની જરૂર નથી, સીધું ચાલે છે
પ્લેટફોર્મ ઇન્ડિપેન્ટ	Windows, Mac, Linux પર ચાલે છે
ઓફ્જેક્ટ-ઓરિએન્ટેડ	કલાસ અને ઓફ્જેક્ટને સપોર્ટ કરે છે
મોટી લાઇબ્રેરી	વ્યાપક બિલ્ટ-ઇન મોડ્યુલ્સ
ડાયનામિક ટાઇપિંગ	વેરિએબલ ટાઇપ ડિક્લેર કરવાની જરૂર નથી

મુખ્ય ફીચર્સ:

- ફી અને ઓપન સોર્સ: દરેક માટે ઉપયોગ કરવા માટે ઉપલબ્ધ
- હાઇ-લેવલ લેંગ્વેજ: માનવ ભાષાની નજીક
- વ્યાપક સપોર્ટ: મોટો કમ્પ્યુનિટી અને ડોક્યુમેન્ટેશન

મેમરી ટ્રીક

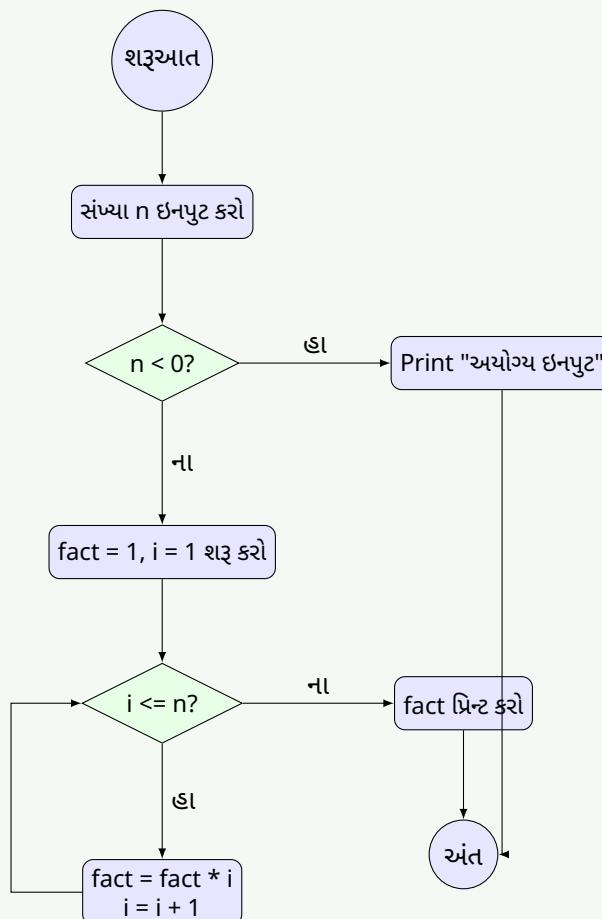
"સરળ ઇન્ટરપ્રિટેડ પ્લેટફોર્મ-ઇન્ડિપેન્ટ ઓફ્જેક્ટ-ઓરિએન્ટેડ લાઇબ્રેરીઝ ડાયનામિક"

પ્રશ્ન 1(ચ) [૭ ગુણ]

આપેલી સંખ્યાનો ફેક્ટોરિયલ શોધવા માટેનો ફ્લોચાર્ટ દોરો તેમજ અંગોરિધમ લખો.

જવાબ

ફ્લોચાર્ટ:



આકૃતિ 1. ફેક્ટોરિયલ માટે ફ્લોચાર્ટ

અભ્યાસિધમ:

1. શરૂઆત
2. સંપૂર્ણ n ઇનપુટ કરો
3. જો $n < 0$, તો "અયોગ્ય ઇનપુટ" પ્રિન્ટ કરો અને પગલું 8 પર જાઓ
4. $\text{fact} = 1$, $i = 1$ શરૂ કરો
5. જ્યાં સુધી $i \leq n$, કરો:
 - $\text{fact} = \text{fact} * i$
 - $i = i + 1$
6. fact પ્રિન્ટ કરો
7. અંત

મુખ્ય મુદ્દાઓ:

- બેઝ કેસ: $0! = 1$ અને $1! = 1$
- વેલિડેશન: નેગેટિવ નંબર માટે ચેક કરો
- લૂપ લોજિક: 1 થી n સુધીના બધા નંબર ગુણો

મેમરી ટ્રીક

"ઇનપુટ વેલિડેટ ઇનિશિયલાઇઝ લૂપ પ્રિન્ટ"

પ્રશ્ન 1(c OR) [7 ગુણ]

ઉદાહરણ સાથે રિલેશનલ અને એસાઇનમેન્ટ ઓપરેટરો સમજાવો.

જવાબ

કોષ્ટક 3. રિલેશનલ ઓપરેટર્સ

ઓપરેટર	વર્ણન	ઉદાહરણ
$==$	બરાબર	$5 == 5$ (True)
$!=$	બરાબર નથી	$5 != 3$ (True)
$>$	મોટું	$7 > 3$ (True)
$<$	નાનું	$2 < 8$ (True)
\geq	મોટું અથવા બરાબર	$5 \geq 5$ (True)
\leq	નાનું અથવા બરાબર	$4 \leq 6$ (True)

કોષ્ટક 4. એસાઇનમેન્ટ ઓપરેટર્સ

ઓપરેટર	વર્ણન	ઉદાહરણ
$=$	સાંદું એસાઇનમેન્ટ	$x = 5$
$+=$	ઉમેરીને એસાઇન કરો	$x += 3$ ($x = x + 3$)
$-=$	બાદ કરીને એસાઇન કરો	$x -= 2$ ($x = x - 2$)
$*=$	ગુણીને એસાઇન કરો	$x *= 4$ ($x = x * 4$)
$/=$	ભાગીને એસાઇન કરો	$x /= 2$ ($x = x / 2$)

Listing 1. ઓપરેટર્સ ઉદાહરણ

```

1 # રલિશનલ ઓપરેટર્સ
2 a, b = 10, 5
3 print(a > b) # True
4 print(a == b) # False
5

```

```

6 # એસાઇનમેન્ટ ઓપરેટર્સ
7 x = 10
8 x += 5 # x બને છે 15
9 x *= 2 # x બને છે 30

```

મેમરી ટ્રીક

“સંબંધ તુલના કરો, મૂલ્યો એસાઇન કરો”

પ્રશ્ન 2(a) [3 ગુણ]

ફ્લોચાર્ટ માટે ઉપયોગમાં લેવાતા વિવિધ પ્રતીકો દોરો અને દરેક પ્રતીકનો હેતુ લખો.

જવાબ

કોષ્ટક 5. ફ્લોચાર્ટ સિમ્બોલ્સ

સિમ્બોલ	નામ	હેતુ
શરૂઆત	અંડાકાર	પ્રોગ્રામની શરૂઆત/અંત
પ્રોસેસ	લંબચોરસ	પ્રોસેસિંગ ઓપરેશન્સ
?	હીરા	શરતી સ્ટેટમેન્ટ્સ
I/O	સમાંતરચયતુજ્કોણ	ડેટા ઇનપુટ/આઉટપુટ
અંગુઠી	વર્તુળ	વિવિધ ભાગોને જોડવા
→	તીર	પ્રવાહની દિશા

મુખ્ય મુદ્દાઓ:

- સ્ટાર્ટ સિમ્બોલ્સ: સાર્વનિક રીતે માન્ય આકારો
- સ્પષ્ટ ફ્લો: તીરો પ્રોગ્રામની દિશા દર્શાવે છે
- તાર્કિક માળખું: પ્રોગ્રામ લોજિકને વિગ્રહિત કરવામાં મદદ કરે છે

મેમરી ટ્રીક

“ટર્મિનલ્સ પ્રોસેસ ડિસિજન્સ ઇનપુટ કનેક્ટર્સ ફ્લો”

પ્રશ્ન 2(b) [4 ગુણ]

સારા અલ્ગોરિધમની લાક્ષણિકતાઓ સૂચિબદ્ધ કરો.

જવાબ

કોષ્ટક 6. સારા અલ્ગોરિધમની લાક્ષણિકતાઓ

વાક્ષણિકતા	વર્ણન
મર્યાદિત	મર્યાદિત પગલાં પછી સમાપ્ત થવું જોઈએ
નિશ્ચિત	દરેક પગલું સ્પષ્ટ રીતે વ્યાખ્યાપિત
ઇનપુટ	શૂન્ય અથવા વધુ ઇનપુટ્સ સ્પષ્ટ
આઉટપુટ	ઓછામાં ઓછું એક આઉટપુટ
અસરકારક	પગલાં સરળ અને શક્ય હોવા જોઈએ
અસ્પષ્ટ નહીં	દરેક પગલાંનો માત્ર એક જ અર્થ

મુખ્ય વાક્ષણિકતાઓ:

- શુદ્ધતા: બધા ચોગ્ય ઇનપુટ્સ માટે સાચા પરિણામો
- કાર્યક્ષમતા: ન્યૂનતમ સમય અને જગ્યાના સંસાધનોનો ઉપયોગ
- સ્પષ્ટતા: સમજવામાં અને અમલ કરવામાં સરળ

મેમરી ટ્રીક

“મર્યાદિત નિશ્ચિત ઇનપુટ આઉટપુટ અસરકારક અસ્પષ્ટ નહીં”

પ્રશ્ન 2(c) [7 ગુણ]

નીચેના ડેટા મૂલ્યોને રજૂ કરવા માટે ચોગ્ય ડેટા ટાઇપનો ઉપયોગ કરો.

જવાબ

કોષ્ટક 7. ડેટા ટાઇપ મેંઝિંગ

ડેટા મૂલ્ય	ડેટા ટાઇપ	ઉદાહરણ
(1) અઠવાડિયામાં દિવસોની સંખ્યા	int	days = 7
(2) ગુજરાતનો રહેવાસી છે કે નહીં	bool	is_resident = True
(3) મોબાઇલ નંબર	str	mobile = "9876543210"
(4) બેંક ખાતાનો બેલેન્સ	float	balance = 15000.50
(5) એક ગોળાનું ઘનક્ષફળ	float	volume = 523.33
(6) ચોરસનો પરિમિતિ	float	perimeter = 20.0
(7) વિદ્યાર્થીનું નામ	str	name = "રાહુલ"

Listing 2. ડેટા ટાઇપ ઉદાહરણો

```

1 # ડેટા ટાઇપ ઉદાહરણો
2 days = 7           # int
3 is_resident = True # bool
4 mobile = "9876543210" # str
5 balance = 15000.50   # float
6 volume = 523.33     # float
7 perimeter = 20.0    # float
8 name = "રાહુલ"      # str

```

મુખ્ય મુદ્દાઓ:

- int: દરારાંશ લિના પૂર્ણ સંખ્યાઓ
- float: દરારાંશ બિંદુ સાથેની સંખ્યાઓ
- str: કોટ્સમાં ટેક્સ્ટ ડેટા
- bool: માત્ર True/False મૂલ્યો

મેમરી ટ્રીક

“ઇન્ટિજર્સ ફ્લોટ સ્ટ્રેગ્સ બુલિયન્સ”

પ્રશ્ન 2(a OR) [3 ગુણ]

નીચેના કોડનું આઉટપુટ શોધો.

જવાબ

Listing 3. Code Snippet

```
1 num1 = 2+9*((3*12)-8)/10
2 print(num1)
```

પગલાવાર ગણતરી:

- પગલું 1: $3 \times 12 = 36$
- પગલું 2: $36 - 8 = 28$
- પગલું 3: $9 \times 28 = 252$
- પગલું 4: $252/10 = 25.2$
- પગલું 5: $2 + 25.2 = 27.2$

આઉટપુટ: 27.2

મુખ્ય મુદ્દાઓ:

- BODMAS નિયમ:** કૌંસ, ઓર્ડર્સ, ભાગાકાર, ગુણાકાર, સરવાળો, બાદબાકી
- ઓપરેટર પ્રિસ્િન્સ:** પહેલા કૌંસ, પછી ગુણાકાર/ભાગાકાર
- પરિણામ ટાઇપ:** ભાગાકાર ઓપરેશનને કારણે ફ્લોટ

મેમરી ટ્રીક

“કૌંસ ઓર્ડર્સ ભાગાકાર ગુણાકાર સરવાળો બાદબાકી”

પ્રશ્ન 2(b OR) [4 ગુણ]

Python માં ઉપયોગમાં લેવાતા વિવિધ પ્રકારના ઓપરેટર્સની સૂચિ બનાવો.

જવાબ

કોષ્ટક 8. Python ઓપરેટર્સ

પ્રકાર	ઓપરેટર્સ	ઉદાહરણ
અરિથમેટિક	$+, -, *, /, \%, **, //$	$5 + 3 = 8$
તુલના	$==, !=, >, <, >=, <=$	$5 > 3 = \text{True}$
લોજિકલ	and, or, not	True and False = False
એસાઇનમેન્ટ	$=, +=, -=, *=, /=$	$x += 5$
બિટવાઈઝ	$\&, , \sim, <<, >>$	$5 \& 3 = 1$
મેમ્બરશિપ	in, not in	'a' in 'cat' = True
આઇડેન્ટિટી	is, is not	x is y

મુખ્ય મુદ્દાઓ:

- અરિથમેટિક:** ગાણિતિક ઓપરેશન્સ
- તુલના:** મૂલ્યોની તુલના કરે છે અને બુલિયન પરત કરે છે
- લોજિકલ:** બુલિયન એક્સપ્રેશન-સને જોડે છે

મેમરી ટ્રીક

“અરિથમેટિક તુલના લોજિકલ એસાઇનમેન્ટ બિટવાઈજ મેમ્બરશિપ આઇડેન્ટીટી”

પ્રશ્ન 2(c OR) [7 ગુણ]

યુઝર દ્વારા દાખલ કરેલા બધા ધન સંખ્યાઓનો સરવાળો અને સરેરાશ શોધવા માટે પ્રોગ્રામ લખો. જ્યારે યુઝર કોઈ નેગેટિવ નંબરમાં એન્ટર કરે ત્યારે યુઝર પાસેથી આગળનું કોઈપણ ઇનપુટ લેવાનું બંધ કરો અને સરવાળો અને સરેરાશ પ્રદર્શિત કરો.

જવાબ

Listing 4. સરવાળો અને સરેરાશ પ્રોગ્રામ

```

1 # ધન સંખ્યાઓનો સરવાળો અને સરેરાશ શોધવાનો પ્રોગ્રામ
2 total_sum = 0
3 count = 0
4
5 print("ધન સંખ્યાઓ દાખલ કરો નેગેટિવ( રોકવા માટે):")
6
7 while True:
8     num = float(input("સંખ્યા દાખલ કરો: "))
9
10    if num < 0:
11        break
12
13    total_sum += num
14    count += 1
15
16    if count > 0:
17        average = total_sum / count
18        print("સરવાળો: {total_sum}")
19        print("સરેરાશ: {average}")
20    else:
21        print("કોઈ ધન સંખ્યાઓ દાખલ કરાયેલ નથી")

```

મુખ્ય મુદ્દાઓ:

- લૂપ કંટ્રોલ: break સ્ટેટમેન્ટ સાથે while લૂપ
- ઇનપુટ વેલિડેશન: નેગેટિવ નંબર્સ માટે ચેક કરો
- શૂન્ય દ્વારા ભાગાકાર: જ્યારે કોઈ નંબર દાખલ ન થયા હોય ત્યારે હેન્ડલ કરો

મેમરી ટ્રીક

“ઇનપુટ લૂપ ચેક કેલક્યુલેટ ડિસ્પ્લે”

પ્રશ્ન 3(a) [3 ગુણ]

ઉદાહરણ સાથે while લૂપ સમજાવો.

જવાબ

While લૂપ સ્ટ્રક્ચર:

```

1 while condition:
2     # statements
3     # update condition

```

ઉદાહરણ:

Listing 5. While લૂપ ઉદાહરણ

```

1 # 1 થી 5 સુધીના નંબરસ પ્રનિઃઠ કરો
2 i = 1
3 while i <= 5:
4     print(i)
5     i += 1

```

મુખ્ય મુદ્દાઓ:

- પ્રી-ટેસ્ટેડ લૂપ: એક્ઝિક્યુશન પહેલાં કંડિશન ચેક થાય છે
- અનંત લૂપ જોખમ: કંડિશન આખરે False થવી જોઈએ
- લૂપ વેરિએબલ: લૂપની અંદર અપડેટ થવું જોઈએ

મેમરી ટ્રીક

“જ્યારે કંડિશન સાચી હોય ત્યારે એક્ઝિક્યુટ કરો”

પ્રશ્ન 3(b) [4 ગુણ]

યુઝર દ્વારા ઇનપુટ કરેલ પૂર્ણાંક સંખ્યાના ડિજિટનો સરવાળો શોધવા માટે પ્રોગ્રામ લખો.

જવાબ

Listing 6. ડિજિટનો સરવાળો પ્રોગ્રામ

```

1 # ડિજિટનો સરવાળો શોધવાનો પ્રોગ્રામ
2 num = int(input("સંખ્યા દાખલ કરો: "))
3 original_num = num
4 digit_sum = 0
5
6 while num > 0:
7     digit = num % 10
8     digit_sum += digit
9     num = num // 10
10
11 print(f"{original_num} ના ડિજિટનો સરવાળો {digit_sum} છે")

```

મુખ્ય મુદ્દાઓ:

- મૌજુદુલો ઓપરેશન: %10 વાપરીને છેલ્લો ડિજિટ કાઢો
- ઇન્ટિઝર ડિવિઝન: //10 વાપરીને છેલ્લો ડિજિટ હટાવો
- શૂન્ય સુધી લૂપ: ડિજિટ્સ બાકી ન રહે ત્યાં સુધી રાખો

મેમરી ટ્રીક

“કાઢો ઉમેરો હટાવો પુનરાવર્તન કરો”

પ્રશ્ન 3(c) [7 ગુણ]

યુઝર-નિર્ધારિત ફુક્શનનો ઉપયોગ કરીને 100 થી 10000 ના વચ્ચેના આર્મ્સ્ટ્રોગ નંબરો છાપવા માટે પ્રોગ્રામ લખો.

જવાબ

Listing 7. આર્મસ્ટ્રોગ નંબર્સ પ્રોગ્રામ

```

1 def is_armstrong(num):
2     """નંબર આર્મસ્ટ્રોગ નંબર છે કે નહીં ચેક કરો"""
3     original = num
4     num_digits = len(str(num))
5     sum_powers = 0
6
7     while num > 0:
8         digit = num % 10
9         sum_powers += digit ** num_digits
10        num //= 10
11
12    return sum_powers == original
13
14 def print_armstrong_range(start, end):
15     """આપેલી રેન્જમાં આર્મસ્ટ્રોગ નંબર્સ પ્રણિત કરો"""
16     print(f"{start} અને {end} વયુંના આર્મસ્ટ્રોગ નંબર્સ:")
17
18     for num in range(start, end + 1):
19         if is_armstrong(num):
20             print(num, end=" ")
21     print()
22
23 # મુખ્ય પ્રોગ્રામ
24 print_armstrong_range(100, 10000)

```

મુખ્ય મુદ્દાઓ:

- ફંક્શન ડેફનિશન: def કીવર્ડ વાપરીને ફંક્શન-સ બનાવો
- આર્મસ્ટ્રોગ લોજિક: ડિજિટ્સનો સરવાળો ડિજિટ્સની સંખ્યાની પાવર સુધી
- રેન્જ ફંક્શન: સ્પષ્ટ રેન્જમાં નંબર્સ જનરેટ કરો

મેમરી ટ્રીક

“ડિફાઇન ચેક કેલક્યુલેટ કમ્પેર પ્રિન્ટ”

પ્રશ્ન 3(a OR) [3 ગુણ]

નીચેના પેટર્ન છાપવા માટે પ્રોગ્રામ લખો.

જવાબ

```

5 4 3 2 1
4 3 2 1
3 2 1
2 1
1

```

Listing 8. પેટર્ન પ્રિન્ટિંગ

```

1 # પેટર્ન પ્રણિતિ પ્રોગ્રામ
2 for i in range(5, 0, -1):
3     for j in range(i, 0, -1):
4         print(j, end=" ")
5     print()

```

મુખ્ય મુદ્દાઓ:

- નેસ્ટેડ લૂપ્સ: બાહ્ય લૂપ રો માટે, અંદરનું કોલમ માટે
- રિવર્સ રેન્જ: ઘટવા માટે range(start, stop, -1)
- પ્રિન્ટ કંટ્રોલ: સ્પેસ માટે end=" ", newline માટે print()

મેમરી ટ્રીક**“બાહ્ય અંદરનું રિવર્સ પ્રિન્ટ”****પ્રશ્ન 3(b OR) [4 ગુણ]****નેસ્ટેડ if...else સ્ટેમ્પેન્ટ સમજાવો.****જવાબ****સ્ક્રીપ્ટ:**

```

1  if condition1:
2      if condition2:
3          # statements
4      else:
5          # statements
6  else:
7      if condition3:
8          # statements
9      else:
10         # statements

```

ઉદાહરણ:**Listing 9. નેસ્ટેડ If-Else ઉદાહરણ**

```

marks = 85

1 if marks >= 50:
2     if marks >= 90:
3         grade = "A+"
4     elif marks >= 80:
5         grade = "A"
6     else:
7         grade = "B"
8 else:
9     grade = "F"

print(f"ગ્રેડ: {grade}")

```

મુખ્ય મુદ્દાઓ:

- અંદરની શરતો: બીજા if-else ની અંદર if-else
- અનેક સ્તરો: અનેક સ્તરો સુધી નેસ્ટ કરી શકાય છે
- લોજિકલ ફ્લોઝ: અંદરની શરતો ફક્ત ત્યારે જ એક્ઝિક્યુટ થાય છે જ્યારે બાહ્ય સાચી હોય

મેમરી ટ્રીક**“બાહ્ય અંદરનું અનેક સ્તરો”**

પ્રશ્ન 3(c OR) [7 ગુણ]

લિસ્ટમાં n નંબરો દાખલ કરવા તેમજ statistics મોડ્યુલનો ઉપયોગ કરીને તેમનો mean, median અને mode શોધવા માટેનો પ્રોગ્રામ લખો.

જવાબ

Listing 10. Statistics મોડ્યુલ પ્રોગ્રામ

```

1 import statistics
2
3 # એલમિનટ્સની સંખ્યા ઇનપુટ કરો
4 n = int(input("એલમિનટ્સની સંખ્યા દાખલ કરો: "))
5 numbers = []
6
7 # નંબર્સ ઇનપુટ કરો
8 for i in range(n):
9     num = float(input(f"નંબર {i+1} દાખલ કરો: "))
10    numbers.append(num)
11
12 # આંકડાશાસ્ત્ર ગણો
13 mean_val = statistics.mean(numbers)
14 median_val = statistics.median(numbers)
15
16 try:
17     mode_val = statistics.mode(numbers)
18 except statistics.StatisticsError:
19     mode_val = "કોઈ ચુનાકી mode નથી"
20
21 # પરણામો દરશાવો
22 print(f"નંબર્સ: {numbers}")
23 print(f"મીન: {mean_val}")
24 print(f"મેડિન: {median_val}")
25 print(f"મોડ: {mode_val}")

```

મુખ્ય મુદ્દાઓ:

- **Statistics મોડ્યુલ:** આંકડાકીય ફંક્શન્સ માટે બિલ્ટ-ઇન મોડ્યુલ
- **લિસ્ટ ઇનપુટ:** પ્રોસ્સિંગ માટે લિસ્ટમાં નંબર્સ સ્ટોર કરો
- **એક્સેપ્શન હેન્ડલિંગ:** mode કેલ્ક્યુલેશન એરર્સ હેન્ડલ કરો

મેમરી ટ્રીક

“ઇમ્પોર્ટ ઇનપુટ કેલ્ક્યુલેટ ડિસ્પ્લે”

પ્રશ્ન 4(a) [3 ગુણ]

Python માં for લૂપ અને while લૂપ વચ્ચે તફાવત લખો.

જવાબ

કોષ્ટક 9. For લૂપ vs While લૂપ

ફીચર	For લૂપ	While લૂપ
હેતુ	જાણીતા પુનરાવર્તનો	અજાણ્યા પુનરાવર્તનો
સિન્ટેક્સ	for var in sequence	while condition
ઇનિશિયલાઇઝેશન	ઓટોમેટિક	મેન્યુઅલ
અપડેટ	ઓટોમેટિક	મેન્યુઅલ
ઉપયોગ	કલેકશન્સ પર પુનરાવર્તન	શરત સુધી પુનરાવર્તન

Listing 11. લૂપ તુલના

```

1 # For લૂપ
2 for i in range(5):
3     print(i)
4
5 # While લૂપ
6 i = 0
7 while i < 5:
8     print(i)
9     i += 1

```

મેમરી ટ્રીક

“For જાણીતા While અજાણ્યા”

પ્રશ્ન 4(b) [4 ગુણ]

નીચેના જોડકા બનાવો.

જવાબ

- A. If statement → 3. ચોક્કસ સ્થિતિના આધારે કોડના બ્લોકને શરતીવાર ચલાવવા માટે વપરાય છે
- B. While loop → 1. જ્યાં સુધી કોઈ ચોક્કસ સ્થિતિ પૂરી થાય ત્યાં સુધી કોડના બ્લોકને વારંવાર ચલાવે છે
- C. Break statement → 5. વર્તમાન લૂપને સમાપ્ત કરે છે અને આગલા પુનરાવર્તન તરફ આગળ વધે છે
- D. Continue statement → 2. વર્તમાન પુનરાવર્તનને અવગણે છે અને આગળના એક તરફ આગળ વધે છે

મુખ્ય મુદ્દાઓ:

- If Statement: શરતીવાર એક્ઝિક્યુશન
- While Loop: શરત સાથે પુનરાવર્તિત એક્ઝિક્યુશન
- Break: લૂપમાંથી સંપૂર્ણ બહાર નીકળો
- Continue: માત્ર વર્તમાન પુનરાવર્તન છોડો

મેમરી ટ્રીક

“If શરતો While પુનરાવર્તન Break બહાર Continue છોડો”

પ્રશ્ન 4(c) [7 ગુણ]

ઉદાહરણની મદદથી નીચેના તફાવત સમજાવો: a) Argument and Parameter b) Global and Local variable

જવાબ**a) Argument vs Parameter:****Listing 12. Arguments vs Parameters**

```

1 def greet(name, age): # name, age પેરામીટર્સ છે
2   print(f"હેલો {name}, તમારી ઉંમર {age} વર્ષ છે")
3
4 greet("રાજ", 20) # રાજ", 20 આર્ગ્યુમેન્ટ્સ છે

```

b) Global vs Local Variable:**Listing 13. Global vs Local Variables**

```

1 x = 10 # Global variable
2
3 def my_function():
4   y = 5 # Local variable
5   global x
6   x = 15 # Global variable ને બદલતું
7   print(f"Local y: {y}")
8   print(f"Global x: {x}")
9
10 my_function()
11 print(f"બહાર Global x: {x}")

```

કોષ્ટક 10. તુલના વિઠળગાવલોકન

પ્રકાર	સ્કોપ	એક્સેસ	ઉદાહરણ
Parameter	ફંક્શન ડેફીનિશન	મૂલ્યો મેળવે છે	def func(param):
Argument	ફંક્શન કોલ	મૂલ્યો પાસ કરે છે	func(argument)
Global	આખો પ્રોગ્રામ	બધે	x = 10
Local	ફંક્શનની અંદર	માત્ર ફંક્શનમાં	ફંક્શનમાં y = 5

મેરી ટ્રીક

“પેરામીટર્સ મેળવે આર્ગ્યુમેન્ટ્સ પાસ કરે Globals બધે Locals ફંક્શન”

પ્રશ્ન 4(a OR) [3 ગુણ]

નીચેના સ્ટેપ્મેન્ટના આઉટપુટ લખો.

જવાબ**Listing 14. Math Functions**

```

1 import math
2 (i) print(math.ceil(-9.7)) # આઉટપુટ: -9
3 (ii) print(math.floor(-9.7)) # આઉટપુટ: -10
4 (iii) print(mathfabs(-12.3)) # આઉટપુટ: 12.3

```

સમજૂતી:

- ceil(-9.7): Ceiling નજીકના integer સુધી ઉપર કરે છે = -9
- floor(-9.7): Floor નજીકના integer સુધી નીચે કરે છે = -10
- fabs(-12.3): Absolute value નેગેટિવ સાઇન દૂર કરે છે = 12.3

મુખ્ય મુદ્દાઓ:

- Math Module:** ગાણિતિક ફંક્શન્સ માટે ઇમ્પોર્ટ જરૂરી
- નેગેટિવ નંબર્સ:** Ceiling અને floor નેગેટિવ સાથે અલગ રીતે કામ કરે છે
- Absolute Value:** હંમેશા પોઝિટિવ મૂલ્ય પરત કરે છે

મેમરી ટ્રીક

“Ceiling ઉપર Floor નીચે Absolute પોઝિટિવ”

પ્રશ્ન 4(b OR) [4 ગુણ]

Function ના ફાયદા લખો.

જવાબ

કોષ્ટક 11. ફંક્શનના ફાયદા

ફાયદો	વર્ણન
કોડ રીયુઝનિભલિટી	એકવાર લખો, ઘણીવાર વાપરો
મોડ્યુલારિટી	જટિલ સમસ્યાઓને નાના ભાગોમાં વિભાજિત કરો
સરળ ડિઝિન્ગ	એરર્સ સરળતાથી શોધો અને ઠીક કરો
કોડ ઓર્ગનાઇઝેશન	વધુ સારુ માળાંનું અને વાંચવાક્ષમતા
મેન્ટેનિભલિટી	અપડેટ અને મોડિફિય કરવું સરળ
જટિલતા ઘટાડવી	જટિલ ઓપરેશન્સને સરળ બનાવો

મુખ્ય ફાયદાઓ:

- પુનરાવર્તન ટાળો: ફરીથી તે જ કોડ લખવાની જરૂર નથી
- સીમ કોલેબોરેશન: અલગ અલગ લોકો અલગ ફંક્શન્સ પર કામ કરી શકે છે
- ટેસ્ટિંગ: દરેક ફંક્શનને સ્વતંત્ર રીતે ટેસ્ટ કરી શકાય છે

મેમરી ટ્રીક

“રીયુઝ મોડ્યુલર ડિબગ ઓર્ગનાઇઝ મેન્ટેન ઘટાડો”

પ્રશ્ન 4(c OR) [7 ગુણ]

બિલ્ટ ઇન ફંક્શન્સનો ઉપયોગ કયા વિના આપેલ લિસ્ટમાં સૌથી નાનો અને સૌથી મોટો નંબર શોધવા માટે પ્રોગ્રામ લખો.

જવાબ

Listing 15. Min Max શોધવા માટે કોડ

```

1 # બિલ્ટિન - ફંક્શન્સ વનિા સૌથી નાનો અને મોટો શોધવાનો પ્રોગ્રામ
2 def find_min_max(numbers):
3     """બિલ્ટિન - ફંક્શન્સ વનિા minimum અને maximum શોધો"""
4     if not numbers:
5         return None, None
6
7     smallest = numbers[0]
8     largest = numbers[0]
9
10    for num in numbers[1:]:

```

```

11     if num < smallest:
12         smallest = num
13     if num > largest:
14         largest = num
15
16     return smallest, largest
17
18 # ઇનપુટ લાસ્ટ
19 n = int(input("એલમેન્ટ્સની સંખ્યા દાખલ કરો: "))
20 numbers = []
21
22 for i in range(n):
23     num = float(input(f"નંબર {i+1} દાખલ કરો: "))
24     numbers.append(num)
25
26 # min અને max શેધો
27 min_num, max_num = find_min_max(numbers)
28
29 print(f"લાસ્ટ: {numbers}")
30 print(f"સૌથી નાનો નંબર: {min_num}")
31 print(f"સૌથી મોટો નંબર: {max_num}")

```

મુખ્ય મુદ્દાઓ:

- મેન્યુઆલ કમ્પ્યુટરિઝન: min()/max() ની જગ્યાએ if શરતોનો ઉપયોગ કરો
- વેસ્ટિબુલ ઇનિશિયલાઇઝ કરો: પહેલા એલિમેન્ટ્સની શરૂઆતી શરૂ કરો
- ક્લૂપ થૂં: દરેક એલિમેન્ટને વર્તમાન min/max સાથે કમ્પેર કરો

મેમરી ટ્રીક

“ઇનિશિયલાઇઝ કમ્પેર અપડેટ રિટર્ન”

પ્રશ્ન 5(a) [3 ગુણ]

Python માં list માટેના sort() અને sorted() મેથડ વચ્ચેનો તફાવત સમજાવો.

જવાબ

કોષ્ટક 12. sort() vs sorted()

ફીચર	sort()	sorted()
રિટર્ન ટાઇપ	None (ઓરિજિનલ બદલે છે)	નવી સોર્ટેડ લિસ્ટ
ઓરિજિનલ લિસ્ટ	ઇન-પ્લેસ બદલે છે	અપરિવર્તિત
ઉપયોગ	list.sort()	sorted(list)
મેમરી	કાર્યક્રમ	વધારાની મેમરી વાપરે છે

Listing 16. Sort તુલના

```

1 # sort() મેથડ
2 list1 = [3, 1, 4, 2]
3 list1.sort()
4 print(list1) # [1, 2, 3, 4]
5
6 # sorted() ફંક્શન
7 list2 = [3, 1, 4, 2]
8 new_list = sorted(list2)
9 print(list2) # [3, 1, 4, 2] અપરવિરૂદ્ધતિ()

```

```
10 print(new_list) # [1, 2, 3, 4]
```

મેમરી ટ્રીક

“Sort બદલે છે Sorted બનાવે છે”

પ્રશ્ન 5(b) [4 ગુણ]

ઉદાહરણ સાથે Python માં સ્ટ્રિંગને ટ્રાવર્સ કરવાની વિવિધ રીત સમજાવો.

જવાબ

સ્ટ્રિંગ ટ્રાવર્સલ મેથ્ડ્સ:

1. For લૂપ વાપરીને:

```
1 text = "Python"
2 for char in text:
3     print(char, end=" ") # P y t h o n
```

2. ઇન્ડેક્સ વાપરીને:

```
1 text = "Python"
2 for i in range(len(text)):
3     print(text[i], end=" ") # P y t h o n
```

3. While લૂપ વાપરીને:

```
1 text = "Python"
2 i = 0
3 while i < len(text):
4     print(text[i], end=" ")
5     i += 1
```

4. Enumerate વાપરીને:

```
1 text = "Python"
2 for index, char in enumerate(text):
3     print(f"{index}:{char}", end=" ") # 0:P 1:y 2:t 3:h 4:o 5:n
```

મેમરી ટ્રીક

“For ઇન્ડેક્સ While Enumerate”

પ્રશ્ન 5(c) [7 ગુણ]

નીચે આપેલા સ્ક્રિપ્ટનું આઉટપુટ લખો.

જવાબ

Listing 17. સ્ટ્રિંગ સ્ક્રિપ્ટસ આઉટપુટ

```
1 (1) s = "Hello, World!"
```

```

2 print(s[0:5])      # આઉટપુટ: Hello
3
4 (2) lst = [1, 2, 3, 4, 5]
5   print(lst[2:4])    # આઉટપુટ: [3, 4]
6
7 (3) s = "python"
8   print(len(s))     # આઉટપુટ: 6
9
10 (4) lst = [5, 2, 3, 1, 8]
11   lst.sort()        # lst બને છે [1, 2, 3, 5, 8]
12
13 (5) s1 = "hello"
14   s2 = "world"
15   print(s1 + s2)    # આઉટપુટ: helloworld
16
17 (6) lst = [1, 2, 3, 4, 5]
18   print(sum(lst))   # આઉટપુટ: 15
19
20 (7) s = "python"
21   print(s[::-1])    # આઉટપુટ: nohtyP

```

મુખ્ય મુદ્દાઓ:

- સ્લાઇસિંગ: [start:end] સબસ્ટ્રિંગ/સબલિસ્ટ કાઢે છે
- સ્ટ્રિંગ લેન્થ: len() કેરેક્ટરની સંખ્યા પરત કરે છે
- લિસ્ટ સોર્ટિંગ: sort() લિસ્ટને ઇન્ન-પ્લેસ બદલે છે
- સ્ટ્રિંગ કંક્ટેનેશન: + ઓપરેટર સ્ટ્રિંગ્સ જોડે છે
- Sum ફંક્શન: બધા લિસ્ટ એલિમેન્ટ્સ ઉમેરે છે
- રિવર્સ સ્લાઇસિંગ: [::-1] સિકવન્સ ઉલટાવે છે

મેમરી ટ્રીક

“સ્લાઇસ લેન્થ સોર્ટ કંક્ટેનેટ સમ રિવર્સ”

પ્રશ્ન 5(a OR) [3 ગુણ]

Python માં type conversion સમજાવો.

જવાબ

કોષ્ટક 13. ટાઇપ કન્વર્શન

ટાઇપ	ફૂકશન	ઉદાહરણ
int()	ઇન્ટિજરમાં કન્વર્ટ કરો	int("5") -> 5
float()	ફ્લોટમાં કન્વર્ટ કરો	float("3.14") -> 3.14
str()	સ્ટ્રિંગમાં કન્વર્ટ કરો	str(25) -> "25"
bool()	બુલિયનમાં કન્વર્ટ કરો	bool(1) -> True
list()	લિસ્ટમાં કન્વર્ટ કરો	list("abc") -> ['a','b','c']

Listing 18. ટાઇપ કન્વર્શન ઉદાહરણો

```

1 # Implicit conversion
2 x = 5 + 3.2 # int + float = float (8.2)
3
4 # Explicit conversion
5 num_str = "123"
6 num_int = int(num_str) # "123" -> 123

```

મુખ્ય મુદ્દાઓ:

- **Implicit:** Python આપોઆપ કન્વર્ટ કરે છે
- **Explicit:** પ્રોગ્રામર મેન્યુઅલી ફંક્શન્સ વાપરીને કન્વર્ટ કરે છે
- **ટાઇપ સેફ્ટી:** કેટલાક કન્વર્જન એરર્સ આપી શકે છે

મેમરી ટ્રીક

“Implicit આપોઆપ Explicit મેન્યુઅલ”

પ્રશ્ન 5(b OR) [4 ગુણ]

ઉદાહરણ સાથે string પર કન્કટેનેશન અને પુનરાવર્તન કામગીરીને સમજાવો.

જવાબ**સ્ટ્રિંગ ઓપરેશન્સ:****1. કન્કટેનેશન (+):**

```

1 str1 = "Hello"
2 str2 = "World"
3 result = str1 + " " + str2
4 print(result) # Hello World

5
6 # મલૂટિપિલ કન્કટેનેશન
7 name = "Python"
8 version = "3.9"
9 info = "Language: " + name + " Version: " + version
10 print(info) # Language: Python Version: 3.9

```

2. પુનરાવર્તન (*):

```

1 text = "Hi! "
2 repeated = text * 3
3 print(repeated) # Hi! Hi! Hi!

4
5 # પેટ્રેન બનાવવું
6 pattern = "_" * 10
7 print(pattern) # -----

```

મુખ્ય મુદ્દાઓ:

- કન્કટેનેશન: + વાપરીને સ્ટ્રિંગ્સ જોડે છે
- પુનરાવર્તન: * વાપરીને સ્ટ્રિંગને n વખત રિપીટ કરે છે
- અપરિવર્તનીય: ઓરિજિનલ સ્ટ્રિંગ્સ અપરિવર્તિત રહે છે

મેમરી ટ્રીક

“Plus જોડે Star રિપીટ કરે”

પ્રશ્ન 5(c OR) [7 ગુણ]

શબ્દમાળામાં સ્વર, વ્યંજન, અપરક્સ, લોઅરક્સ અક્ષરોની સંખ્યાની ગણતરી પ્રદર્શિત કરવા માટેનો પ્રોગ્રામ લખો.

જવાબ

Listing 19. સ્ટ્રિંગ વિશ્લેષણ પ્રોગ્રામ

```

1 def analyze_string(text):
2     """વિવિધ કેરેક્ટર પ્રકારો માટે સ્ટ્રાઇન્નું વશ્વિલેખણ કરો"""
3     vowels = "aeiouAEIOU"
4
5     vowel_count = 0
6     consonant_count = 0
7     uppercase_count = 0
8     lowercase_count = 0
9
10    for char in text:
11        if char.isalpha(): # કેરેક્ટર આલફાબેટ છે કે નહીં ચેક કરો
12        if char in vowels:
13            vowel_count += 1
14        else:
15            consonant_count += 1
16
17        if char.isupper():
18            uppercase_count += 1
19        elif char.islower():
20            lowercase_count += 1
21
22    return vowel_count, consonant_count, uppercase_count, lowercase_count
23
24 # ઇનપુટ સ્ટ્રાઇન્નું
25 text = input("સ્ટ્રાઇન્ દાખલ કરો: ")
26
27 # સ્ટ્રાઇન્નું વશ્વિલેખણ કરો
28 vowels, consonants, uppercase, lowercase = analyze_string(text)
29
30 # પરિણામો દર્શાવો
31 print(f"સ્ટ્રાઇન્: {text}")
32 print(f"સ્લોવર: {vowels}")
33 print(f"વ્યાંજન: {consonants}")
34 print(f"અપરકેસ: {uppercase}")
35 print(f"લોઓરકેસ: {lowercase}")

```

મુખ્ય મુદ્દાઓ:

- કેરેક્ટર કલાસિફિકેશન: isalpha(), isupper(), islower() નો ઉપયોગ કરો
- સ્વર ચેક: સ્વર સ્ટ્રિંગ સાથે કાંપેર કરો
- લૂપ પ્રોસેસિંગ: દર્દક કેરેક્ટરને વ્યક્તિગત રીતે ચેક કરો

મેમરી ટ્રીક

“ચેક કલાસિફિકેશન કાઉન્ટ ડિસ્પલે”