Subject Name (Gujarati)

1323203 -- Summer 2024

Semester 1 Study Material

Detailed Solutions and Explanations

પ્રશ્ન 1(અ) [3 માર્ક્સ]

ફ્લોચાર્ટ અને અલ્ગોરિધમના મહત્વની યાદી આપો.

જવાબ

ફ્લોચાર્ટનું મહત્વ	અલ્ગોરિધમનું મહત્વ
પ્રોગ્રામ લોજિકનું દૃશ્ય નિરૂપણ	સમસ્યાને ઉકેલવા માટેનું પગલાંવાર પ્રક્રિયા
ભૂલોને સરળતાથી શોધવા અને સુધારવા	ભાષાથી સ્વતંત્ર ઉકેલ અભિગમ
જટિલ પ્રક્રિયાઓને સમજવામાં મદદ	પ્રોગ્રામિંગની પાયારૂપ શરૂઆત
ટીમના સભ્યો વચ્ચે સંદેશાવ્યવહાર સુધારે	કોડિંગ શરૂ કરતા પહેલા લોજિક નિર્ધારિત કરે

મેમરી ટ્રીક

``VASE નિર્ણયો'' - Visualize, Analyze, Sequence, Execute

પ્રશ્ન 1(બ) [4 માર્ક્સ]

દાખલ કરેલ સંખ્યા ઈવન કે ઓડ છે તે શોધવા માટે ફ્લોચાર્ટ દોરો.

જવાબ

```
flowchart LR
    A[Start] {-{-} B[Input Number n]}
    B {-{-} C\{n \% 2 == 0?\}}
    C {-{-}|Yes| D[Print Even Number]}
    C {-{-}|No| E[Print Odd Number]}
    D {-{-} F[End]}
    E {-{-} F}
```

મુખ્ય પગલાં:

- **ડેટા એકત્રીકરણ**: વપરાશકર્તા પાસેથી નંબર મેળવો
- **મોડ્યુલો ઓપરેશન**: 2 વડે ભાગીને શેષ તપાસો
- **શરતી આઉટપુટ**: શેષના આધારે પરિણામ દર્શાવો

મેમરી ટ્રીક

``MODE'' - Modulo Operation Determines Evenness

પ્રશ્ન 1(ક) [7 માર્ક્સ]

બધા લોજિકલ ઓપરેટરોની યાદી બનાવો અને પાયથોન કોડનું ઉદાહરણ આપીને દરેકને સમજાવો.

જવાબ

ઓપરેટર	વર્ણન	ઉદાહરણ	આઉટપુટ
`and`	બંને સ્ટેટમેન્ટ સાચા હોય તો True રિટર્ન કરે	`x = 5; print(x > 3 and x < 10)`	`True`
`or`	બે સ્ટેટમેન્ટમાંથી એક સાચું હોય તો True રિટર્ન કરે	`x = 5; print(x > 10 or x == 5)`	`True`
`not`	પરિણામને ઉલટાવે, જો પરિણામ સાચું હોય તો False રિટર્ન કરે	`x = 5; print(not(x > 3))`	`False`

કોડ ઉદાહરણ:

```
\#      AND
age = 25
income = 50000
print("      :", age {} 18 and income {} 30000)  \# True

\#      OR
has\_credit\_card = False
has\_cash = True
print("         :", has\_credit\_card or has\_cash)  \# True

\#      NOT
is\_holiday = False
print("          :", not is\_holiday)  \# True
```

મેમરી ટ્રીક

``AON સ્પષ્ટતા'' - And, Or, Not લોજિકલ સ્પષ્ટતા માટે

પ્રશ્ન 1(ક) OR [7 માર્ક્સ]

એક પાયથોન પ્રોગ્રામ લખો કરો જે આપેલ ડેટા પર સાદા વ્યાજ અને ચક્રવૃદ્ધિ વ્યાજની ગણતરી કરી શકે.

જવાબ

```
\#

\#
principal = float(input("          : "))
rate = float(input("           (\%  ): "))
time = float(input("           (    ): "))

\#
simple\_interest = (principal * rate * time) / 100

\#
compound\_interest = principal * ((1 + rate/100) ** time {-} 1)

\#
print("     :", round(simple\_interest, 2))
print("       :", round(compound\_interest, 2))
```

મુખ્ય સૂત્રો:

- **સાદું વ્યાજ (SI)**: મૂળ રકમ $\times \times /100$
- **ચક્રવૃદ્ધિ વ્યાજ (CI)**: મૂળ રકમ $\times((1 + /100)\text{^} - 1)$

મેમરી ટ્રીક

``PRT નાણાં વૃદ્ધિ'' - Principal, Rate, Time નાણાંની વૃદ્ધિ

પ્રશ્ન 2(અ) [3 માર્ક્સ]

આપેલ ત્રણ નંબરોમાંથી ન્યૂનતમ સંખ્યા શોધવા માટે પાયથોન પ્રોગ્રામ બનાવો.

જવાબ

```
\#

\#
num1 = float(input("           : "))
num2 = float(input("           : "))
num3 = float(input("            : "))

\#   {-  min()                   }
minimum = min(num1, num2, num3)

\#
print("       :", minimum)
```

મેમરી ટ્રીક

``MIN શોધે ન્યૂનતમ'' - Minimum Is Numerically શોધાય ન્યૂનતમ સાથે

પ્રશ્ન 2(બ) [4 માર્ક્સ]

સ્યુડોકોડ વ્યાખ્યાયિત કરો. x, y અને z ત્રણમાંથી સૌથી મોટી સંખ્યા શોધવા માટે સ્યુડોકોડ લખો.

જવાબ

સ્યુડોકોડની વ્યાખ્યા

કમ્પ્યુટર પ્રોગ્રામે શું કરવું જોઈએ તેનું વિગતવાર અને વાંચી શકાય તેવું વર્ણન, જે પ્રોગ્રામિંગ ભાષાને બદલે ઔપચારિક શૈલીમાં લખાયેલી કુદરતી ભાષામાં વ્યક્ત કરવામાં આવે છે.

ત્રણ નંબરોમાંથી સૌથી મોટો શોધવા માટે સ્યુડોકોડ:

```
BEGIN
    INPUT x, y, z
    SET largest = x

    IF y > largest THEN
        SET largest = y
    END IF

    IF z > largest THEN
        SET largest = z
    END IF

    OUTPUT "          : ", largest
END
```

મેમરી ટ્રીક

``PIE લખાણ'' - Program Ideas Expressed સરળ લખાણમાં

પ્રશ્ન 2(ક) [7 માર્ક્સ]

પાયથોનમાં વાઈલ લૂપને તેના સિન્ટેક્સ, ફ્લોચાર્ટ અને પાયથોન કોડના ઉદાહરણ સાથે સમજાવો.

જવાબ

સિન્ટેક્સ:

```
while   :
    \#
```

ફ્લોચાર્ટ:

```
flowchart LR
    A[Start] {-{-} B[Variables Initialize   ]}
    B {-{-} C\{      ?\}}
    C {-{-}| | D[           ]}
    D {-{-} C}
    C {-{-}| | E[End]}
```

કોડ ઉદાહરણ:

```
\#    5        while
count = 1

while count {=} 5:
    print(count)
    count += 1  \#

\#    :
\# 1
\# 2
\# 3
\# 4
\# 5
```

મુખ્ય લક્ષણો:

- **એન્ટ્રી કંટ્રોલ**: લૂપ એક્ઝિક્યુશન પહેલાં શરત ચકાસવામાં આવે છે
- **ઇનિશિયલાઇઝેશન**: લૂપ પહેલાં વેરિએબલ્સ સેટ કરવામાં આવે છે
- **અપડેશન**: લૂપની અંદર વેરિએબલ્સ અપડેટ કરવામાં આવે છે
- **ટર્મિનેશન**: શરત ખોટી થાય ત્યારે લૂપ બહાર નીકળે છે

મેમરી ટ્રીક

``IUTE લૂપ'' - Initialize, Update, Test for Exit

પ્રશ્ન 2(અ) OR [3 માર્ક્સ]

પાયથોનમાં કન્ટિન્યુ સ્ટેટમેન્ટનું ટૂંકમાં વર્ણન કરો.

જવાબ

પાયથોનમાં કન્ટિન્યુ સ્ટેટમેન્ટ
કન્ટિન્યુ સ્ટેટમેન્ટ લૂપના વર્તમાન ઇટરેશનને છોડી દે છે અને આગલા ઇટરેશનથી ચાલુ રાખે છે જ્યારે એનકાઉન્ટર થાય, ત્યારે કન્ટિન્યુ સ્ટેટમેન્ટ પછીનો લૂપનો કોડ છોડી દેવામાં આવે છે ચોક્કસ શરતોને છોડીને લૂપને ચાલુ રાખવા માટે ઉપયોગી છે

કોડ ઉદાહરણ:

```
\#
for i in range(1, 6):
    if i \% 2 == 0:
        continue
    print(i)  \#    1, 3, 5
```

મેમરી ટ્રીક

``SKIP આગળ'' - Skip Keeping Iteration Process

પ્રશ્ન 2(બ) OR [4 માર્ક્સ]

નીચેના કોડનું આઉટપુટ શું હશે?

```
x=8
y=2
print (x*y)
print (x ** y)
print (x \% y)
print(x{}y)
```

જવાબ

ઓપરેશન	પરિણામ	સમજૂતી
x*y	16	ગુણાકાર: $8 \times 2 = 16$
x**y	64	પાવર: $8^2 = 64$
x%y	0	મોડ્યુલો (શેષ): $8 \div 2 = 40$
x>y	True	તુલના: 8 > 2 સાચું છે

મેમરી ટ્રીક

``MEMO'' - Multiply, Exponent, Modulo, Operator comparison

પ્રશ્ન 2(ક) OR [7 માર્ક્સ]

પાયથોનમાં ઈફ-ઈએલઈએફ-એલ્સ લેડરને તેના સિન્ટેક્સ, ફ્લોચાર્ટ અને પાયથોન કોડના ઉદાહરણ સાથે સમજાવો.

જવાબ

સિન્ટેક્સ:

```
if   1:
    \#       1
elif   2:
    \#       2
elif   3:
    \#       3
else:
    \#       4
```

ફ્લોચાર્ટ:

```
flowchart LR
    A[Start] {-{-} B\{  1?\}}
    B {-{-}|  | C[      1        ]}
```

```
    B {-{-}|  | D\{  2?\}}
    D {-{-}|  | E[      2       ]}
    D {-{-}|  | F\{  3?\}}
    F {-{-}|  | G[      3       ]}
    F {-{-}|  | H[      4       ]}
    C {-{-} I[End]}
    E {-{-} I}
    G {-{-} I}
    H {-{-} I}
```

કોડ ઉદાહરણ:

```
\#
marks = 75

if marks {=} 90:
    grade = "A+"
elif marks {=} 80:
    grade = "A"
elif marks {=} 70:
    grade = "B"
elif marks {=} 60:
    grade = "C"
else:
    grade = "D"

print(" :", grade)  \#    :   : B
```

મુખ્ય લક્ષણો:
- **અનુક્રમિક મૂલ્યાંકન**: શરતો ઉપરથી નીચે તપાસવામાં આવે છે
- **અનન્ય એક્ઝિક્યુશન**: માત્ર એક બ્લોક એક્ઝિક્યુટ થાય છે
- **ડિફોલ્ટ એક્શન**: જો કોઈ શરત સાચી ન હોય તો else બ્લોક એક્ઝિક્યુટ થાય છે

મેમરી ટ્રીક

``SEEP લોજિક'' - Sequential Evaluation with Exclusive Path

પ્રશ્ન 3(અ) [3 માર્ક્સ]

લૂપ્સનો ઉપયોગ કરીને 1 થી 20 વચ્ચેની એકી સંખ્યાઓ છાપવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

```
\# 1   20

\# range   step   for
for number in range(1, 21, 2):
    print(number, end=" ")

\#    : 1 3 5 7 9 11 13 15 17 19
```

વૈકલ્પિક અભિગમ:

```
\# if      for
for number in range(1, 21):
    if number \% 2 != 0:
        print(number, end=" ")
```

મેમરી ટ્રીક

``STEO'' - Skip Two, Extract Odds

પ્રશ્ન 3(બ) [4 માર્ક્સ]

નેસ્ટેડ ઈફ સ્ટેટમેન્ટને સંક્ષિપ્તમાં સમજાવો.

જવાબ

નેસ્ટેડ ઈફ સ્ટેટમેન્ટ
બીજા if સ્ટેટમેન્ટની અંદર એક if સ્ટેટમેન્ટ
વધુ જટિલ શરતી લોજિકની મંજૂરી આપે છે
બાહ્ય if સાચું હોય ત્યારે જ આંતરિક if મૂલ્યાંકન કરવામાં આવે છે
નેસ્ટિંગના ઘણા સ્તરો હોઈ શકે છે

કોડ ઉદાહરણ:

```
age = 25
income = 50000

if age {} 18:
    print("  ")
    if income {} 30000:
        print("            ")
    else:
        print("            ")
else:
    print("  ")
```

મેમરી ટ્રીક

``LION'' - Layered If-statements Operating Nested

પ્રશ્ન 3(ક) [7 માર્ક્સ]

યુઝર ડિફાઈન ફંક્શનનો ઉપયોગ કરીને દાખલ કરેલ નંબર `આર્મસ્ટ્રોંગ નંબર' અથવા પેલિન્ડ્રોમ છે તે તપાસવા માટે પ્રોગ્રામ લખો એ જેમાં કૉલિંગ ફંક્શનમાં આર્ગ્યુમેંટ તરીકે નંબર આપવામા આવે છે.

જવાબ

```
\#

def check\_number(num):
    \#
    temp = num
    digits = len(str(num))
    sum = 0

    while temp {} 0:
        digit = temp \% 10
        sum += digit ** digits
        temp //= 10

    is\_armstrong = (sum == num)

    \#
    is\_palindrome = (str(num) == str(num)[::{-}1])
```

```
    \#
    return is\_armstrong, is\_palindrome

\#
number = int(input("          : "))

\#
armstrong, palindrome = check\_number(number)

if armstrong:
    print(number, "           ")
else:
    print(number, "          ")

if palindrome:
    print(number, "         ")
else:
    print(number, "        ")
```

આર્મસ્ટ્રોંગ ઉદાહરણો:

- 153: $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$
- 370: $3^3 + 7^3 + 0^3 = 27 + 343 + 0 = 370$

મેમરી ટ્રીક

``APTEST'' - Armstrong Palindrome Test Equal Sum Test

પ્રશ્ન 3(અ) OR [3 માર્ક્સ]

૧ થી ૧૦૦ સુધી નો સરવાળો શોધવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

```
\# 1   100

\#     1:
total = 0
for num in range(1, 101):
    total += num
print("             :", total)

\#     2:    n(n+1)/2
n = 100
sum\_formula = n * (n + 1) // 2
print("              :", sum\_formula)

\#    :
\#              : 5050
\#               : 5050
```

મેમરી ટ્રીક

``SUM સૂત્ર'' - Sum Using Mathematical સૂત્ર

પ્રશ્ન 3(બ) OR [4 માર્ક્સ]

નીચેની પેટર્ન છાપવા માટે પાયથોન પ્રોગ્રામ લખો.

```
1
2 3
```

```
4 5 6
7 8 9 10
```

જવાબ

```
\#

num = 1
for i in range(1, 5):  \# 4
    for j in range(i):  \#
        print(num, end=" ")
        num += 1
    print()  \#
```

પેટર્ન લોજિક:

- **પંક્તિ 1**: 1 સંખ્યા (1)
- **પંક્તિ 2**: 2 સંખ્યાઓ (2, 3)
- **પંક્તિ 3**: 3 સંખ્યાઓ (4, 5, 6)
- **પંક્તિ 4**: 4 સંખ્યાઓ (7, 8, 9, 10)

મેમરી ટ્રીક

``CNIR'' - Counter Number Increases with Rows

પ્રશ્ન 3(ક) OR [7 માર્ક્સ]

ફંક્શનનો ઉપયોગ કરીને પ્રોગ્રામ લખો જે દાખલ કરેલ નંબરને ઉલટાવે

જવાબ

```
\#

def reverse\_number(num):
    """               """
    return int(str(num)[::{-}1])

def reverse\_string(text):
    """               """
    return text[::{-}1]

\#
def main():
    choice = input("            ? (n      , s       ): ")

    if choice.lower() == {n}:
        num = int(input("          : "))
        print("      :", reverse\_number(num))
    elif choice.lower() == {s}:
        text = input("           : ")
        print("       :", reverse\_string(text))
    else:
        print("       !")

\#
main()
```

નંબર ઉલટાવવા માટે વૈકલ્પિક પદ્ધતિ:

```
def reverse\_number\_algorithm(num):
    reversed\_num = 0
    while num {} 0:
        digit = num \% 10
        reversed\_num = reversed\_num * 10 + digit
```

```
        num //= 10
    return reversed\_num
```

મેમરી ટ્રીક

``FLIP અંકો'' - Function Logic Inverts Position of અંકો

પ્રશ્ન 4(અ) [3 માર્ક્સ]

યોગ્ય પાયથોન કોડ ઉદાહરણ સાથે પાયથોન મેથ મોડ્યુલનું વર્ણન કરો.

જવાબ

પાયથોન મેથ મોડ્યુલની વિશેષતાઓ
ગાણિતિક ફંક્શન્સ અને સ્થિરાંકો પ્રદાન કરે છે
ત્રિકોણમિતિય, લોગરિધમિક અને અન્ય ફંક્શન્સ શામેલ છે
pi અને e જેવા ગાણિતિક સ્થિરાંકો ધરાવે છે
ઉપયોગ કરતા પહેલા import કરવું જરૂરી છે

કોડ ઉદાહરણ:

```
import math

\#
print("pi     :", math.pi)  \# 3.141592653589793
print("e    :", math.e)    \# 2.718281828459045

\#
print("16      :", math.sqrt(16))  \# 4.0
print("5      3:", math.pow(5, 3))  \# 125.0

\#           (   )
print("90^     :", math.sin(math.pi/2))  \# 1.0
print("0^      :", math.cos(0))  \# 1.0

\#
print("100      10   :", math.log10(100))  \# 2.0
print("e        :", math.log(math.e))  \# 1.0
```

મેમરી ટ્રીક

``CALM ઓપરેશન્સ'' - Constants And Logarithmic Mathematical ઓપરેશન્સ

પ્રશ્ન 4(બ) [4 માર્ક્સ]

વેરીએબલના સ્કોપને સમજાવતો પાયથોન પ્રોગ્રામ લખો.

જવાબ

```
\#

\#
global\_var = "      "

def demonstration():
    \#
    local\_var = "      "
```

```
    \#
    print("       {-        :"}, global\_var)

    \#
    print("       {-        :"}, local\_var)

    \#
    global\_var = "              "
    print("       {-       :"}, global\_var)

\#
demonstration()

\#
print("       {-        :"}, global\_var)

\#
\# print("       {-        :", local\_var)  \#  !}
```

આઉટપુટ:

```
       -        :
       -       :
       -      :
        -       :
```

મેમરી ટ્રીક

``GLOVES'' - Global Local Variable Encapsulation System

પ્રશ્ન 4(ક) [7 માર્ક્સ]

લિસ્ટ પદ્ધતિઓ અને તેના બિલ્ટ-ઇન કાર્યો સમજાવો

જવાબ

પદ્ધતિ/ફંક્શન	વર્ણન	ઉદાહરણ	આઉટપુટ
`append()`	અંતે એલિમેન્ટ ઉમેરે છે	`fruits = ['apple']; fruits.append('banana'); print(fruits)`	`['apple', 'banana']`
`insert()`	ચોક્કસ પોઝિશન પર એલિમેન્ટ ઉમેરે	`nums = [1, 3]; nums.insert(1, 2); print(nums)`	`[1, 2, 3]`
`remove()`	ચોક્કસ આઈટમ દૂર કરે	`colors = ['red', 'blue']; colors.remove('red'); print(colors)`	`['blue']`
`pop()`	ચોક્કસ ઇન્ડેક્સ પર આઈટમ દૂર કરે	`letters = ['a', 'b', 'c']; x = letters.pop(1); print(x, letters)`	`b ['a', 'c']`
`clear()`	બધા એલિમેન્ટ્સ દૂર કરે	`items = [1, 2]; items.clear(); print(items)`	`[]`
`len()`	એલિમેન્ટ્સની સંખ્યા પાછી આપે	`print(len([1, 2, 3]))`	`3`

`sorted()`	સૉર્ટેડ લિસ્ટ પાછી આપે	`print(sorted([3, 1, 2]))` → `[1, 2, 3]`
`max()/min()`	મહત્તમ/લઘુત્તમ મૂલ્ય પાછું આપે	`print(max([5, 10, 3]), min([5, 10, 3]))` → `10 3`

કોડ ઉદાહરણ:

```
\#
my\_list = [3, 1, 4, 1, 5]
print(" :", my\_list)

\#
my\_list.append(9)
print("append  :", my\_list)

my\_list.insert(2, 7)
print("insert  :", my\_list)

\#
my\_list.remove(1)  \#    1
print("remove  :", my\_list)

popped = my\_list.pop()  \#
print("pop     :", popped)
print("pop  :", my\_list)

\#
print("  :", len(my\_list))
print("  :", sorted(my\_list))
print("  :", sum(my\_list))
print("1    :", my\_list.count(1))
```

મેમરી ટ્રીક

``LISP ઓપરેશન્સ'' - List Insert Sort Pop ઓપરેશન્સ

પ્રશ્ન 4(અ) OR [3 માર્ક્સ]

પાયથોન સ્ટાન્ડર્ડ લાઇબ્રેરી ગાણિતિક કાર્યોની સૂચિ બનાવો.

જવાબ

ગાણિતિક ફંક્શન	વર્ણન	ઉદાહરણ
`abs()`	નિરપેક્ષ મૂલ્ય પાછું આપે	`abs(-5)` → 5
`round()`	નજીકના પૂર્ણાંક સુધી ગોળ કરે	`round(3.7)` → 4
`max()`	સૌથી મોટી આઈટમ પાછી આપે	`max(1, 5, 3)` → 5
`min()`	સૌથી નાની આઈટમ પાછી આપે	`min(1, 5, 3)` → 1
`sum()`	ઇટરેબલની આઈટમ્સનો સરવાળો કરે	`sum([1, 2, 3])` → 6
`pow()`	x ને y ની ઘાત પાછી આપે	`pow(2, 3)` → 8
`divmod()`	ભાગફળ અને શેષ પાછા આપે	`divmod(7, 2)` → `(3, 1)`

math મોડ્યુલમાંથી વધારાના:

- `math.sqrt()`: વર્ગમૂળ
- `math.floor()`: નીચે ગોળ કરે
- `math.ceil()`: ઉપર ગોળ કરે
- `math.factorial()`: ફેક્ટોરિયલ
- `math.gcd()`: મહત્તમ સામાન્ય અવયવ

મેમરી ટ્રીક

``SMART ગણતરી'' - Standard Mathematical Arithmetic Routines and Tools

પ્રશ્ન 4(બ) OR [4 માર્ક્સ]

પાયથોનમાં બિલ્ટ ઇન ફંક્શન સમજાવો.

જવાબ

પાયથોનમાં બિલ્ટ-ઇન ફંક્શન્સ
કોઈપણ મોડ્યુલ ઇમ્પોર્ટ કર્યા વિના પાયથોનમાં ઉપલબ્ધ પ્રી-ડિફાઇન્ડ ફંક્શન્સ
કોઈપણ પ્રીફિક્સ વિના સીધા જ કૉલ કરી શકાય છે
સામાન્ય ઓપરેશન્સ કરવા માટે ડિઝાઇન કરેલ છે
ઉદાહરણોમાં print(), len(), type(), input(), range() શામેલ છે

કેટેગરીઓ સાથે ઉદાહરણો:

```
\#
print(int("10"))        \# 10
print(float("10.5"))    \# 10.5
print(str(10))          \# "10"
print(list("abc"))      \# [{a, b, c]}

\#
print(abs({-}7))          \# 7
print(round(3.7))       \# 4
print(max(5, 10, 3))    \# 10

\#
print(len("hello"))     \# 5
print(sorted([3,1,2])) \# [1, 2, 3]
print(sum([1, 2, 3]))   \# 6
```

મેમરી ટ્રીક

``EPIC ફંક્શન્સ'' - Embedded Python Integrated Core ફંક્શન્સ

પ્રશ્ન 4(ક) OR [7 માર્ક્સ]

વાક્યમાં રહેલ સ્વરો, વ્યંજન, અપરકેસ, લોઅરકેસ અક્ષરોની સંખ્યા ગણવા અને દર્શાવવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

```
\#        ,    ,    ,

def analyze\_string(text):
    \#
    vowels = 0
    consonants = 0
    uppercase = 0
    lowercase = 0

    \#
    vowel\_set = \{{a}, {e}, {i}, {o}, {u}\}

    \#
    for char in text:
        \#
```

```
        if char.isalpha():
            \#
            if char.isupper():
                uppercase += 1
            else:
                lowercase += 1

            \#          ( {-    )}
            if char.lower() in vowel\_set:
                vowels += 1
            else:
                consonants += 1

    \#
    return vowels, consonants, uppercase, lowercase

\#
text = input("          : ")

\#
vowels, consonants, uppercase, lowercase = analyze\_string(text)

\#
print("      :", vowels)
print("       :", consonants)
print("           :", uppercase)
print("            :", lowercase)
```

ઉદાહરણ:

- ઇનપુટ: ``Hello World!''
- આઉટપુટ:
  - સ્વરો: 3 (e, o, o)
  - વ્યંજનો: 7 (H, l, l, W, r, l, d)
  - અપરકેસ: 2 (H, W)
  - લોઅરકેસ: 8 (e, l, l, o, o, r, l, d)

મેમરી ટ્રીક

``VOCAL વિશ્લેષણ'' - Vowels Or Consonants And Letter case

પ્રશ્ન 5(અ) [3 માર્ક્સ]

લિસ્ટ મા આપેલ બે એલીમેંટ ને સ્વેપ કરવા માટે પાયથોન કોડ લખો.

જવાબ

```
\#

def swap\_elements(lst, pos1, pos2):
    """                      """
    lst[pos1], lst[pos2] = lst[pos2], lst[pos1]
    return lst

\#
my\_list = [10, 20, 30, 40, 50]
print("    :", my\_list)

\#     1   3
result = swap\_elements(my\_list, 1, 3)
print("    1   3               :", result)
```

```
\#    :
\#      : [10, 20, 30, 40, 50]
\#      1   3                : [10, 40, 30, 20, 50]
```

મેમરી ટ્રીક

``STEP લોજિક'' - Swap Two Elements with Python લોજિક

પ્રશ્ન 5(બ) [4 માર્ક્સ]

આપેલ સ્ટ્રિંગમાં સબસ્ટ્રિંગ હાજર છે કે કેમ તે તપાસવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

```
\#

def check\_substring(main\_string, sub\_string):
    """                          """
    if sub\_string in main\_string:
        return True
    else:
        return False

\#
main\_string = input("           : ")
sub\_string = input("              : ")

\#
if check\_substring(main\_string, sub\_string):
    print(f"{}\{sub\_string\}{ }\{main\_string\}{       "})
else:
    print(f"{}\{sub\_string\}{ }\{main\_string\}{        "})
```

find() પદ્ધતિનો ઉપયોગ કરીને વૈકલ્પિક રીત:

```
def check\_substring\_find(main\_string, sub\_string):
    """           find         """
    position = main\_string.find(sub\_string)
    return position != {-}1  \#               True
```

મેમરી ટ્રીક

``FIND પદ્ધતિ'' - Find IN Directly with પદ્ધતિઓ

પ્રશ્ન 5(ક) [7 માર્ક્સ]

ટપલ ઓપરેશન, ફંકશન અને મેથડ સમજાવો.

જવાબ

ઓપરેશન/ફંકશન/મેથડ	વર્ણન	ઉદાહરણ	આઉટપુટ
બનાવટ	કૌંસ સાથે ટપલ બનાવવું	`t = (1, 2, 3)`	`(1, 2, 3)`
ઇન્ડેક્સિંગ	ટપલ એલિમેન્ટ્સ એક્સેસ કરવા	`t[1]`	`2`
સ્લાઇસિંગ	ટપલનો સબસેટ મેળવવો	`t[1:3]`	`(2, 3)`
કેટેનેશન	બે ટપલ જોડવા	`(1, 2) + (3, 4)`	`(1, 2, 3, 4)`

રિપિટેશન	ટપલ એલિમેન્ટ્સ રિપીટ કરવા	`(1, 2) * 2`	`(1, 2, 1, 2)`
મેમ્બરશિપ	એલિમેન્ટ છે કે નહીં તે તપાસવું	`3 in (1, 2, 3)`	`True`
len()	આઇટમ્સની સંખ્યા મેળવવી	`len((1, 2, 3))`	`3`
min()/max()	લઘુત્તમ/મહત્તમ મૂલ્ય શોધવું	`min((3, 1, 2))`	`1`
count()	મૂલ્યની સંખ્યા ગણવી	`(1, 2, 1).count(1)`	`2`
index()	મૂલ્યની પોઝિશન શોધવી	`(1, 2, 3).index(2)`	`1`
sorted()	ટપલમાંથી સૉર્ટેડ લિસ્ટ પાછી આપે	`sorted((3, 1, 2))`	`[1, 2, 3]`

કોડ ઉદાહરણ:

```
\#
my\_tuple = (3, 1, 4, 1, 5, 9)
print("    :", my\_tuple)

\#
print("       :", my\_tuple[0])
print("      :", my\_tuple[{-}1])
print("    (1:4):", my\_tuple[1:4])

\#
tuple2 = (2, 7)
combined = my\_tuple + tuple2
print("    :", combined)

repeated = tuple2 * 3
print("      :", repeated)

\#
print("  :", len(my\_tuple))
print("1     :", my\_tuple.count(1))
print("4      :", my\_tuple.index(4))
print("      :", min(my\_tuple))
print("      :", max(my\_tuple))
print("   :", sorted(my\_tuple))  \#

\#
a, b, c, *rest = my\_tuple
print("     :", a, b, c, rest)
```

મેમરી ટ્રીક

``ICONS'' - Immutable Collection Operations, Numbering, and Searching

પ્રશ્ન 5(અ) OR [3 માર્ક્સ]

લિસ્ટ મા આપેલ એલીમેંટ નો સરવાળો શોધવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

```
\#

def sum\_of\_list(numbers):
    """                          """
    total = 0
```

```
    for num in numbers:
        total += num
    return total

\#
num\_elements = int(input("              : "))
my\_list = []

\#
for i in range(num\_elements):
    element = float(input(f"    \{i+1\}      : "))
    my\_list.append(element)

\#
result1 = sum\_of\_list(my\_list)
print("                  :", result1)

\#   {-  sum()                  }
result2 = sum(my\_list)
print("  {-                :"}, result2)
```

મેમરી ટ્રીક

``SALT'' - Sum All List Together

પ્રશ્ન 5(બ) OR [4 માર્ક્સ]

સેટ ફંકશન અને ઓપરેશન દર્શાવવા માટે એક પ્રોગ્રામ લખો.

જવાબ

```
\#

\#
set1 = \{1, 2, 3, 4, 5\}
set2 = \{4, 5, 6, 7, 8\}

print("  1:", set1)
print("  2:", set2)

\#
print("{n}      :")
print("   :", set1 | set2)  \#     : set1.union(set2)
print("    :", set1 \& set2)  \#     : set1.intersection(set2)
print("    (set1{-set2):"}, set1 {-} set2)  \#     : set1.difference(set2)
print("        :", set1 \^{} set2)  \#     : set1.symmetric\_difference(set2)

\#
print("{n}     :")
set3 = set1.copy()
print(" 1     :", set3)

set3.add(6)
print("6       :", set3)

set3.remove(1)
print("1         :", set3)

set3.discard(10)  \#
print("10           :", set3)
```

```
popped = set3.pop()
print("         :", popped)
print("    :", set3)

set3.clear()
print("         :", set3)
```

મેમરી ટ્રીક

``COSI મેથડ્સ'' - Create, Operate, Search, Investigate with સેટ મેથડ્સ

પ્રશ્ન 5(ક) OR [7 માર્ક્સ]

ડિક્શનેરી ફંક્શન અને ઓપરેશન સમજાવવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

```
\#

\#
student = \{
    {name}: {John},
    {roll\_no}: 101,
    {marks}: 85,
    {subjects}: [{Python}, {Math}, {English}]
\}

print("      :", student)

\#
print("{n}           :")
print("  :", student[{name}])
print("   :", student[{marks}])

\# get()     {-            }
print("     (get     ):", student.get({roll\_no}))
print("    (get     ):", student.get({address}, {      }))  \#

\#
print("{n}         :")
student[{marks}] = 90
print("             :", student)

\#     {-         }
student[{address}] = {New York}
print("          :", student)

\#
print("{n}         :")
removed\_value = student.pop({address})
print("       :", removed\_value)
print("pop()   :", student)

\#
last\_item = student.popitem()
print("           :", last\_item)
print("popitem()   :", student)

\#
print("{n}       :")
```

```
print("  :", list(student.keys()))
print("  :", list(student.values()))
print("   :", list(student.items()))
```

મુખ્ય ઓપરેશન્સ:

- **એક્સેસ**: કી અથવા get() મેથડનો ઉપયોગ કરીને
- **મોડિફાય**: અસ્તિત્વમાં રહેલી કીને નવું મૂલ્ય આપવું
- **એડ**: નવી કીને મૂલ્ય આપવું
- **રિમૂવ**: pop(), popitem(), અથવા del સ્ટેટમેન્ટનો ઉપયોગ કરીને
- **ઇટરેટ**: કીઝ, વેલ્યુઝ, અથવા આઇટમ્સ દ્વારા

મેમરી ટ્રીક

``ACME ડિક્શનેરી'' - Access, Create, Modify, Extract from ડિક્શનેરી