પાયથોન પ્રોગ્રામિંગ (1323203) - ઉનાળુ 2024 સોલ્યુશન

Milav Dabgar

જૂન 18, 2024

પ્રશ્ન 1(અ) [3 ગુણ]

ફ્લોચાર્ટ અને અલ્ગોરિધમના મહત્વની યાદી આપો.

જવાબ

કોષ્ટક 1. ફ્લોચાર્ટ અને અલ્ગોરિધમનું મહત્વ

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

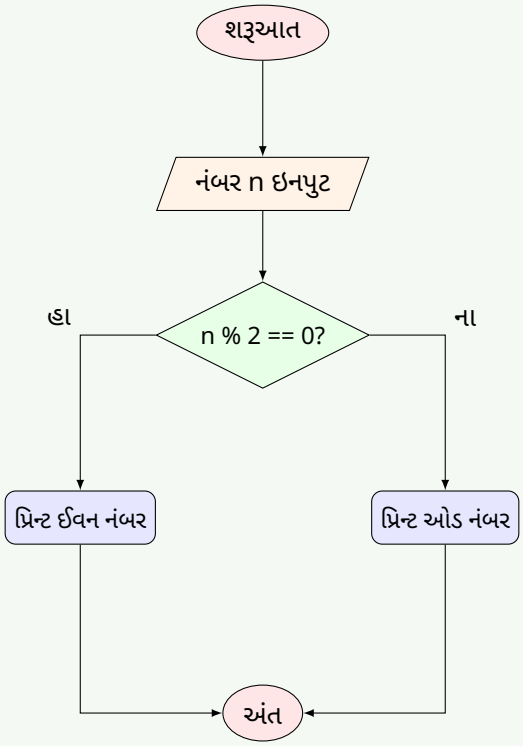
``VASE Decisions'' - Visualize, Analyze, Sequence, Execute

પ્રશ્ન 1(બ) [4 ગુણ]

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

જવાબ

આકૃતિ 1. ઈવન કે ઓડ સંખ્યા તપાસવા માટે ફ્લોચાર્ટ



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 ગુણ]

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

જવાબ

કોષ્ટક 2. લોજિકલ ઓપરેટરો

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:

```
# લોજકિલ AND ઉદાહરણ
age = 25
income = 50000
print("Loan eligibility:", age > 18 and income > 30000)  # True

# લોજકિલ OR ઉદાહરણ
has_credit_card = False
```

```
has_cash = True
print("Can purchase:", has_credit_card or has_cash)  # True

# લોજિકલ NOT ઉદાહરણ
is_holiday = False
print("Should work today:", not is_holiday)  # True
```

મેમરી ટ્રીક

``AON Clarity'' - And, Or, Not for logical clarity

OR

પ્રશ્ન 1(ક) [7 ગુણ]

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

જવાબ

```
# સાદા અને ચક્રવૃદ્ધિ વ્યાજની ગણતરી માટેનો પ્રોગ્રામ

# ઇનપુટ મૂલ્યો
principal = float(input("Enter principal amount: "))
rate = float(input("Enter rate of interest (in %): "))
time = float(input("Enter time period (in years): "))

# સાદા વ્યાજની ગણતરી
simple_interest = (principal * rate * time) / 100

# ચક્રવૃદ્ધિ વ્યાજની ગણતરી
compound_interest = principal * ((1 + rate/100) ** time - 1)

# પરિણામો દર્શાવો
print("Simple Interest:", round(simple_interest, 2))
print("Compound Interest:", round(compound_interest, 2))
```

રાસાયણિક સમીકરણ/સૂત્ર

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

મેમરી ટ્રીક

``PRT Money Grows'' - Principal, Rate, Time make money grow

પ્રશ્ન 2(અ) [3 ગુણ]

આપેલ ત્રણ નંબરોમાંથી ન્યૂનતમ સંખ્યા શોધવા માટે પાયથોન પ્રોગ્રામ બનાવો.

જવાબ

```
# ત્રણ નંબરોમાંથી ન્યૂનતમ શોધવાનો પ્રોગ્રામ

# ત્રણ નંબર ઇનપુટ લો
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
num3 = float(input("Enter third number: "))

# બિલ્ટઇન- min() ફંક્શનનો ઉપયોગ કરીને ન્યૂનતમ શોધો
minimum = min(num1, num2, num3)

# પરિણામ દર્શાવો
print("Minimum number is:", minimum)
```

મેમરી ટ્રીક

``MIN Finds Least'' - Minimum Is Numerically Found with Least

પ્રશ્ન 2(બ) [4 ગુણ]

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

જવાબ

કોષ્ટક 3. સ્યુડોકોડની વ્યાખ્યા

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 number is: ", largest
END
```

મેમરી ટ્રીક

``PIE Writing'' - Program Ideas Expressed in simple writing

પ્રશ્ન 2(ક) [7 ગુણ]

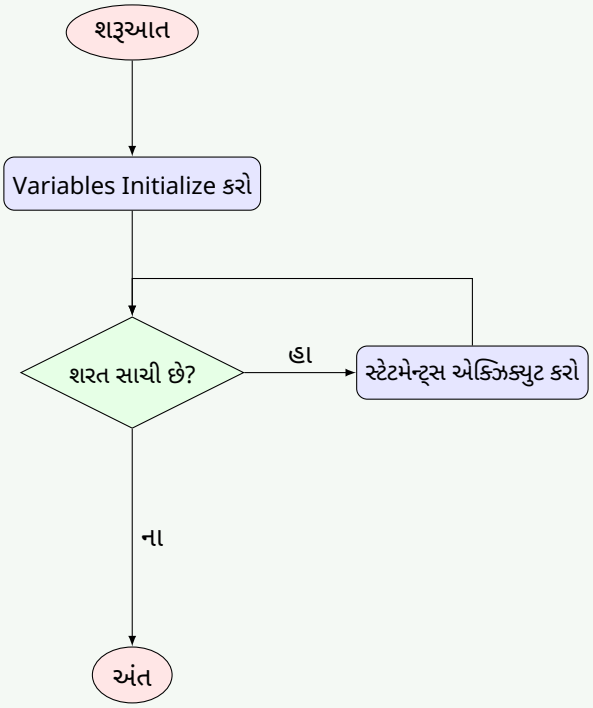
પાયથોનમાં વાઈલ લૂપને તેના સિન્ટેક્સ, ફ્લોચાર્ટ અને પાયથોન કોડના ઉદાહરણ સાથે સમજાવો.

જવાબ

સિન્ટેક્સ:

```
while condition:
    # code to be executed
```

આકૃતિ 2. વાઈલ લૂપનો ફ્લોચાર્ટ



કોડ ઉદાહરણ:

```
# પ્રથમ 5 કુદરતી સંખ્યાઓ while લૂપનો ઉપયોગ કરીને પ્રિન્ટ કરો
count = 1

while count <= 5:
    print(count)
    count += 1  # કાઉન્ટર વધારો

# આઉટપુટ:
# 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 Loop'' - Initialize, Update, Test for Exit

OR

પ્રશ્ન 2(અ) [3 ગુણ]

પાયથોનમાં કન્ટિન્યુ સ્ટેટમેન્ટનું ટૂંકમાં વર્ણન કરો.

જવાબ

કોષ્ટક 4. કન્ટિન્યુ સ્ટેટમેન્ટ

વર્ણન
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:

```
# બેકી સંખ્યાઓ પ્રિન્ટ કરવાનું છોડી દો
for i in range(1, 6):
    if i % 2 == 0:
        continue
    print(i)  # માત્ર 1, 3, 5 પ્રિન્ટ થાય
```

મેમરી ટ્રીક

``SKIP Ahead'' - Skip Keeping Iteration Process

OR

પ્રશ્ન 2(બ) [4 ગુણ]

નીચેના કોડનું આઉટપુટ શું હશે?

જવાબ

કોડ:

```
x=8
y=2
print (x*y)
print (x ** y)
print (x % y)
print(x>y)
```

કોષ્ટક 5. આઉટપુટ વિશ્લેષણ

ઓપરેશન	પરિણામ	સમજૂતી
x*y	16	ગુણાકાર: $8 \times 2 = 16$
x**y	64	પાવર: $8^2 = 64$
x%y	0	મોડ્યુલો (શેષ): $8 \div 2 = 4$ શેષ 0
x>y	True	તુલના: $8 > 2$ સાચું છે

મેમરી ટ્રીક

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

OR

પ્રશ્ન 2(ક) [7 ગુણ]

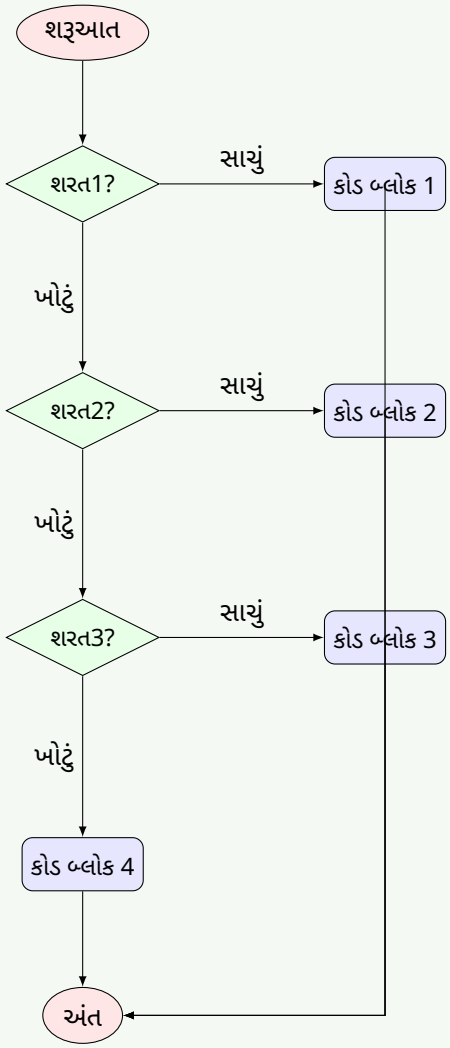
પાયથોનમાં ઈફ-ઈએલઈએફ-એલ્સ લેડરને તેના સિન્ટેક્સ, ફ્લોચાર્ટ અને પાયથોન કોડના ઉદાહરણ સાથે સમજાવો.

જવાબ

સિન્ટેક્સ:

```
if condition1:
    # code block 1
elif condition2:
    # code block 2
elif condition3:
    # code block 3
else:
    # code block 4
```

આકૃતિ 3. ઈફ-ઈએલઈએફ-એલ્સ લેડરનો ફ્લોચાર્ટ



કોડ ઉદાહરણ:

```python
# માર્ક્સના આધારે ગ્રેડની ગણતરી
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:", grade)  # આઉટપુટ: 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 Logic'' - Sequential Evaluation with Exclusive Path

પ્રશ્ન 3(અ) [3 ગુણ]

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

જવાબ

```
# 1 થી 20 વચ્ચેની એકી સંખ્યાઓ છાપવાનો પ્રોગ્રામ

# range અને step સાથે for લૂપનો ઉપયોગ
for number in range(1, 21, 2):
    print(number, end=" ")

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

વૈકલ્પિક અભિગમ:

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

મેમરી ટ્રીક

``STEO'' - Skip Two, Extract Odds

પ્રશ્ન 3(બ) [4 ગુણ]

નેસ્ટેડ ઈફ સ્ટેટમેન્ટને સંક્ષિપ્તમાં સમજાવો.

જવાબ

કોષ્ટક 6. નેસ્ટેડ ઈફ સ્ટેટમેન્ટ

વર્ણન
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("પુખ્ત")
    if income > 30000:
        print("ક્રેડિટ કાર્ડ માટે પાત્ર")
    else:
```

```
    print("ક્રેડિટ કાર્ડ માટે અપાત્ર")
else:
    print("સગીર")
```

મેમરી ટ્રીક

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

પ્રશ્ન 3(ક) [7 ગુણ]

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

જવાબ

```
# આર્મસ્ટ્રોંગ નંબર અથવા પેલિન્ડ્રોમ તપાસવાનો પ્રોગ્રામ

def check_number(num):
    # આર્મસ્ટ્રોંગ નંબર તપાસો
    temp = num
    digits = len(str(num))
    sum = 0

    while temp > 0:
        digit = temp % 10
        sum += digit ** digits
        temp //= 10

    is_armstrong = (sum == num)

    # પેલિન્ડ્રોમ તપાસો
    is_palindrome = (str(num) == str(num)[::-1])

    # પરિણામો પાછા આપો
    return is_armstrong, is_palindrome

# વપરાશકર્તા પાસેથી ઇનપુટ લો
number = int(input("એક નંબર દાખલ કરો: "))

# ફંક્શન કૉલ કરો અને પરિણામો દર્શાવો
armstrong, palindrome = check_number(number)

if armstrong:
    print(number, "is an Armstrong number")
else:
    print(number, "is not an Armstrong number")

if palindrome:
    print(number, "is a Palindrome")
else:
    print(number, "is not a Palindrome")
```

ઉદાહરણો:

- **આર્મસ્ટ્રોંગ**: 153 ($1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$)
- **પેલિન્ડ્રોમ**: 121 (આગળ અને પાછળ સમાન)

મેમરી ટ્રીક

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

OR

પ્રશ્ન 3(અ) [3 ગુણ]

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

જવાબ

```
# 1 થી 100 સુધીની સંખ્યાઓનો સરવાળો શોધવાનો પ્રોગ્રામ

# પદ્ધતિ 1: લૂપનો ઉપયોગ
total = 0
for num in range(1, 101):
    total += num
print("Sum using loop:", total)

# પદ્ધતિ 2: સૂત્ર n(n+1)/2 નો ઉપયોગ
n = 100
sum_formula = n * (n + 1) // 2
print("Sum using formula:", sum_formula)

# આઉટપુટ:
# Sum using loop: 5050
# Sum using formula: 5050
```

મેમરી ટ્રીક

``SUM Formula'' - Sum Using Mathematical Formula

OR

પ્રશ્ન 3(બ) [4 ગુણ]

નીચેની પેટર્ન છાપવા માટે પાયથોન પ્રોગ્રામ લખો.

જવાબ

પેટર્ન:

```
1
2 3
4 5 6
7 8 9 10
```

કોડ:

```
# સંખ્યા પેટર્ન છાપવાનો પ્રોગ્રામ

num = 1
for i in range(1, 5):  # 4 પંક્તિઓ
    for j in range(i):  # પંક્તિ નંબર જેટલા કોલમ
```

```
        print(num, end=" ")
        num += 1
    print()  # દરેક પંક્તિ પછી નવી લાઈન
```

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

OR

પ્રશ્ન 3(ક) [7 ગુણ]

ફંક્શનનો ઉપયોગ કરીને પ્રોગ્રામ લખો જે દાખલ કરેલ નંબરને ઉલટાવે

જવાબ

```
# દાખલ કરેલ મૂલ્યને ઉલટાવવા માટે ફંક્શન ઉપયોગ કરતો પ્રોગ્રામ

def reverse_number(num):
    """સંખ્યાને ઉલટાવવા માટેનું ફંક્શન"""
    return int(str(num)[::-1])

def reverse_string(text):
    """સ્ટ્રિંગને ઉલટાવવા માટેનું ફંક્શન"""
    return text[::-1]

# મુખ્ય પ્રોગ્રામ
def main():
    choice = input("What do you want to reverse? (n for number, s for string): ")

    if choice.lower() == 'n':
        num = int(input("Enter a number: "))
        print("Reversed number:", reverse_number(num))
    elif choice.lower() == 's':
        text = input("Enter a string: ")
        print("Reversed string:", reverse_string(text))
    else:
        print("Invalid choice!")

# મુખ્ય ફંક્શન કૉલ કરો
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 Digits'' - Function Logic Inverts Position of Digits

પ્રશ્ન 4(અ) [3 ગુણ]

યોગ્ય પાયથોન કોડ ઉદાહરણ સાથે પાયથોન મેથ મોડ્યુલનું વર્ણન કરો.

જવાબ

કોષ્ટક 7. પાયથોન મેથ મોડ્યુલ

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

# સ્થિરાંકો
print("Value of pi:", math.pi)  # 3.141592653589793
print("Value of e:", math.e)    # 2.718281828459045

# મૂળભૂત ગાણિતિક ફંક્શન્સ
print("16 નો વર્ગમૂળ:", math.sqrt(16))  # 4.0
print("5 ની ઘાત 3:", math.pow(5, 3))  # 125.0

# ત્રિકોણમિતિય ફંક્શન્સ રેડિયન()
print("90° નો સાઇન:", math.sin(math.pi/2))  # 1.0
print("0° નો કોસાઇન:", math.cos(0))  # 1.0

# લોગરિધમિક ફંક્શન્સ
print("100 નો આધાર 10 લોગ:", math.log10(100))  # 2.0
print("e નો નેચરલ લોગ:", math.log(math.e))  # 1.0
```

મેમરી ટ્રીક

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

પ્રશ્ન 4(બ) [4 ગુણ]

વેરીએબલના સ્કોપને સમજાવતો પાયથોન પ્રોગ્રામ લખો.

જવાબ

```
# પાયથોનમાં વેરીએબલ સ્કોપ દર્શાવતો પ્રોગ્રામ

# ગ્લોબલ વેરીએબલ
global_var = "I am global"

```

```python
def demonstration():
    # લોકલ વેરીએબલ
    local_var = "I am local"

    # ગ્લોબલ વેરીએબલ એક્સેસ કરવું
    print("Inside function - Global variable:", global_var)

    # લોકલ વેરીએબલ એક્સેસ કરવું
    print("Inside function - Local variable:", local_var)

    # ગ્લોબલ નામ ધરાવતું લોકલ વેરીએબલ બનાવવું
    global_var = "I am local with global name"
    print("Inside function - Shadowed global:", global_var)

# ફંક્શન કૉલ
demonstration()

# ગ્લોબલ વેરીએબલ એક્સેસ કરવું
print("Outside function - Global variable:", global_var)

# લોકલ વેરીએબલ એક્સેસ કરવાનો પ્રયાસ ભૂલ ઉત્પન્ન કરશે
# print("Outside function - Local variable:", local_var)  # Error!
```

મેમરી ટ્રીક

``GLOVES'' - Global Local Variable Encapsulation System

પ્રશ્ન 4(ક) [7 ગુણ]

લિસ્ટ પદ્ધતિઓ અને તેના બિલ્ટ-ઇન કાર્યો સમજાવો

જવાબ

કોષ્ટક 8. લિસ્ટ પદ્ધતિઓ અને ફંક્શન્સ

પદ્ધતિ	વર્ણન	ઉદાહરણ	આઉટપુટ
append()	અંતે એલિમેન્ટ ઉમેરે છે	l=['a']; l.append('b')	['a', 'b']
insert()	ચોક્કસ પોઝિશન પર એલિમેન્ટ ઉમેરે	l=[1,3]; l.insert(1,2)	[1, 2, 3]
remove()	ચોક્કસ આઈટમ દૂર કરે	l=['r','b']; l.remove('r')	['b']
pop()	ચોક્કસ ઇન્ડેક્સ પર આઈટમ દૂર કરે	l=['a','b']; l.pop(1)	'b'
clear()	બધા એલિમેન્ટ્સ દૂર કરે	l=[1,2]; l.clear()	[]
len()	એલિમેન્ટ્સની સંખ્યા પાછી આપે	len([1, 2, 3])	3
sorted()	સૉર્ટેડ લિસ્ટ પાછી આપે	sorted([3, 1, 2])	[1, 2, 3]
max()	મહત્તમ મૂલ્ય પાછું આપે	max([5, 10, 3])	10

કોડ ઉદાહરણ:

```python
# લસ્ટ બનાવવી
my_list = [3, 1, 4, 1, 5]
my_list.append(9)        # અંતે ઉમેરો
my_list.insert(2, 7)     # ઇન્ડેક્સ 2 પર ઉમેરો
my_list.remove(1)        # પ્રથમ 1 દૂર કરો
popped = my_list.pop()   # છેલ્લું એલમિન્ટ દૂર કરો

print("લંબાઈ:", len(my_list))
```

```
print("સૉર્ટેડ:", sorted(my_list))
print("સરવાળો:", sum(my_list))
print("1 ની સંખ્યા:", my_list.count(1))
```

મેમરી ટ્રીક

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

OR

પ્રશ્ન 4(અ) [3 ગુણ]

પાયથોન સ્ટાન્ડર્ડ લાઇબ્રેરી ગાણિતિક કાર્યોની સૂચિ બનાવો.

જવાબ

કોષ્ટક 9. ગાણિતિક ફંક્શન્સ

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) → 5
min()	સૌથી નાની આઈટમ પાછી આપે	min(1, 5) → 1
sum()	ઇટરેબલની આઈટમ્સનો સરવાળો કરે	sum([1, 2]) → 3
pow()	x ને y ની ઘાત પાછી આપે	pow(2, 3) → 8

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 Calculations'' - Standard Mathematical Arithmetic Routines and Tools

OR

પ્રશ્ન 4(બ) [4 ગુણ]

પાયથોનમાં બિલ્ટ ઇન ફંક્શન સમજાવો.

જવાબ

કોષ્ટક 10. બિલ્ટ-ઇન ફંક્શન્સ

વર્ણન
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:

```
# ટાઇપ કન્વર્ઝન
print(int("10"))    # 10
print(str(10))      # "10"

# ગાણિતિક ફંક્શન્સ
print(abs(-7))      # 7
print(max(5, 10, 3))  # 10

# કલેક્શન પ્રોસેસિંગ
print(len("hello"))   # 5
print(sorted([3,1,2])) # [1, 2, 3]
```

મેમરી ટ્રીક

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

OR

પ્રશ્ન 4(ક) [7 ગુણ]

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

જવાબ

```
# સ્ટ્રિંગમાં સ્વરો, વ્યંજન, અપરકેસ, લોઅરકેસ ગણતરી માટેનો પ્રોગ્રામ

def analyze_string(text):
    # કાઉન્ટર્સ ઇનિશિયલાઇઝ કરો
    vowels = 0
    consonants = 0
    uppercase = 0
    lowercase = 0

    # સ્વરો ડિફાઇન કરો
    vowel_set = {'a', 'e', 'i', 'o', 'u'}

    # દરેક અક્ષરનું વિશ્લેષણ
    for char in text:
        # તપાસો કે શું અક્ષર છે
        if char.isalpha():
            # કેસ તપાસો
            if char.isupper():
                uppercase += 1
            else:
                lowercase += 1

            # તપાસો કે સ્વર છે કેસસેન્સિટિવિ(-)
```

```python
        if char.lower() in vowel_set:
            vowels += 1
        else:
            consonants += 1

    # પરિણામો પાછા આપો
    return vowels, consonants, uppercase, lowercase

# ઇનપુટ લો
text = input("Enter a string: ")

# ગણતરી મેળવો
vowels, consonants, uppercase, lowercase = analyze_string(text)

# પરિણામો દર્શાવો
print("Number of vowels:", vowels)
print("Number of consonants:", consonants)
print("Number of uppercase characters:", uppercase)
print("Number of lowercase characters:", lowercase)
```

મેમરી ટ્રીક

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

પ્રશ્ન 5(અ) [3 ગુણ]

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

જવાબ

```python
# લિસ્ટમાં બે એલિમેન્ટ્સ સ્વેપ કરવાનો પ્રોગ્રામ

def swap_elements(lst, pos1, pos2):
    """લિસ્ટમાં બે એલિમેન્ટ્સ સ્વેપ કરવા માટેનું ફંક્શન"""
    lst[pos1], lst[pos2] = lst[pos2], lst[pos1]
    return lst

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

# પોઝિશન 1 અને 3 પરના એલિમેન્ટ્સ સ્વેપ કરો
result = swap_elements(my_list, 1, 3)
print("પોઝિશન 1 અને 3 પરના એલિમેન્ટ્સ સ્વેપ કર્યા પછી:", result)

# આઉટપુટ:
# મૂળ લિસ્ટ: [10, 20, 30, 40, 50]
# પોઝિશન 1 અને 3 પરના એલિમેન્ટ્સ સ્વેપ કર્યા પછી: [10, 40, 30, 20, 50]
```

મેમરી ટ્રીક

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

પ્રશ્ન 5(બ) [4 ગુણ]

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

જવાબ

```
# સ્ટ્રિંગમાં સબસ્ટ્રિંગની હાજરી તપાસવાનો પ્રોગ્રામ

def check_substring(main_string, sub_string):
    """સ્ટ્રિંગમાં સબસ્ટ્રિંગની હાજરી તપાસવા માટેનું ફંક્શન"""
    if sub_string in main_string:
        return True
    else:
        return False

# વપરાશકર્તા પાસેથી ઇનપુટ લો
main_string = input("Enter main string: ")
sub_string = input("Enter substring: ")

# તપાસો અને પરિણામ દર્શાવો
if check_substring(main_string, sub_string):
    print(f"'{sub_string}' is present in '{main_string}'")
else:
    print(f"'{sub_string}' is not present in '{main_string}'")
```

મેમરી ટ્રીક

``FIND Method'' - Find IN Directly with Methods

પ્રશ્ન 5(ક) [7 ગુણ]

ટપલ ઓપરેશન, ફંકશન અને મેથડ સમજાવો.

જવાબ

કોષ્ટક 11. ટપલ ઓપરેશન્સ

ઓપરેશન	વર્ણન	ઉદાહરણ	પરિણામ
બનાવટ	કૌંસ સાથે બનાવવું	t=(1,2)	(1, 2)
ઇન્ડેક્સિંગ	એલિમેન્ટ્સ એક્સેસ કરવા	t[1]	2
સ્લાઇસિંગ	સબસેટ મેળવવો	t[0:1]	(1,)
કેટેનેશન	બે ટપલ જોડવા	(1)+(2)	(1, 2)
રિપિટેશન	રિપીટ કરવા	(1)*2	(1, 1)
મેમ્બરશિપ	અસ્તિત્વ તપાસવું	1 in t	True
len()	આઇટમ્સની સંખ્યા	len(t)	2
count()	સંખ્યા ગણવી	t.count(1)	1
index()	પોઝિશન શોધવી	t.index(2)	1

કોડ ઉદાહરણ:

```
my_tuple = (3, 1, 4, 1, 5, 9)
print("પ્રથમ:", my_tuple[0])
print("સ્લાઇસ:", my_tuple[1:4])
print("1 ની સંખ્યા:", my_tuple.count(1))
```

```
print("4 નો ઇન્ડેક્સ:", my_tuple.index(4))
a, b, c, *rest = my_tuple # અનપેકિંગ
```

મેમરી ટ્રીક

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

OR

પ્રશ્ન 5(અ) [3 ગુણ]

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

જવાબ

```
# લિસ્ટના એલિમેન્ટ્સનો સરવાળો શોધવાનો પ્રોગ્રામ

def sum_of_list(numbers):
    """લિસ્ટના બધા એલિમેન્ટ્સનો સરવાળો શોધવા માટેનું ફંક્શન"""
    total = 0
    for num in numbers:
        total += num
    return total

# ઉદાહરણ
num_elements = int(input("Enter number of elements: "))
my_list = []

# એલિમેન્ટ્સ મેળવો
for i in range(num_elements):
    element = float(input(f"Enter element {i+1}: "))
    my_list.append(element)

# સરવાળો ગણો
result1 = sum_of_list(my_list)
print("Sum using custom function:", result1)

# બિલ્ટઇન- sum()
result2 = sum(my_list)
print("Sum using built-in function:", result2)
```

મેમરી ટ્રીક

``SALT'' - Sum All List Together

OR

પ્રશ્ન 5(બ) [4 ગુણ]

સેટ ફંકશન અને ઓપરેશન દર્શાવવા માટે એક પ્રોગ્રામ લખો.

જવાબ

```python
# સેટ ફંક્શન અને ઓપરેશન્સ દર્શાવતો પ્રોગ્રામ

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

# સેટ ઓપરેશન્સ
print("યુનિયન:", set1 | set2)
print("ઇન્ટરસેક્શન:", set1 & set2)
print("ડિફરન્સ:", set1 - set2)
print("સમિટ્રિક ડિફરન્સ:", set1 ^ set2)

# સેટ મેથડ્સ
set3 = set1.copy()
set3.add(6)
set3.remove(1)
set3.discard(10) # ભૂલ નહીં
popped = set3.pop()
set3.clear()
```

મેમરી ટ્રીક

``COSI Methods'' - Create, Operate, Search, Investigate with Set Methods

OR

પ્રશ્ન 5(ક) [7 ગુણ]

ડિક્શનેરી ફંક્શન અને ઓપરેશન સમજાવવા માટે પાયથોન પ્રોગામ લખો.

જવાબ

```python
# ડિક્શનેરી ફંક્શન અને ઓપરેશન્સ દર્શાવતો પ્રોગ્રામ

# ડિક્શનેરી બનાવવી
student = {
    'name': 'John',
    'roll_no': 101,
    'marks': 85
}

# એક્સેસ
print("Name:", student['name'])
print("Roll No:", student.get('roll_no'))

# સુધારવું અને ઉમેરવું
student['marks'] = 90
student['address'] = 'New York'

# દૂર કરવું
removed = student.pop('address')
last_item = student.popitem()

# મેથડ્સ
print("Keys:", list(student.keys()))
print("Values:", list(student.values()))
```

```
print("Items:", list(student.items()))

# ક્લિયર
student.clear()
```

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

મેમરી ટ્રીક

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