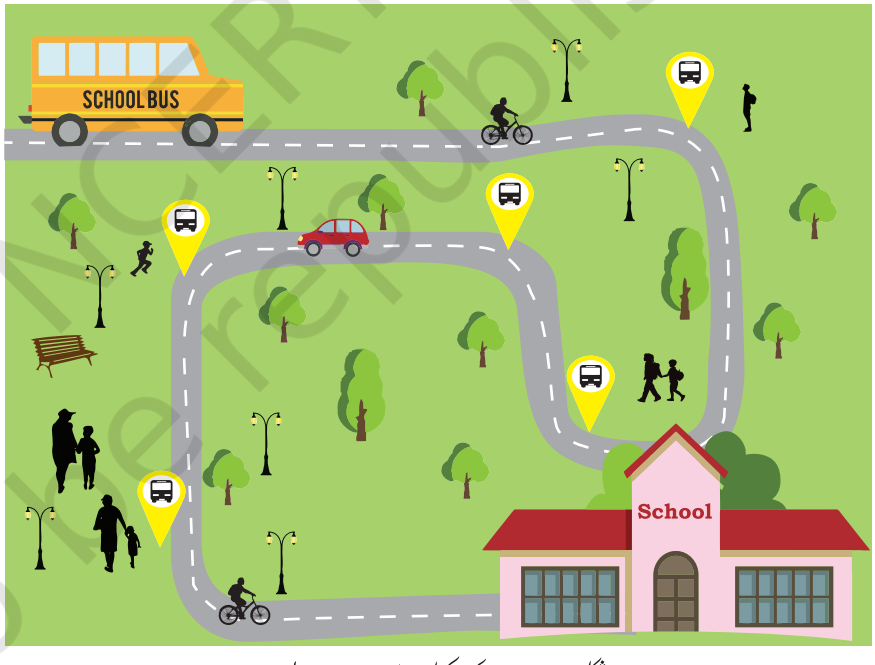
5196CH06

باب-6

کنٹرول کا تسلسل
(FLOW OF CONTROL)

6.1 تعارف

شکل 6.1 میں، ہم دیکھ سکتے ہیں کہ ایک بس بچوں کو اسکول لے جا رہی ہے۔ اسکول پہنچنے کا واحد ایک ہی راستہ ہے۔ ڈرائیور کے پاس اس کے سوا کوئی اور متبادل نہیں ہے کہ اسکول پہنچنے کے لیے سڑک پر میل کے ایک پتھر کے بعد دوسرے پتھر تک مسلسل چلتے رہنا ہے۔ ہم باب 5 میں پڑھ چکے ہیں کہ تسلسل کا تصور یہی ہے جس میں پائتھن پروگرام کی ابتدا سے اختتام تک ایک بیان کے بعد دوسرے بیان پر عمل درآمد کرتا ہے۔ ہم ابھی تک اسی قسم کے پروگرام تحریر کرتے رہے ہیں۔



شکل 6.1 : بچوں کو اسکول لے جاتے ہوئے بس

ایک پروگرام 6-1 پر غور کیجیے جو سلسلے وار ایگزیکیوٹ ہوتا ہے، یعنی پروگرام کے بیانات ہر عمل درآمد اسی ترتیب میں ہوتا ہے جس ترتیب میں انھیں تحریر کیا گیا ہے۔

پروگرام میں بیانات پر عمل درآمد جس ترتیب میں ہوتا ہے اسے کنٹرول کا تسلسل (Flow of Control) کہتے ہیں۔ کنٹرول کے تسلسل کا نفاذ کنٹرول ساختوں (Control Structures) کی مدد سے کیا جاسکتا ہے۔ پائتھن میں دو قسم کی کنٹرول ساختوں کا استعمال کیا جاتا ہے یعنی ایک انتخاب (Selection) و دوسری تکرار (Repetition)۔

''اگر کسی کوڈ کو مناسب طور پر انڈینٹ نہیں کیا گیا ہے تو کیا آپ کو اس سے نفرت نہ ہوگی۔ اس [انڈینٹ] کو اگر سنٹیکس کا حصہ بنایا جائے تو یہ اس بات کا ضامن ہوگا کہ مکمل کوڈ کو صحیح طور پر انڈینٹ کیا گیا ہے''

— جی وین روسم
(G.van Rossum)

اس باب میں

- کنٹرول کے تسلسل کا تعارف
- انتخاب
- انڈینٹیشن
- تکرار
- منقطع اور جاری بیان
- نیسٹیڈ لوپ

پروگرام 6-1 دو اعداد کے فرق کو پرنٹ کرنے کے لیے پروگرام

```
#Program 6-1
#Program to print the difference of two input numbers
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
diff = num1 - num2
print("The difference of",num1,"and",num2,"is",diff)
```

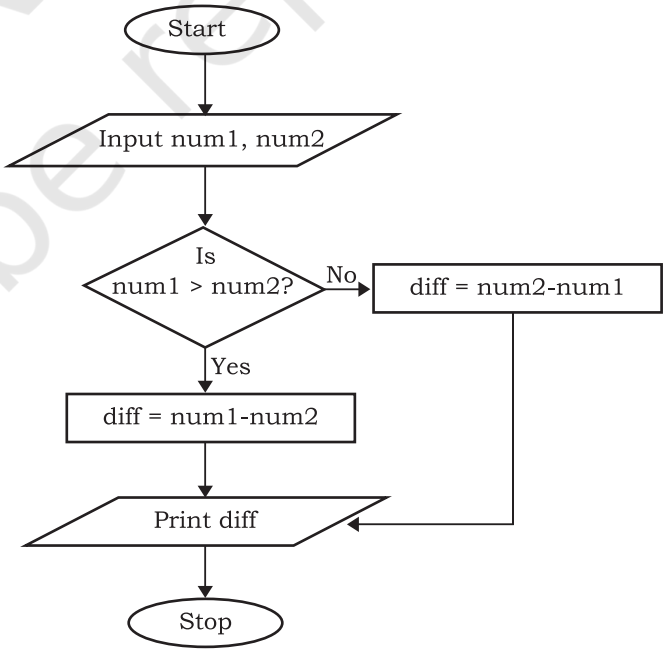
نتیجہ:

```
Enter first number 5
Enter second number 7
The difference of 5 and 7 is -2
```

6.2 انتخاب (SELECTION)

اب فرض کیجیے کہ ہمارے پاس پین خریدنے کے لیے 10 روپے ہیں۔ اسٹیشنری کی دکان پر پین کی ایسی بہت سی اقسام ہیں جن میں سے ہر ایک کی قیمت 10 روپے ہے۔ یہاں ہمیں یہ فیصلہ کرنا ہے کہ کون سا پین خریدنا ہے۔ اسی طرح، جب ہم ایک مقام سے دوسرے مقام تک پہنچنے کے لیے ڈیجیٹل نقشے کی سمتی خدمات سے استفادہ کرتے ہیں تو ہم یہ دیکھتے ہیں کہ بعض اوقات یہ ایک سے زیادہ راستے متبادل کے طور پر پیش کرتا ہے، مثلاً سب سے کم بھیڑ والا راستہ، مختصر ترین فاصلے پر مبنی راستہ وغیرہ۔ ہم اپنی ترجیحات کے مطابق راستے کے تعین کا فیصلہ کرتے ہیں۔ فیصلے کا تعلق دو یا دو سے زیادہ ممکنہ متبادلات میں سے کسی ایک کے انتخاب سے ہے۔ پروگرامنگ میں if..else بیان کی مدد سے فیصلہ سازی یا انتخاب کے اسی تصور کو بروئے کار لایا جاتا ہے۔

اب فرض کیجیے کہ پروگرام 6-1 میں ہم دو اعداد num1 اور num2 کے مثبت فرق کو ظاہر کرنا چاہتے ہیں۔ اس کے لیے ہمیں اپنے طریقہ کار میں تبدیلی لانی ہوگی۔ شکل 6.2 میں دیے گئے فلوچارٹ کو دیکھیے جو چھوٹے عدد کو بڑے عدد میں سے گھٹاتا ہے جس کے نتیجے میں ہمیشہ ہمیں ہی مثبت فرق حاصل ہوتا ہے۔ یہ انتخاب دو اعداد num1 اور num2 کے لیے داخل کی گئی قدروں پر مبنی ہے۔



شکل 6.2: فلوچارٹ فیصلہ سازی کو ظاہر کرتا ہے۔

نوٹ

if بیان کا سنٹیکس مندرجہ ذیل ہے:

```
if condition:
     statement(s)
```

مندرجہ ذیل مثال میں اگر استعمال کنندہ کے ذریعے داخل کی جانے والی عمر 18 سے زیادہ ہے تو پرنٹ کیجیے کہ استعمال کنندہ ووٹ دینے کے اہل ہے۔ اگر شرط پوری ہو جاتی ہے تو انڈینٹ اسٹیٹمنٹ پر عمل درآمد کیا جاتا ہے۔ انڈینٹیشن اس بات کی طرف اشارہ کرتا ہے کہ اس کا ایگزیکیوشن شرط کے تابع ہے۔ if بیان کے تحت بلاک کے طور پر ظاہر ہونے والے بیانات کی تعداد کی کوئی حد نہیں ہے۔

مثال 6.1

```
age = int(input("Enter your age "))
if age >= 18:
    print("Eligible to vote")
```

بیان if..else جو کہ if بیان کی ہی ایک قسم ہے، ہمیں دو متبادل بیانات (paths) لکھنے کی سہولت فراہم کرتا ہے اور کنٹرول کنڈیشن اس بات کا تعین کرتی ہے کہ کون سے بیان پر عمل درآمد کیا جائے۔ if..else بیان کے لیے سنٹیکس مندرجہ ذیل ہے۔

```
if condition:
    statement(s)
else:
    statement(s)
```

آیئے اب ووٹنگ سے متعلق مثال میں یہ ترمیم کرتے ہیں کہ اگر استعمال کنندہ کے ذریعے داخل کی گئی عمر 18 سال سے زیادہ ہے تو یہ ظاہر کیا جائے کہ استعمال کنندہ ووٹ دینے کے اہل ہے بصورت دیگر یہ ظاہر کیا جائے کہ استعمال کنندہ ووٹ دینے کے اہل نہیں ہے۔

```
age = int(input("Enter your age: "))
if age >= 18:
     print("Eligible to vote")
else:
     print("Not eligible to vote")
```

آیئے اب اسی تصور کو استعمال کرتے ہوئے پروگرام 1-6 میں اس طرح ترمیم کرتے ہیں کہ یہ آؤٹ پُٹ کے طور پر ہمیشہ ہی مثبت فرق ظاہر کرے۔ شکل 6.2 میں دیے ہوئے فلوچارٹ میں یہ بات واضح ہے کہ ہمیں اس بات کا فیصلہ کرنا ہے کہ آیا num1 > num2 ہے یا نہیں اور اسی کے مطابق عمل انجام دینا ہے۔

چوں کہ num2 کے مقابلے num1 بڑا ہو سکتا ہے یا num1 کے مقابلے num2 بڑا ہو سکتا ہے لہٰذا، ہمیں دو بلاک متعین کرنے ہوں گے جیسا کہ پروگرام 2-6 میں دکھایا گیا ہے۔

کئی مرتبہ ایسی صورت حال سے بھی سامنا ہوتا ہے جس میں متعدد شرائط کی پابندی شامل ہوتی ہے اور

اس وجہ سے کئی متبادلات تحریر کرنے پڑ سکتے ہیں۔اس قسم کے معاملات میں ہم elif(if..elif کا مطلب ہے else..if) کا استعمال کر کے شرائط کو باہم منسلک کر سکتے ہیں۔

پروگرام 6-2 دو اعداد کے مثبت فرق کو پرنٹ کرنے کے لیے پروگرام

```
#Program 6-2
#Program to print the positive difference of two numbers
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
if num1 > num2:
     diff = num1 - num2
else:
     diff = num2 - num1
print("The difference of",num1,"and",num2,"is",diff)
```

نتیجہ:

```
Enter first number: 5
Enter second number: 6
The difference of 5 and 6 is 1
```

elif کا استعمال کرتے ہوئے انتخابی ساخت (Selection structure) کے لیے سنٹیکس ذیل میں دیا گیا ہے۔

```
if condition:
     statement(s)
elif condition:
     statement(s)
elif condition:
     statement(s)
else:
     statement(s)
```

مثال 6.2 اس بات کی جانچ کرنا کہ آیا کوئی عدد مثبت، منفی یا صفر ہے۔

```
number = int(input("Enter a number: ")
if number > 0:
     print("Number is positive")
elif number < 0:
     print("Number is negative")
else:
     print("Number is zero")
```

مثال 6.3 چوراہے پر سگنل کے رنگ کے مطابق مناسب پیغام کو ظاہر کرنا

```
signal = input("Enter the colour: ")
if signal == "red" or signal == "RED":
     print("STOP")
elif signal == "orange" or signal ==
"ORANGE":
     print("Be Slow")
elif signal == "green" or signal == "GREEN":
     print("Go!")
```

elif کی تعداد کا انحصار زیر جانچ شرائط کی تعداد پر ہوتا ہے۔ اگر پہلی شرط پوری نہیں ہوتی ہے تو اگلی شرط کی جانچ کی جاتی ہے اور اسی طرح یہ عمل آگے جاری رہتا ہے۔ اگر کوئی ایک شرط پوری ہو جاتی ہے تو متعلقہ انڈینٹ والا بلاک ایگزیکیوٹ ہو جاتا ہے اور if بیان کا اختتام ہو جاتا ہے۔

آیئے دو اعداد پر ریاضی کے بنیادی عملوں کو انجام دینے کے مقصد سے ایک سادہ کیلکولیٹر کی تشکیل کے لیے پروگرام تحریر کریں۔ پروگرام میں مندرجہ ذیل باتیں شامل ہونی چاہئیں۔

- استعمال کنندہ سے دو اعداد حاصل کرنا
- استعمال کنندہ سے کہا جائے کہ وہ ان میں سے کوئی ایک آپریٹر (+، -، *، /) داخل کرے۔ اگر استعمال کنندہ ان کے علاوہ کچھ اور داخل کرتا ہے تو غلطی کی طرف اشارہ کرنے والا پیغام ظاہر ہو جائے۔
- آپریٹر "-" کے معاملے میں صرف مثبت فرق کو ظاہر کرنا۔
- اگر استعمال کنندہ دوسرا عدد 0 اور آپریٹر "/" داخل کرتا ہے تو یہ پیغام ظاہر کیا جائے: ''0 کے علاوہ کوئی اور عدد داخل کیجیے'':

پروگرام 6-3 ریاضی کے صرف چار بنیادی عملوں کو انجام دینے کے مقصد سے ایک سادہ کیلکولیٹر کی تشکیل کے لیے پروگرام تحریر کیجیے۔

```
#Program to create a four function calculator
result = 0
val1 = float(input("Enter value 1: "))
val2 = float(input("Enter value 2: "))
op = input("Enter any one of the operator (+,-,*,/): ")
if op == "+":
    result = val1 + val2
elif op == "-":
    if val1 > val2:
        result = val1 - val2
    else:
        result = val2 - val1
elif op == "*":
        result = val1 * val2
elif op == "/":
```

```
        if val2 == 0:
            print("Error! Division by zero is not allowed. Program
    terminated")
        else:
         result = val1/val2
else:
    print("Wrong input,program terminated")
print("The result is ",result)
```

نتیجہ:

```
Enter value 1: 84
Enter value 2: 4
Enter any one of the operator (+,-,*,/): /
The result is 21.0
```

پروگرام میں، "-"اور"/" آپریٹر کے لیےelif بلاک کے اندرif..else شرط موجود ہے۔اسے نیسٹیڈ if کہتے ہیں۔ہمif..else بیانات کے اندر نیسٹنگ کی کئی سطحیں تشکیل دے سکتے ہیں۔

6.3 انڈینٹیشن (INDENTATION)

اکثر پروگرامنگ لینگویجز میں، بلاک کے اندر والے بیانات کو منجھلے بریکٹ (Curly brackets) میں رکھا جاتا ہے۔تاہم پائتھن میں بلاک کے ساتھ ساتھ نیسٹیڈ بلاک کے لیے انڈینٹیشن کا استعمال کیا جاتا ہے۔بیان کے شروع میں چھوڑی گئی جگہ (Leading whitespace) جس میں خالی جگہ اور ٹیب شامل ہیں، انڈینٹیشن کہلاتی ہے۔پائتھن میں ایک ہی سطح کے انڈینٹیشن کے تحت آنے والے بیانات کوڈ کے صرف ایک بلاک سے وابستہ ہوتے ہیں۔انٹرپریٹر انڈینٹیشن کی سطحوں کی جانچ بہت مستعدی کے ساتھ کرتا ہے اور اگر انڈینٹیشن درست نہیں ہے تو سنٹیکس کی غلطی (Syntex error) کو ظاہر کر دیتا ہے۔انڈینٹیشن کی ہر ایک سطح کے لیے صرف ایک ٹیب کا استعمال ایک عام طریقہ ہے۔

پروگرام 6-4 میں if-else بیان میں بیانات کے دو بلاک ہیں اور ہر ایک بلاک کے بیانات کا انڈینٹیشن خالی جگہوں یا ٹیب کی مساوی مقدار کے ساتھ کیا گیا ہے۔

پروگرام 6-4 پہلے سے متعین اعداد میں سب سے بڑا عدد معلوم کرنے کے لیے پروگرام

```
#Program 6-4
#Program to find larger of the two numbers
num1 = 5
num2 = 6
if num1 > num2:                          #Block1
        print("first number is larger")
        print("Bye")
else:                                    #Block2
```

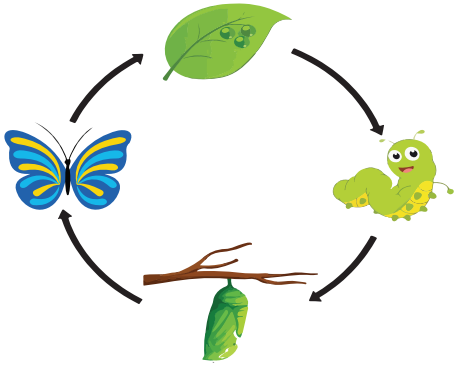
```
print("second number is larger")
print("Bye Bye")
```

نتیجہ:

```
second number is larger
Bye Bye
```

6.4 تکرار (REPETITION)

ہم اکثر و بیشتر بہت سے کاموں کو مکرر انداز میں انجام دیتے ہیں، مثلاً بجلی کا بل، جس کی ادائیگی ہر ماہ کی جاتی ہے۔ شکل 6.3 میں تتلی کے دور حیات کو دکھایا گیا ہے جس کے چار مراحل ہیں مثلاً تتلی انڈے دیتی ہے، انڈے کیٹر پلر میں تبدیل ہو جاتے ہیں جو پیوپا بن جاتے ہیں اور بالآخر نمو پا کر بالغ تتلی کی شکل اختیار کر لیتے ہیں۔ یہ دور تتلی کے انڈے دینے کے ساتھ دوبارہ شروع ہو جاتا ہے۔ اس قسم کی تکرار کو اعادہ (Iteration) کہتے ہیں۔ پروگرام میں بیانات کے ایک سیٹ کی تکرار کو لوپ کی مدد سے ممکن بنایا جا سکتا ہے۔ اس کے بارے میں مزید جاننے کے لیے آیئے پروگرام 6-5 پر غور کریں۔



شکل 6.3: قدرتی ماحول میں واقع ہونے والا تکراری عمل

پروگرام 6-5 پہلے پانچ طبیعی اعداد کو پرنٹ کرنے کے لیے پروگرام

```
#Program 6-5
#Print first five natural numbers
print(1)
print(2)
print(3)
print(4)
print(5)
```

نتیجہ :

```
1
2
3
4
5
```

اگر ہم سے پہلے 100,000 فطری اعداد کو پرنٹ کرنے کے لیے کہا جائے تو ہمیں کیا کرنا چاہیے؟ 100,000 پرنٹ اسٹیٹمنٹ لکھنا مسئلہ کا کوئی کارگر حل نہیں ہے۔ یہ تھکا دینے والا اور مشکل بھرا کام ہے اور مذکورہ کام کو انجام دینے کا بہترین طریقہ نہیں ہے۔ لوپ یا تکرار پر مشتمل پروگرام اس مسئلہ کا بہتر حل ہے۔ پروگرام کی منطق کو ذیل میں دیا گیا ہے:

1۔ ایک متغیر مثلاً Count لیجیے اور اسے قدر 1 تفویض کیجیے۔

2۔ Count کی قدر کو پرنٹ کیجیے۔

3۔ متغیر میں اضافہ کیجیے (Count += 1)

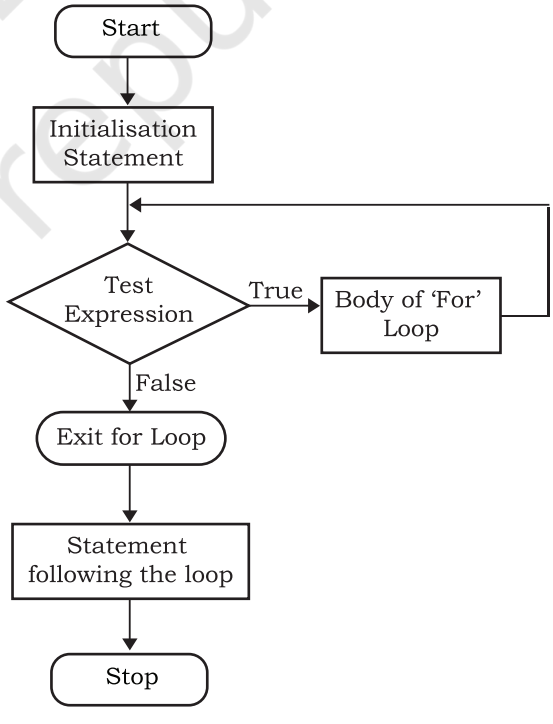
4۔ قدم 2 اور 3 کو اس وقت تک دہرایئے جب تک Count کی قدر 100,000 یا اس سے کم ہے (Count <= 100,000)

لوپ کی تشکیل سے پروگرام میں موجود بیانات کے سیٹ کو شرط کے مطابق مکرر طور پر ایگزیکیوٹ کرنے میں مدد ملتی ہے۔ لوپ کے اندر موجود بیانات کو اس وقت تک بار بار ایگزیکیوٹ کیا جاتا ہے جب تک منطقی شرط درست ثابت ہوتی ہے۔ اس شرط کی جانچ اس متغیر کی قدر کی بنیاد پر کی جاتی ہے جسے لوپ کنٹرول متغیر کہا جاتا ہے۔ جیسے ہی شرط غلط ثابت ہوتی ہے لوپ اختتام کو پہنچ جاتا ہے۔ یہ پروگرامر کی ذمے داری ہے کہ وہ اس بات کو یقینی بنائے کہ یہ شرط بالآخر غلط ثابت ہو سکے تا کہ لوپ سے باہر آنے کی حالت پیدا ہو سکے اور یہ لامتناہی لوپ نہ بن سکے۔ مثال کے طور پر، اگر ہم شرط Count <= 100,000 کو متعین نہیں کرتے ہیں تو پروگرام کبھی بھی اختتام کو نہیں پہنچے گا۔ پائتھن میں دو قسم کے لوپ ہوتے ہیں for اور While۔

6.4.1 'For' لوپ

for اسٹیٹمنٹ کا استعمال قدروں یا سلسلے کی رینج کے اعادے کے لیے کیا جاتا ہے۔ for لوپ رینج کے ہر ایک آئٹم کے لیے ایگزیکیوٹ ہوتا ہے۔ یہ قدریں عددی بھی ہو سکتی ہیں یا جیسا کہ ہم آئندہ ابواب میں دیکھیں گے یہ اسٹرنگ، لسٹ یا ٹپل جیسے ڈیٹا ٹائپ کے عناصر ہو سکتی ہیں۔

لوپ کے ہر ایک اعادے کے ساتھ کنٹرول متغیر اس بات کی جانچ کرتا ہے کہ آیا رینج کی ہر ایک قدر کو عمل میں لایا گیا ہے یا نہیں۔ جب رینج کے سبھی آئٹم ختم ہو جاتے ہیں تو لوپ کے اندر موجود بیانات ایگزیکیوٹ نہیں ہوتے ہیں اور کنٹرول for لوپ کے فوراً بعد والے بیان پر منتقل ہو جاتا ہے۔ for لوپ کا استعمال کرتے وقت ہمیں یہ بات پہلے ہی معلوم ہو جاتی ہے کہ لوپ کتنی مرتبہ ایگزیکیوٹ ہوگا۔ for لوپ پر عمل درآمد کو ظاہر کرنے والا فلو چارٹ شکل 6.4 میں دکھایا گیا ہے۔



شکل 6.4: لوپ کے لیے فلو چارٹ

(A) For لوپ کے لیے سنٹیکس

```
for <control-variable> in <sequence/
items in range>:
    <statements inside body of the
     loop>
```

نوٹ

پروگرام 6-6 for لوپ کا استعمال کرکے اسٹرنگ "PYTHON" میں موجود حروف کو پرنٹ کرنے کے لیے پروگرام

```
#Program 6-6
#Print the characters in word PYTHON using for loop
for letter in 'PYTHON':
     print(letter)
```

نتیجہ:

```
P
Y
T
H
O
N
```

پروگرام 6-7 for لوپ کا استعمال کرکے دیے ہوئے سلسلے میں موجود اعداد کو پرنٹ کرنے کے لیے پروگرام

```
#Program 6-7
#Print the given sequence of numbers using for loop
count = [10,20,30,40,50]
for num in count:
     print(num)
```

نتیجہ:

```
10
20
30
40
50
```

پروگرام 6-8 for لوپ کا استعمال کرکے دیے ہوئے سلسلے میں موجود جفت اعداد کو پرنٹ کرنے کے لیے پروگرام

```
#Program 6-8
#Print even numbers in the given sequence
numbers = [1,2,3,4,5,6,7,8,9,10]
for num in numbers:
     if (num % 2) == 0:
          print(num,'is an even Number')
```

نتیجہ:

```
2  is an even Number
4  is an even Number
```

```
6  is an even Number
8  is an even Number
10 is an even Number
```

نوٹ: لوپ کے متن کا انڈنٹیشن for بیان کی مناسبت سے ہوتا ہے۔

(B) *Range()* فنکشن

پائتھن میں Range() ایک بلٹ ان فنکشن ہے۔ Range() فنکشن کا سنٹیکس اس طرح ہے:

```
range([start], stop[, step])
```

اس کا استعمال دی ہوئی ابتدائی قدر سے لے کر اختتامی قدر (اختتامی قدر شامل نہیں ہے) تک دیے ہوئے فرق کے ساتھ صحیح اعداد کے سلسلے پر مشتمل فہرست کی تشکیل کے لیے کیا جاتا ہے۔ ہم فنکشن کے بارے میں آئندہ باب میں پڑھیں گے۔ شروع میں صرف یہ یاد رکھیے کہ فنکشن پیرامیٹر کے ساتھ کام کرتا ہے۔ فنکشن Range() میں اسٹارٹ، اسٹاپ اور اسٹیپ پیرامیٹر ہیں۔

اسٹارٹ اور اسٹیپ پیرامیٹر اختیاری ہوتے ہیں۔ اگر ابتدائی قدر متعین نہیں ہے تو فہرست 0 سے شروع ہوتی ہے۔ اسی طرح اگر اسٹیپ بھی متعین نہیں ہے تو فہرست کی ہر اگلی قدر میں 1 کا اضافہ ہوگا۔ Range() فنکشن کے سبھی پیرامیٹر صحیح اعداد ہونے چاہئیں۔ اسٹیپ پیرامیٹر مثبت صحیح عدد بھی ہوسکتا ہے اور منفی صحیح عدد بھی لیکن صفر نہیں۔

مثال 6.4

```
#start and step not specified
>>> list(range(10)-)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

#default step value is 1
>>> list(range(2, 10))
[2, 3, 4, 5, 6, 7, 8, 9]

#step value is 5
>>> list(range(0, 30, 5))
[0, 5, 10, 15, 20, 25]

#step value is -1. Hence, decreasing
#sequence is generated
>>> range(0, -9, -1)
[0, -1, -2, -3, -4, -5, -6, -7, -8]
```

فنکشن Range() کو عام طور سے اعداد کے سلسلے کی تشکیل کرنے کے لیے for لوپ میں استعمال کیا جاتا ہے۔

پروگرام 6-9 اعداد کی دی ہوئی رینج میں 10 کے اضعاف کو پرنٹ کرنے کے لیے پروگرام

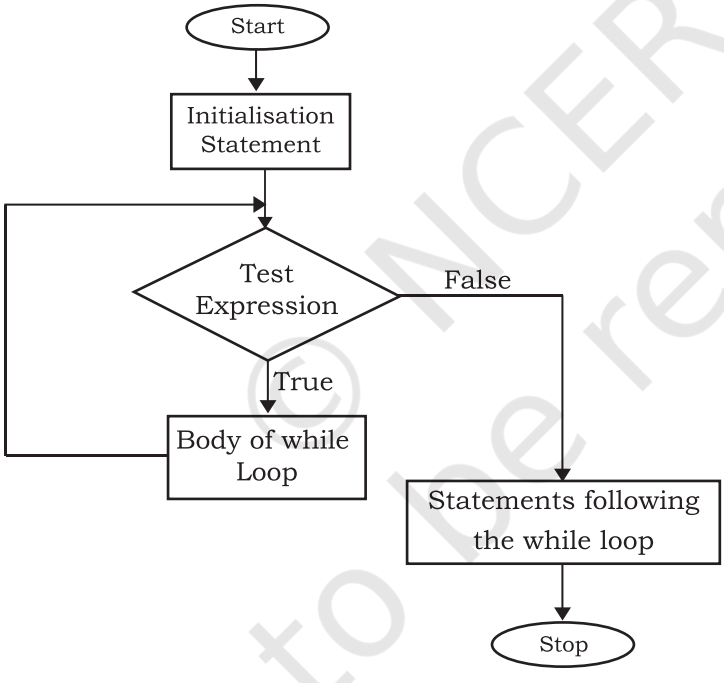
```
#Program 6-9
#Print multiples of 10 for numbers in a given range
for num in range(5):
     if num > 0:
          print(num * 10)
```

نتیجہ:

```
10
20
30
40
```

6.4.2 'While' لوپ

while اسٹیٹمنٹ کوڈ کے ایک بلاک کو مکرر طور پر اس وقت تک ایگزیکیوٹ کرتا ہے جب تک کہ لوپ کی کنٹرول شرط درست ثابت ہوتی ہے۔ while لوپ کی کنٹرول شرط پر عمل درآمد لوپ کے اندر موجود کسی بھی بیان پر عمل درآمد سے پہلے کیا جاتا ہے۔ ہر تکرار کے بعد کنٹرول شرط کی دوبارہ جانچ کی جاتی ہے اور لوپ اس وقت تک جاری رہتا ہے جب تک کہ شرط درست ثابت ہوتی ہے۔ جیسے ہی شرط غلط ثابت ہوتی ہے لوپ کے اندر موجود بیانات پر عمل درآمد بند ہو جاتا ہے اور کنٹرول while لوپ کے فوراً بعد والے بیان پر منتقل ہو جاتا ہے۔ اگر while لوپ کی شرط شروع میں ہی غلط ثابت ہو جاتی ہے تو لوپ کے متن پر ایک مرتبہ بھی عمل درآمد نہیں ہوتا ہے۔



شکل 6.5: فلوچارٹ برائے while لوپ

while لوپ کے اندر موجود بیانات کے لیے یہ ضروری ہے کہ وہ اس بات کو یقینی بنائیں کہ شرط بالآخر غلط ثابت ہو سکے بصورت دیگر لوپ لامتناہی لوپ بن جائے گا اور پروگرام میں منطقی غلطی (logical error) ظاہر ہو جائے گی۔ while لوپ کا فلوچارٹ شکل 6.5 میں دکھایا گیا ہے۔

while لوپ کا سنٹیکس

```
while test_condition:
  body of while
```

پروگرام 6-10 while لوپ کا استعمال کرتے ہوئے پہلے 5 طبیعی اعداد کو پرنٹ کرنے کے لیے پروگرام

```
#Program 6-10
#Print first 5 natural numbers using while loop
count = 1
while count <= 5:
     print(count)
     count += 1
```

نتیجہ:

```
1
2
3
4
5
```

پروگرام 6-11 while لوپ کا استعمال کرتے ہوئے مکمل عدد کے اجزائے ضربی معلوم کرنے کے لیے پروگرام

```
#Program 6-11
#Find the factors of a number using while loop
num = int(input("Enter a number to find its factor: "))
print (1, end=' ')    #1 is a factor of every number
factor = 2
while factor <= num/2 :
     if num % factor == 0:
#the optional parameter end of print function specifies the delimeter
#blank space(' ') to print next value on same line
          print(factor, end=' ')
     factor += 1
print (num, end=' ') #every number is a factor of itself
```

نتیجہ:

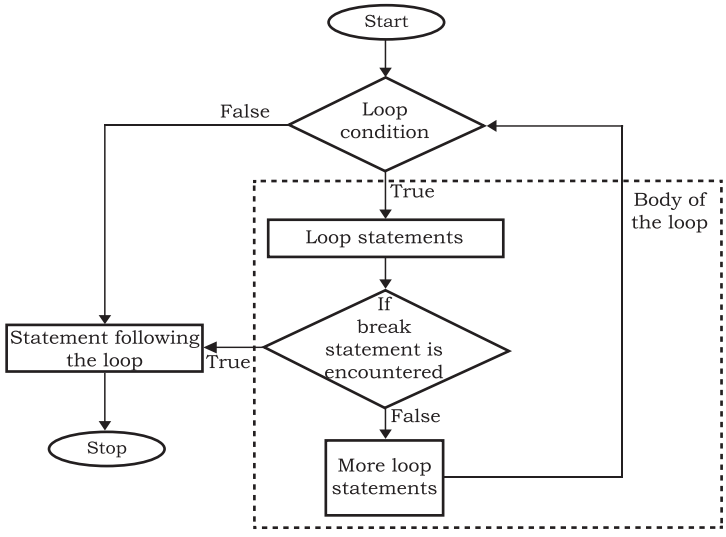
```
Enter a number to find its factors : 6
1 2 3 6
```

نوٹ: لوپ کے متن کا انڈینٹیشن while بیان کی مناسبت سے کیا جاتا ہے۔ اسی طرح if کے تحت آنے والے بیانات کا انڈینٹیشن if بیان کے مقام کی مناسبت سے کیا جاتا ہے۔

6.5 منقطع اور جاری بیان (BREAK AND CONTINUE STATEMENT)

لوپ پر مشتمل ساختوں کی مدد سے پروگرام مجوزہ عمل کی موثر طور پر تکرار کر سکتا ہے۔ بعض حالات میں جب کوئی مخصوص شرط پوری ہو جاتی ہے تو ہمیں لوپ کو مزید جاری رکھنے سے پہلے لوپ سے باہر نکلنا پڑ سکتا ہے (ہمیشہ

کے لیے لوپ سے باہر آنا) یا لوپ کے کچھ بیانات کو چھوڑنا پڑ سکتا ہے۔ ان ضروریات کی تکمیل بالترتیب break اور continue بیانات کی مدد سے کی جاسکتی ہے۔ پائتھن میں یہ بیانات پروگرامر کو ایک ٹول (tool) کے طور پر فراہم کیے جاتے ہیں تاکہ اسے پروگرام پر عمل درآمد کے تسلسل کو کنٹرول کرنے کے لیے مزید لچک مہیا کی جاسکے۔



شکل 6.5 : لوپ میں منقطع بیان کو استعمال کرنے کے لیے فلوچارٹ

6.5.1 منقطع بیان

منقطع بیان (Break statement) عمل درآمد کے عام تسلسل کو متنبہ کرتا ہے کیوں کہ موجودہ لوپ کو منقطع کر دیتا ہے اور اس لوپ کے بعد والے بیان پر عمل درآمد دوبارہ شروع کر دیتا ہے۔

پروگرام 6-12 break اسٹیٹمنٹ کے استعمال کو ظاہر کرنے کے لیے پروگرام

```
#Program 6-12
#Program to demonstrate the use of break statement in loop
num = 0
for num in range(10):
     num = num + 1
     if num == 8:
          break
     print('Num has value ' + str(num))
print('Encountered break!! Out of loop')
```

نتیجہ:

```
Num has value 1
Num has value 2
Num has value 3
Num has value 4
Num has value 5
Num has value 6
Num has value 7
Encountered break!! Out of loop
```

نوٹ: جب num کی قدر 8 ہوجاتی ہے تو break اسٹیٹمنٹ ایگزیکیوٹ ہوتا ہے اور for لوپ منقطع ہوجاتا ہے۔

پروگرام 6-13 استعمال کنندہ کے ذریعے داخل کیے گئے سبھی مثبت اعداد کا حاصل جمع معلوم کیجیے۔ جیسے ہی استعمال کنندہ منفی عدد داخل کرتا ہے تو پروگرام استعمال کنندہ سے مزید ان پُٹ حاصل کرنا بند کر دیتا ہے اور حاصل جمع کو ظاہر کر دیتا ہے۔

```
#Program 6-13
#Find the sum of all the positive numbers entered by the user
#till the user enters a negative number.
entry = 0
sum1 = 0
print("Enter numbers to find their sum, negative number ends the
loop:")
while True:
#int() typecasts string to integer
     entry = int(input())
     if (entry < 0):
          break
     sum1 += entry
print("Sum =", sum1)
```

نتیجہ:

```
Enter numbers to find their sum, negative number ends the loop:
3
4
5
-1
Sum = 12
```

پروگرام 6-14 یہ معلوم کرنے کے لیے پروگرام کہ آیا داخل کیا گیا عدد مفرد ہے یا نہیں۔

```
#Program 6-14
#Write a Python program to check if a given number is prime or not.
num = int(input("Enter the number to be checked: "))
flag = 0                    #presume num is a prime number
if num > 1 :
     for i in range(2, int(num / 2)):
          if (num % i == 0):
               flag = 1      #num is a not prime number
               break         #no need to check any further

     if flag == 1:
          print(num , "is not a prime number")
     else:
          print(num , "is a prime number")
```

```
else :
        print("Entered number is <= 1, execute again!")
```

نتیجہ 1:

```
Enter the number to be checked: 20
20 is not a prime number
```

نتیجہ 2:

```
Enter the number to check: 19
19 is a prime number
```

نتیجہ 3:

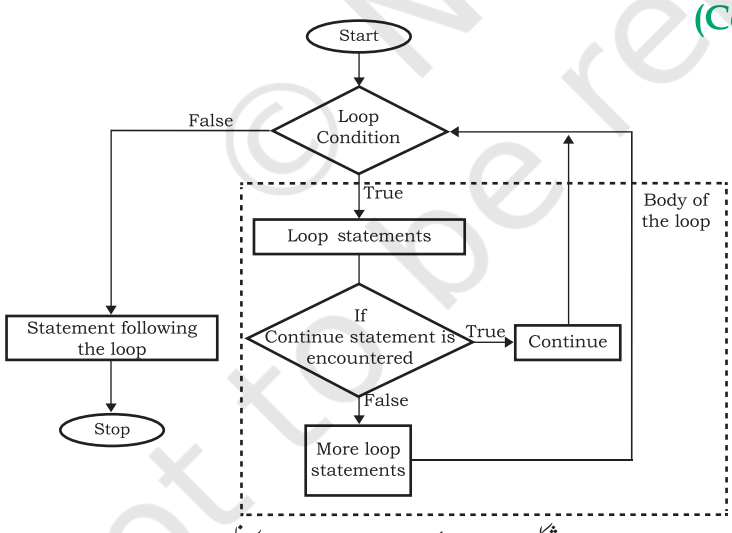
```
Enter the number to check: 2
2 is a prime number
```

نتیجہ 4:

```
Enter the number to check: 1
Entered number is <= 1,  execute again!
```

6.5.2 مسلسل بیان (Continue Statement)

جب کنٹرول کسی Continue بیان پر پہنچتا ہے تو وہ موجودہ تکرار کے لیے لوپ کے اندر موجود باقی ماندہ بیانات کو چھوڑ دیتا ہے اور اگلی تکرار کے لیے لوپ کے شروع میں چلا جاتا ہے۔ اگر لوپ کی شرط ابھی بھی درست ثابت ہوتی ہے تو لوپ دوبارہ شروع ہو جاتا ہے بصورت دیگر کنٹرول لوپ کے فوراً بعد والے بیان پر منتقل ہو جاتا ہے۔ شکل 6.7 میں Continue بیان کا فلوچارٹ دکھایا گیا ہے۔



شکل 6.7 : جاری (continue) بیان کا فلوچارٹ

پروگرام 6-15 مسلسل (Continue) بیان کے استعمال کو ظاہر کرنے کے لیے پروگرام

```
#Program 6-15
#Prints values from 0 to 6 except 3
num = 0
for num in range(6):
```

```
num = num + 1
if num == 3:
          continue
     print('Num has value ' + str(num))
print('End of loop')
```

نتیجہ:

```
Num has value 1
Num has value 2
Num has value 4
Num has value 5
Num has value 6
End of loop
```

مشاہدہ کیجیے کہ آؤٹ پُٹ (نتیجہ) میں قدر 3 کو پرنٹ نہیں کیا گیا ہے، لیکن لوپ، Continue اسٹیٹمنٹ کے بعد for لوپ کے منقطع ہو جانے تک دیگر قدروں کو پرنٹ کرنے کے لیے جاری رہتا ہے۔

6.6 نیسٹیڈ لوپ (NESTED LOOP)

ایک لوپ کے اندر دوسرا لوپ موجود ہو سکتا ہے۔ جب کسی لوپ کے اندر کوئی دوسرا لوپ موجود ہوتا ہے تو اسے نیسٹیڈ لوپ کہتے ہیں۔

پروگرام 6-16 نیسٹیڈ for لوپ کے استعمال کو ظاہر کرنے کے لیے پروگرام

```
#Program 6-16
#Demonstrate working of nested for loops
for var1 in range(3):
    print( "Iteration " + str(var1 + 1) + " of outer loop")
    for var2 in range(2):    #nested loop
        print(var2 + 1)
    print("Out of inner loop")
print("Out of outer loop")
```

نتیجہ:

```
Iteration 1 of outer loop
1
2
Out of inner loop
Iteration 2 of outer loop
1
2
Out of inner loop
```

```
Iteration 3 of outer loop
1
2
Out of inner loop
Out of outer loop
```

پائتھن میں اس بات کی کوئی قید نہیں ہے کہ کسی لوپ کے اندر یا نیسٹنگ کی سطحوں پر کتنے لوپ بنائے جا سکتے ہیں۔ کسی بھی قسم کے لوپ (for/while) کو دوسرے لوپ (for/while) کے اندر رکھا جا سکتا ہے۔

پروگرام 6-17 استعمال کنندہ کے ذریعے داخل کیے گئے عدد کے لیے پیٹرن کو پرنٹ کرنے والا پروگرام

```
#Program 6-17
#Program to print the pattern for a number input by the user
#The output pattern to be generated is
#1
#1 2
#1 2 3
#1 2 3 4
#1 2 3 4 5
num = int(input("Enter a number to generate its pattern = "))
for i in range(1,num + 1):
     for j in range(1,i + 1):
          print(j, end = " ")
     print()
```

نتیجہ:

```
Enter a number to generate its pattern = 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

پروگرام 6-18 نیسٹیڈ for لوپ کا استعمال کر کے 2 سے 50 کے درمیان مفرد اعداد معلوم کرنے کے لیے پروگرام

```
#Program 6-18
#Use of nested loops to find the prime numbers between 2 to 50

num = 2
for i in range(2, 50):
     j= 2
     while ( j <= (i/2)):
```

```
        if (i % j == 0):      #factor found
            break             #break out of while loop
        j += 1
    if ( j > i/j) :           #no factor found
        print ( i, "is a prime number")
print ("Bye Bye!!")
```

نتیجہ:

```
2 is a prime number
3 is a prime number
5 is a prime number
7 is a prime number
11 is a prime number
13 is a prime number
17 is a prime number
19 is a prime number
23 is a prime number
29 is a prime number
31 is a prime number
37 is a prime number
41 is a prime number
43 is a prime number
47 is a prime number
Bye Bye!!
```

پروگرام 19-6 دیے ہوئے عدد کے فیکٹوریل کی تحسیب کرنے کے لیے پروگرام

```
#Program 6-19
#The following program uses a for loop nested inside an if..else
#block to calculate the factorial of a given number

num = int(input("Enter a number: "))
fact = 1
# check if the number is negative, positive or zero
if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1, num + 1):
        fact = fact * i
    print("factorial of ", num, " is ", fact)
```

نوٹ

نتیجہ:

```
Enter a number: 5
Factorial of  5  is  120
```

خلاصہ

- If اسٹیٹمنٹ کا استعمال انتخاب یا فیصلہ سازی کے لیے کیا جاتا ہے۔
- for اور while لوپ کوڈ کے سیکشنوں کو کسی شرط کے تحت مکرر طور پر ایگزیکیوٹ کرنے میں مدد کرتے ہیں۔
- for اسٹیٹمنٹ کا استعمال قدروں یا سلسلے کی رینج کے اعادے کے لیے کیا جاتا ہے۔
- for لوپ کے اندر موجود بیانات اس وقت تک ایگزیکیوٹ ہوتے ہیں جب تک کہ رینج کے سبھی آئٹم ختم نہیں ہو جاتے۔
- while لوپ کے اندر موجود بیانات اس وقت تک ایگزیکیوٹ ہوتے ہیں جب تک while کی شرط غلط ثابت نہیں ہو جاتی۔
- اگر while لوپ کی شرط شروع میں ہی غلط ثابت ہو جاتی ہے تو لوپ کے متن پر ایک مرتبہ بھی عمل درآمد نہیں ہوتا ہے۔
- while لوپ کے اندر موجود بیانات کے لیے یہ ضروری ہے کہ وہ اس بات کو یقینی بنائیں کہ شرط بالآخر غلط ثابت ہو سکے بصورت دیگر لوپ لامتناہی لوپ بن جائے گا اور پروگرام میں منطقی غلطی (logical error) ظاہر ہو جائے گی۔
- بریک (Break) اسٹیٹمنٹ لوپ کے باقی حصے کو چھوڑتے ہوئے فوراً لوپ سے باہر آجاتا ہے اور لوپ کے ٹھیک بعد والے بیان پر عمل درآمد جاری رہتا ہے۔ جب کنٹرول کسی continue بیان پر پہنچتا ہے تو وہ اگلی تکرار کے لیے لوپ کے شروع میں چلا جاتا ہے۔
- جب کسی لوپ کے اندر کوئی دوسرا لوپ موجود ہوتا ہے تو اسے نیسٹیڈ لوپ (Nested Loop) کہتے ہیں۔

مشق

1۔ if اسٹیٹمنٹ میں else اور elif کے درمیان کیا فرق ہے؟

2۔ Range() فنکشن کا کیا مقصد ہے؟ ایک مثال دیجیے۔

3۔ مثالوں کی مدد سے Break اور Continue اسٹیٹمنٹ کے درمیان فرق کی وضاحت کیجیے۔

4۔ لامتناہی لوپ کیا ہوتا ہے؟ ایک مثال دیجیے۔

5۔ مندرجہ ذیل پروگرام کا آؤٹ پُٹ (نتیجہ) معلوم کیجیے۔

نوٹ

(i)
```
a = 110
while a > 100:
     print(a)
     a -= 2
```

(ii)
```
for i in range(20,30,2):
     print(i)
```

(iii)
```
country = 'INDIA'
for  i in country:
     print (i)
```

(iv)
```
i = 0; sum = 0
while i < 9:
     if i % 4 == 0:
          sum = sum + i
     i = i + 2
print (sum)
```

(v)
```
for x in range(1,4):
     for y in range(2,5):
          if x * y > 10:
               break
          print (x * y)
```

(vi)
```
var = 7
while var > 0:
     print ('Current variable value: ', var)
     var = var -1
     if var == 3:
          break
     else:
          if var == 6:
               var = var -1
               continue
     print ("Good bye!")
```

پروگرامنگ سے متعلق مشقیں

1۔ ایک پروگرام لکھیے جو استعمال کنندہ کا نام اور عمر ان پُٹ کے طور پر حاصل کرتا ہے اور ایک پیغام ظاہر کرتا ہے کہ آیا استعمال کنندہ ڈرائیونگ لائسنس کے لیے درخواست دینے کے اہل ہے یا نہیں (اہل ہونے کے لیے 18 برس کی عمر لازمی ہے)

2۔ دیے گئے عدد کا پہاڑہ پرنٹ کرنے کے لیے فنکشن لکھیے۔ عدد استعمال کنندہ کے ذریعے داخل کیا جائے گا۔

3۔ ایک پروگرام لکھیے جو استعمال کنندہ کے ذریعے داخل کیے گئے پانچ اعداد میں سے سب سے بڑے اور سب سے چھوٹے عدد کو پرنٹ کرتا ہے۔

4۔ ایک پروگرام لکھیے جو اس بات کی جانچ کرتا ہے کہ استعمال کنندہ کے ذریعے داخل کیا گیا سال لوند کا سال ہے یا نہیں۔

نوٹ

5۔ مندرجہ ذیل تسلسل کی تشکیل کے لیے پروگرام لکھیے: n تک .....-25, 20, -15, 10, -5 جہاں n استعمال کنندہ کے ذریعے داخل کیا گیا ایک صحیح عدد ہے۔

6۔ $1/n^3$.....1/27 + 1/8 +1 کا حاصل جمع معلوم کرنے کے لیے ایک پروگرام لکھیے، جہاں n استعمال کنندہ کے ذریعے داخل کیا گیا عدد ہے۔

7۔ استعمال کنندہ کے ذریعے داخل کیے گئے صحیح عدد کے ہندسوں کا حاصل جمع معلوم کرنے کے لیے پروگرام لکھیے۔

8۔ ایک فنکشن لکھیے جو اس بات کی جانچ کرتا ہے کہ آیا ان پُٹ عدد پیلنڈ روم ہے یا نہیں۔
(نوٹ: کسی عدد یا اسٹرنگ کو اس وقت پیلنڈ روم کہا جاتا ہے جب یہ معکوس ترتیب میں لکھنے پر بھی اپنی اصل شکل میں ظاہر ہوتا ہے۔ مثال کے طور پر 12321 ایک پیلنڈ روم ہے جب کہ 123421 پیلنڈ روم نہیں ہے)

9۔ مندرجہ ذیل پیٹرن کو پرنٹ کرنے کے لیے ایک پروگرام لکھیے:

i)	* * * * * * * * * * * * *	ii)	1 2 1 2 3 2 1 2 3 4 3 2 1 2 3 4 5 4 3 2 1 2 3 4 5
iii)	1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	iv)	* * * * * * * *

10۔ اگر نیچے دیے گئے جدول کے مطابق گریڈ دیے جائیں تو کسی طالب علم کا گریڈ معلوم کرنے کے لیے پروگرام لکھیے۔

نمبروں کا فیصد	گریڈ
90% سے زیادہ	A
80% تا 90%	B
70% تا 80%	C
60% تا 70%	D
60% سے کم	E

نوٹ

طالب علم کے ذریعے حاصل کردہ فیصد نمبر پروگرام کے لیے ان پُٹ ہے۔

نظیری مطالعہ پر مبنی سوالات

آیئے ہم باب 5 میں فروغ دیے گئے اپنے SMIS کی فعالیت میں مزید اضافہ کریں۔

6.1 مینو کی مدد سے چلنے والا ایک ایسا پروگرام لکھیے جس میں مندرجہ ذیل متبادلات موجود ہوں :

- دسویں جماعت میں پانچ اہم مضامین میں طالب علم کے ذریعے حاصل کردہ نمبروں کو قبول کرنا اور انھیں ظاہر کرنا۔
- سبھی مضامین کے نمبروں کے حاصل جمع کی تحسیب کرنا۔ نمبروں کے حاصل جمع کو مضامین کی تعداد (یعنی 5) سے تقسیم کرنا، فیصد (Percentage = Total Marks/5) کی تحسیب کرنا اور Percentage کو ظاہر کرنا۔
- مندرجہ ذیل ضوابط کے مطابق طالب علم کا گریڈ معلوم کرنا۔

گریڈ	ضابطہ
A	percentage > 85
B	percentage < 85 && percentage >= 75
C	percentage < 75 && percentage >= 50
D	percentage > 30 && percentage <= 50
Reappear	percentage <30

آیئے باب 5 کے آخر میں "DOCUMENTATION TIPS" کے تحت دیے گئے معیارات کی بنیاد پر دیگر افراد کے نظیری مطالعہ پر نظر ثانی کریں اور انھیں بازرسی فراہم کریں۔