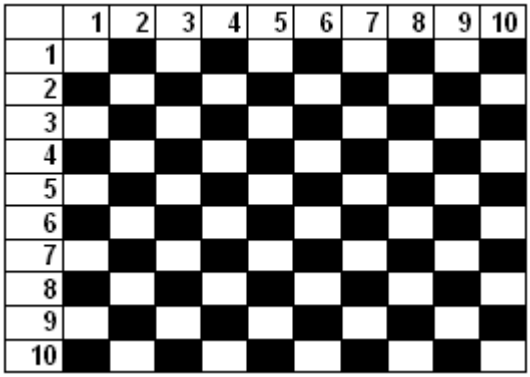

```
x1=int(input('Номер рядка першої клітинки: '))
y1=int(input('Номер стовпця першої клітинки: '))
x2=int(input('Номер рядка другої клітинки: '))
y2=int(input('Номер стовпця другої клітинки: '))
if 1<=x1<=8 and 1<=y1<=8 and 1<=x2<=8 and 1<=y2<=8:
     k1=x1%2==y1%2
     k2=x2%2==y2%2
     if k1==k2:
          print('Клітинки одного кольору.')
     else:
          print('Клітинки різного кольору.')
else:
     print('Невірні вхідні дані.')
```

6. Циклічні оператори

Досить часто і в житті, і при написанні програм існує необхідність повторення деякої дії певної кількості раз. Тобто при написанні програм може знадобитися певна конструкція, за якою можна буде організувати повторне виконання операторів. Таку конструкцію називають конструкцією повторення або циклом. А кожну повторену дію – кроком циклу або ітерацією. Отже, можна зазначити, що цикл у програмуванні – це

повторюване виконання одних і тих самих простих або складених операторів.

Всі цикли складаються з заголовку та тіла циклу. Заголовок циклу відповідає за налагодження циклу, тобто умову повторення циклу. Тіло ж циклу відповідає за самі дії, які мають повторно виконуватися. Так наприклад, уявімо собі першокласника, який дуже полюбляє морозиво. Йому мама видала певну суму грошей на морозиво. Зрозуміло, він побіг його купувати, але згадав, що множити та ділити він не вміє. Як же йому вирішити цю проблему? Спочатку він перевірить, чи вистачить йому грошей на купівлю пачки морозива. Якщо так, то він її купить і знову погляне на залишок грошей. В цьому прикладі можна виділити заголовок циклу – поки грошей достатньо, і тіло циклу – купівля морозива.

6.1. Цикл з передумовою (Цикл while)

Цикл з передумовою є одним з самих універсальних циклів в мові Python, але достатньо повільний. Цикл є циклом з передумовою, оскільки умова записується і перевіряється до тіла циклу. Цикл з передумовою ще називають циклом While, оскільки саме з цього ключового слова він починається.

Синтаксис оператора циклу while:

```
while Логічний_вираз:
    Блок_інструкцій
```

Логічний вираз також називають умовою виконання циклу, а блок інструкцій – тілом циклу, яке може містити довільні оператори. За циклом while виконується вказаний набір інструкцій до тих пір, поки умова циклу істинна.

При виконанні циклу while спочатку обчислюється значення логічного виразу, якщо це значення є істинним (істинність умови визначається так само як і в операторі if), то виконується тіло циклу і відбувається повернення до перевірки логічного виразу. Процес продовжується доти, поки значення логічного виразу не стане хибним. Після цього робота циклу завершиться і відбувається перехід до інструкції після тіла циклу while.

Якщо при першому обчисленні значення логічного виразу є хибним, тіло циклу не виконується жодного разу.

Щоб цикл закінчив роботу, в його тілі повинен бути оператор, що впливає на значення логічного виразу. Окрім того логічний вираз має бути коректним, тобто його значення повинно бути визначеним ще до першої перевірки.

Як правило цикл while використовується, коли неможливо визначити точне значення кількості проходів використання циклу.

6.1.1. Приклади розв'язування задач

Приклад. Обчислити значення $\sum_{n=1}^{\infty} \frac{1}{n^2}$ з точністю до 10^{-6}.

Тут сумування проводиться до тих пір, поки черговий доданок не стане менше заданої точності обчислень.

```
s=0
n=1
x=1
while abs(x)>=1e-6:
     s=s+x
     n=n+1
     x=1/n**2
print(s)
```

Приклад. Визначити кількість цифр натурального числа n:

```
n = int(input())
length = 0
while n > 0:
    n = n // 10
    length = length + 1
print(length)
```

Приклад. Обчислити суму непарних додатних чисел, менших за n.

```
n=int(input())
s=0
i=1
while i<n:
     s=s+i
```

```
        i=i+2
print('Сума непарних чисел =',s)
```

6.2. Тип діапазон (range)

Тип діапазон (range) є незмінюваною послідовністю цілих чисел.

Для задання діапазону призначені функції:

`range(stop)` – задання послідовності цілих чисел від 0 до `stop-1` з кроком `1`.

`range(start, stop[, step])` - задання послідовності, яка є арифметичною прогресію від `start` до `stop-1` з кроком `step`. Якщо параметр `step` опущений, він за замовчуванням дорівнює 1.

За функцією range(5) отримаємо діапазон в з елементів 0, 1, 2, 3, 4

За функцією range(1, 5) отримаємо діапазон в з елементів 1, 2, 3, 4

За функцією range(0, 10, 3) отримаємо діапазон в з елементів 0, 3, 6, 9

Для отримання діапазону, в якому значення будуть зменшуватися, необхідно використовувати функцію range з трьома параметрами. Третій параметр має бути від'ємним, а перший більшим ніж другий.

За функцією range(0, -5, -2) отримаємо діапазон в з елементів 0, -2, -4

Проте вивести на екран елементи утвореного діапазону звичайними методами неможливо, так:

```
>>> r = range(1, 5)
>>> r
range(1, 5)
```

Можна лише перевірити приналежність деякого числа до діапазону, використовуючи оператор in.

```
>>> 4 in r
True
>>> 6 in r
False
```

6.3. Цикл for

Окрім циклу з передумовою, в мові Python є цикл for, за яким надається можливість перебору всіх елементів з деякого набору

(послідовності, бінарної послідовності, рядка, множини, словника, файлу). В загальному можна зазначити, що використовуваним набором в циклі for, може будь який набір, що підтримує ітерування. Перебір елементів можна пояснити так. У нас є набір, що складається з ряду елементів. При переборі ми спочатку беремо з даного набору перший елемент, і в тілі циклу виконуємо над ним визначені дії. Потім беремо другий елемент, і над ним знову виконуємо ті ж дії. І так далі продовжуємо над всіма елементами набору. При такому опрацюванні не потрібно турбуватися про індекси елементів і їх кількість.

Іншими словами, можна зазначити, цикл for являє собою формальний запис інструкції виду: «Виконати операцію X для всіх елементів, що входять в набір M». Тому цикл for інколи називають циклом перегляду.

Синтаксис оператора циклу for записується так:

```
for Ідексна_змінна in Послідовність:
    Блок_інструкцій
```

На початку індексній змінній надається значення першого елемента послідовності, потім виконується тіло циклу (блок інструкцій) і індексній змінній надається значення наступного елемента послідовності. Так продовжується доти, поки індексній змінній послідовно не будуть надані значення всіх елементів послідовності. Тобто індексна змінна буде пробігати всі елементи послідовності.

Як правило, цикли for використовуються для виконання операцій над всіма елементами послідовності або виконання операцій таку кількість разів, яка відповідає кількості елементів в послідовності.

Цикл for дещо складніший і менш універсальний, але виконується значно швидше циклу while.

6.3.1. Приклади розв'язування задач

Приклад. Вивести на екран квадрати додатних цілих чисел, менших за n.

```
n=int(input())
for i in range(1,n):
     print(i**2)
```

Якщо значення змінної n буде рівне нулю або від'ємне, то тіло циклу не виконається жодного разу.

Приклад. Надрукувати числа від 10 до 1.

```
for i in range(10,0,-1):
     print(i)
```

Приклад. Обчислити суму ряду $\sum_{i=1}^{n} \frac{i}{2i+1}$ для заданого n.

```
n=int(input())
s=0
for i in range(1,n+1):
     s=s+i/(2*i+1);
print('Сума =', s)
```

Приклад. Обчислити суму непарних додатних чисел з проміжку [n; m].

```
n=int(input())
if n%2==0:n=n+1
m=int(input())
s=0
for i in range(n,m+1,2):
    s=s+i
print('Сума непарних чисел =',s)
```

В даній програмі спочатку уточнюється початок проміжку таким чином, щоб початком було перше непарне число з заданого проміжку. Потім за функцією range(n,m+1,2) формується діапазон всіх непарних чисел з заданого проміжку і відбувається їх підсумовування.

6.4. Інструкції управління циклами

Оператор continue

Оператор ***continue*** призначений для переривання поточної ітерації циклу і переходу до наступної. Тобто оператори, що будуть іти в тілі циклу після виклику continue, на даному кроці виконуватися не будуть.

Приклад. Обчислити суму непарних додатних чисел з проміжку [n; m], не включати до суми чисел, які діляться націло на 5.

```
n=int(input())
if n%2==0:n=n+1
m=int(input())
s=0
for i in range(n,m+1,2):
    if i%5==0:
        continue
    s=s+i
print('Сума непарних чисел =',s)
```

При виконанні тіла циклу, коли значення змінної і буде ділитися націло на 5 (залишок від цілочисельного ділення і на 5 дорівнює 0), відбувається перехід на наступний елемент послідовності.

Оператор break

Оператор ***break*** призначений для дострокового припинення роботи циклу (`for` або `while`), тобто зупинки виконання тіла циклу, навіть якщо умова виконання циклу ще не набула значення False або послідовність елементів не закінчилась.

Зрозуміло, інструкцію `break` варто викликати тільки всередині інструкції `if`, тобто вона повинна виконуватися тільки при виконанні якоїсь особливої умови.

Приклад. Обчислити суму непарних додатних чисел з проміжку [n; m]. При додаванні до суми числа, яке діляться націло на 5, припинити підсумовування.

```
n=int(input())
if n%2==0:n=n+1
m=int(input())
s=0
for i in range(n,m+1,2):
    s=s+i
    if i%5==0:
        break
print('Сума непарних чисел =',s)
```

6.5. Блок else в циклах

Блок ***else*** може використовуватися як додатковий блок циклів while та for, сфера застосування якого досить схожа до застосування однойменного блоку в конструкціях обробника винятків (try-except) та умовного оператора (if-else), тобто «якщо цього виконати не можна, то (інакше) виконати це».

Синтаксис операторів циклу з блоком else:

```
while Логічний_вираз:
    Блок_інструкцій_1
else:
    Блок_інструкцій_2
```

Або

```
for Ідексна_змінна in Послідовність:
    Блок_інструкцій_1
else:
    Блок_інструкцій_2
```

Інструкція всередині блоку else (`Блок_інструкцій_2`) виконується в тому випадку, коли цикл завершився згідно з умовою повторення циклу, вказаною в заголовку циклу (для циклу while у випадку, коли умова циклу стала хибною, для циклу for – коли були перебрані всі елементи послідовності).

Іншим варіантом завершення циклу є вихід з циклу за виконанням оператору `break` і в такому випадку інструкції блоку else не виконуються. Отже використання `else` доцільне тільки разом з інструкцією `break`.

Можливе використання оператора else в циклах - реалізація пошукових циклів. Наприклад, необхідно виконати пошук певного елемента, а у випадку його відсутності повідомити користувача про його відсутність.

```
for obj in objects:
    if obj.key == search_key:
        found_obj = obj
        break
else:
    print('No object found.')
```

Приклад. Обчислити суму непарних додатних чисел з проміжку [n; m]. При додаванні до суми числа, яке ділиться націло на 5, припинити підсумовування. Вивести додаткове повідомлення, якщо результуюча сума містить доданок, який ділиться націло на 5.

```
n=int(input())
if n%2==0:n=n+1
m=int(input())
s=0
for i in range(n,m+1,2):
    s=s+i
    if i%5==0:
        break
else:
    print('Сума не містить доданку, що ділиться на
5')
print('Сума непарних чисел =',s)
```

6.6. Вкладені цикли

В усіх операторах циклу мови Python оператор, який є тілом циклу, може сам бути оператором циклу або містити у собі оператор циклу. Утворена конструкція називається вкладеним циклом.

Приклад. За даним натуральним $n \le 9$ вивести сходи з n сходинок, де i-а сходинка складається з чисел від 1 до i без пропусків.

```
n=int(input('Вкажіть n'))
for i in range(1,n+1):
    for j in range(1,i+1):
        print(j,end='')
    print()
```

Приклад. Обчислити суму ряду $\sum_{i=1}^{10}\sum_{j=1}^{5}\frac{1}{i+j^2}$.

```
s=0
for i in range(1,11):
     for j in range(1,6):
```