DISPLAYING A NUMBER IN THE SHAPE OF A SNAKE

Rustamova Nigina Bobir kizi
student
Bukhara State University,
Uzbekistan, Bukhara
Rustamov Khakim Sharipovich
associate professor,
Bukhara State University,
Uzbekistan, Bukhara
https://doi.org/10.5281/zenodo.10275506

ABSTRACT

This research article discusses solving some problems in Python programming language. Solving such logical problems helps improve the student's knowledge. In this article we will work with numbers by arranging the natural numbers so that they have the shape of a snake. This will help you to use and understand the topic of loops correctly. We also solve using simple methods, these simple methods do not require more memory and are also faster. This solution method is called the best algorithm. If you solve problems with a better algorithm, it shows that your programming level has improved. If you solve difficult problems every day, you will improve every day. If you are interested, let's move on to explaining and solving such problems.

Keywords: input()-entered information, time- module for working with time, ljust()-aligns a string to the left.

An algorithm is a clear sequence of actions, the implementation of which gives some predetermined result. In simple words, it is a set of instructions for a specific task.[1]

In this code, you enter a natural number a and the program calculates the square of the given number a2 and displays all natural numbers in the range [1; a^2] in the form of a snake.

To begin with, I wrote a code that works with 3 loops:

1. First loop: we have the function of reading the number of lines
2. Second cycle: if the number of lines is odd, then the line is written from right to left.
3. Third cycle: if the number of lines is even, then the line is written from left to right.

First way

```
import time
a=int(input(' enter number:'))
begin=time.time()
b=a**2
s=0
x=b
while x!=0:
  s+=1
  x=x//10
d=0  # this number determines the line number, i.e. whether the string is even or
     #odd
n=0  # this is the maximum number of each line
result= " " # our result
for i in range(a):
```

```
    d+=1
    if d%2==1:
        for i in range(1+n,(a+n)+1):
            result+=str(i).ljust(s)+' '
        result+='\n'
        n+=a
    else:
        for i in range((a+n),n,-1):
            result+=str(i).ljust(s)+' '
        result+='\n'
        n+=a

print(result)
finish=time.time()
print(finish-begin)
```

Input data:

enter number: 4

Output:

1 2 3 4
8 7 6 5
9 10 11 12
16 15 14 13
0.01557016372680664
Time it took to run the first program 0.01557016372680664

Second way

The first loop counts the number of lines

The second loop adjusts the numbers that are output and the expression inside the conditional statement, the result depending on the number of lines.

```
import time
a=int(input(' enter number:'))
begin=time.time()
s=0
x=a**2
while x!=0:
    s+=1
    x=x//10
for i in range(1, a+1):
    for j in range(1,a+1):
        if i%2==1:
            print(str((i-1)*a+j).ljust(s), end='  ')
        else:
            print(str(i*a-j+1).ljust(s),end='  ')
    print('\n')
finish=time.time()
print(finish-begin)
```

Input data:

enter number: 4

Output:

```
1   2   3   4
8   7   6   5
9  10  11 12
16 15 14 13
0.062463998794555664
```

Time it took to run the second program 0.062463998794555664

Third way

In the introductory part of the code, we will add any natural number, which will create a square array the size of the same number. And then the array elements change depending on the list number:

-if the number of lists is even, then the if function is executed;

— if the number of lists is odd, then the else function is executed.

The last two loops turn the list into a string.

```
import time
n=int(input(' enter number:'))
begin=time.time()
massiv=[[0]*n for i in range(n)]
x=0
y=0
ln=len(str(n**2))
for a in range(n):
    for b in range(n):
        x+=1
        if (a+1)%2==0:
            massiv[a][b]=n*(a+1)-y
            y+=1
        else:
            massiv[a][b]=x
            y=0
for i in massiv:
    for s in i:
        print(str(s).ljust(ln),end=' ')
    print()
finish=time.time()
print(finish-begin)
```

Input data:

введите номер:4

Output:

```
1   2   3   4
8   7   6   5
9  10 11 12
16 15 14 13
```

0.04685807228088379

Time it took to run the third program 0.04685807228088379

Conclusion

The best solution to this problem is the first method, because it works faster than others. Each algorithmic problem must be solved at least 3 times. Because with every decision you learn and practice finding other, shorter options that take less time and memory. Today we learned how to properly use loops and subloops to create an array.

References:

1.skillfactory.ru