Лабораторна робота №5

Оператор циклу for

Мета роботи

- Ознайомлення з оператором for та генератором списку range

Короткі теоретичні відомості

Під час читання наступного теоретичного матеріалу, наполегливо радимо Вам повторювати усі наведені по ходу приклади та вправи.

1. Оператор for

1.1 for

Оператор for в Python трохи відрізняється від аналогічного оператора в C або Pascal. Замість незмінного проходження по арифметичній прогресії з чисел (як в Pascal) або надання користувачеві можливості вказати крок ітерації та умову останову (як в C), оператор for в Python проходить по всіх елементах будь-якої послідовності (списку або стрічки) в тому порядку, в якому вони в ній розташовуються.

Оператор for в Python має вигляд:

```
for певна_змінна in певний_діапазон
```

Блок коду після заголовка виконується, поки *певна_змінна* належить до *певного_діапазону*. Цей діапазон може бути списком, числовою послідовністю, масивом любих значень:

Створимо скрипт forexample.py:

```
print ("start")
for i in [1, 2, 3, 4, 5]:
     print i
print ("finish")
```

Результат роботи:

```
start
```

```
12345 finish
```

Оператор **for** забезпечує виконання циклу таким чином. Здається змінна циклу і список значень. У даному випадку і - змінна циклу, список значень - [1, 2, 3, 4, 5]. При виконанні циклу змінна і на кожному кроці приймає одне із значень заданого списку.

Список може бути довільним:

```
>>> phrase=['red', 'green', 'blue', 'yellow', 'black', 'white']
>>> for word in phrase:
      print len(word),word

3 red
5 green
4 blue
6 yellow
5 black
5 white
>>>
```

Ця програма виконує оператор print len(word), word для кожного елемента зі списку. Цей процес називається ітераціями. Наведемо інший приклад for циклу.

```
>>> total=0
>>> for w in phrase:
      total+=len(w)


>>> total/len(phrase)
4
>>>
```

1.2 Генерація списку з послідовними номерами

Функція **range(N)** відповідає за генерацію списку з N елементів арифметичної прогресії. Нагадуємо: рахунок елементів починається з нуля:

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(20)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
>>> range(40)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
37, 38, 39]
```

Також можна вказати початковий елемент прогресії:

```
>>> range(5,10)
[5, 6, 7, 8, 9]
>>> range(5,20)
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
>>> range(15,40)
[15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 38, 39]
```

Також можна вказувати початковий елемент і крок прогресії:

```
>>> range(5,10,2)
[5, 7, 9]
>>> range(15,40,5)
[15, 20, 25, 30, 35]
```

1.3 Використання функції range()

Особливо зручно поєднувати використання циклу **for** з використанням функції **range()**: Для скрипта:

```
print ("start")
for i in range(21):
print i,
print ("finish")
```

Результат роботи:

```
start
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 finish
```

Зверніть увагу на використання коми в команді print для того, щоб роздрукувати отримані значення скрипта в один рядок.

Також дана функція може використовуватися в парі з функцією **len()**, що виходить дуже зручним при обробці масивів. Наприклад, задамося завданням роздрукувати кожен другий елемент списку. При цьому довжина списку заздалегідь невідома. Використовуємо для цього команду **range()** у форматі **range(start, stop, step)** - приклад см нижче:

```
>>> range(0,20,2)
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

1.4 Використання оператора break

Оператор **break** достроково завершує цикл.

Змінимо завдання з роздрукуванням елементів списку. Нехай нам потрібно виводити на друк тільки до тих пір, поки в ньому не зустрінеться елемент з рядком *stop*. Після чого - закінчити цикл і надрукувати в тому ж рядку *end of list*, проігнорувавши інші елементи списку:

```
print ("start")
sp = [52,34,23,42,4,'stop',2314,234,56,6]
for i in sp:
     if i == 'stop':
          print 'end of list'
          break
     print i,
print ("finish")
```

Результат роботи:

```
start
52 34 23 42 4 end of list
finish
```

1.5 Використання оператора else всередині циклу for

Цикл **for** може включати в себе елемент **else**, який виконується після нормального завершення циклу, але не виконується після переривання циклу за директивою **break**. Цей оператор зручно використовувати в багатьох випадках. Наприклад, для перевірки - чи зустрічаються в списку певні елементи чи ні.

Нехай слід перевірити - чи зустрічаються в списку sp парні елементи. Це можна зробити за допомогою оператора розподілу %, який видає залишок від цілочисельного ділення. Перевірка наявності у списку парних елементів зведеться до того, щоб послідовно перевірити всі елементи на парність шляхом отримання залишку від ділення на два. Якщо зустрінеться шуканий парний елемент, то обчислення можна перервати і надрукувати що такий елемент є, якщо він не зустрінеться - після виконання циклу слід надрукувати, що таких елементів у списку немає:

```
sp = [52,34,23,42,4,'stop',2314,234,56,6]
for i in sp:
     if i % 2 == 0:
          print "list have at least one odd element"
```

```
            print i
            break
else:
     print "list does not have any odd element"
```

Результат роботи:

```
list have at least one odd element
52
```

Повторимо, прибравши зі списку всі парні елементи:

```
print ("start")
sp = [23,7,11,1,1,5,9]
for i in sp:
     if i % 2 == 0:
          print "list have at least one odd element"
          print i
          break
else:
     print "list does not have any odd element"
print ("finish")
```

Результат роботи:

```
start
list does not have any odd element
finish
```

1.6 continue

Оператор **continue** використовується для того, щоб пропустити один крок циклу. **continue** відразу повертає керування на початок циклу.

Приклад:

```
print ("start")
for i in ['p','y','t','h','o','n']:
     if i == 'h':
          continue
     print i
print ("finish")
```

Результат роботи:

```
start
pyton
finish
```

Завдання та порядок виконання роботи

1. Ознайомитися з наведеними вище теоретичними відомостями.

2. Виконати приклади, які приводяться в теоретичних відомостях.

3. Виконати наступні завдання:

   3.1. Написати скрипт, який:

   - на початку, у вигляді коментаря, буде містити назву курсу та номер лабораторної роботи, а також ваше ім'я та прізвище, та номер Вашої заліковки
   - у першому рядку буде виводити назву курсу та номер лабораторної роботи
   - у другому – буде виводити ваше ім'я та прізвище, а також номер Вашої заліковки.
   - Використовуючи оператори циклів (for або while) написати програму за варіантом:

1) Дано натуральне число N. Отримати всі прості дільники цього числа.
2) Вивести всі парні та числа які діляться на 3 від N до M. Значення N та M задаються з клавіатури.
3) Дано натуральне число *n*. Перевірити, чи можна подати *n*! у вигляді добутку трьох послідовних цілих чисел
4) Написати програму виведення всіх чисел від 1 до N, які закінчуються цифрою 3.
5) Дано натуральне число N. Серед чисел 1, ..., N знайти такі числа, запис яких співпадає з останніми цифрами запису їх квадрату. Наприклад, 6 ($6^2 = 36$), 25 ($25^2 = 625$) і т.д.
6) Вивести всі прості числа які менше введеного числа N.
7) Знайти суму чисел Фібоначчі, які не перевищують 1000 (числа Фібоначчі f_n обчислюються за формулами $f_0=f_1=1$; $f_n=f_{n-1}+f_{n-2}$ при n=2,3,... Числова послідовність Фібоначчі 1, 1, 2, 3, 5, 8, 13, ...).
8) Знайти перше число Фібоначчі, яке більше за m (m>1) (числа Фібоначчі f_n обчислюються за формулами $f_0=f_1=1$; $f_n=f_{n-1}+f_{n-2}$ при n=2,3,... Числова послідовність Фібоначчі 1, 1, 2, 3, 5, 8, 13, ...).
9) Знайти максмимальне значення функції y=sin(x*x)*x+cos(x), на відрізку [c,d] з кроком 0.001.
10) Визначити, чи є задане шестизначне число "щасливим" (сума перших трьох цифр має дорівнювати сумі останніх трьох цифр)?
11) Дано натуральне чотирицифрове число n. З'ясувати, чи є воно паліндромом (таким, що читається однаково зліва направо та справа наліво, наприклад 1221).
12) Написати програму, яка серед двоцифрових чисел вибирає числа, рівні сумі своїх цифр.
13) Написати програму знаходження суми n перших членів послідовності 1/2+3/4+5/6+...

14) Вивести 5 простих чисел, які більші введеного числа.
15) Вивести всі прості числа до числа n, різниця між якими дорівнює 6, тобто (p, p + 6). Наприклад (5,11), (7,13), (11,17), (13,19), (17,23), …
16) Знайти мінімальне значення функції y=sin(x)*x, на відрізку [c,d] з кроком 0.001.
17) Написати програму, яка виводить всі числа Мерсена від 1 до n . Просте число називається числом Мерсена, якщо його можна представити у вигляді $2^p - 1$, де p — теж просте число.
18) Знайти мінімальне значення функції y=sin(x)*x-cos(x), на відрізку [c,d] з кроком 0.001.
19) Вивести всі числа від n до m, які діляться на 3, але не діляться на 2.
20) Вивести ряд Фібоначчі до n-того члена (числа Фібоначчі f_n обчислюються за формулами $f_0=f_1=1$; $f_n=f_{n-1}+f_{n-2}$ при n=2,3,... Числова послідовність Фібоначчі 1, 1, 2, 3, 5, 8, 13, ...).

4. Зберегти скрипт у файл з наступною назвою <Surname>_Task5.py (наприклад Rodionov_Task5.py)

5. Запустити даний скрипт за допомогою інтерпретатора та переконатись у його працездатності та правильності результатів.

6. Спробувати здати та захистити лабораторну роботу у викладача практичних занять

Література

1. A Byte of Python (Russian) (http://wombat.org.ua/AByteOfPython/#id14)

2. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с..

3. Computational Physics with Python by M. Newman (http://www-personal.umich.edu/~mejn/computational-physics/)

Інтернет посилання

- Бібліотека math: http://bit.ly/18jMBaD
- http://ru.wikipedia.org/wiki/Python
- MIT: Introduction to Computer Science and Programming
- http://python.org