Програмування

ТЕМА 3. ЦИКЛІЧНІ ПРОГРАМИ

Цикл

Циклом називається повторення виконання деякої інструкції *P*

Циклічна програма – це програма яка є ланцюгом команд введення, виведення, присвоєння або тотожної команди, розгалуження а також циклу

3.1 Цикл з умовою продовження

Цикл з умовою продовження

Синтаксис

```
while F:
    P
```

де *F* – умова, *P* - інструкція

Правило циклу з умовою продовження.

- 1. Обчислюється значення F_0 умови *F*.
- 2.1 Якщо F_0 == False, то цикл завершує свою роботу.
- 2.2 Якщо F_0 == True, то виконується інструкція *P* і знову починає виконуватись цикл за цим же правилом.

Приклади циклів з умовою продовження

```
while x > 0:
    x = x - 1

while i < n:
    i = i + 1
    y = y * x

while y > 0:
    x = x -1
```

Скінченність циклів

Останній цикл `while y > 0:`

`x = x -1`,

один раз почавшись, ніколи не закінчиться.

Очевидно, що, якщо інструкція *P* не змінює умову *F*, то цикл буде нескінченним.

Тому **необхідною умовою скінченності циклу** є: інструкція *P* повинна змінювати умову *F*.

Хоарівська трійка

Хоарівська трійка – це трійка

$\{ F \} P \{ G \}$,

- де F, G - умови, P – інструкція.
- При цьому умова F називається передумовою інструкції P, а G - післяумовою P.

Будемо записувати Хоарівську трійку наступним чином

$\#\{F\}$

P

$\#\{G\}$

Хоарівська трійка.2

Хоарівська трійка **справджується**, якщо за умови істинності *F* до виконання інструкції *P*, умова *G* буде істинною після виконання *P*.

Приклад Хоарівської трійки, яка справджується:

```
#{x == 1}
x = x + 2
#{x == 3}
```

Властивості циклу з умовою продовження

а) Цикл рівносильний такому розгалуженню

```
while F:      ≡      if F:
    P                    P
                         while F:
                             P
```

Властивості циклу з умовою продовження

b) Справджується трійка

```
#{True}
while F:
    P
#{not F}
```

Приклад: обчислення факторіалу

Обчислити значення $n!$ При заданому n.

$n! = n(n-1)\ldots 1$

$0! = 1$

Приклад: обчислення елементів послідовностей

Обчислити $(-1)^n \frac{x^n}{n!}$ при заданих *x* та *n*.

Приклад: Обчислення сум

Обчислити суму $1 - x + \frac{x^2}{2} - \ldots + (-1)^n \frac{x^n}{n!}$

Приклад: Числа Фібоначчі

Обчислити задане число Фібоначчі.

Числа Фібоначчі задаються співвідношеннями

$$\begin{cases} f_0 = 1,\ f_1 = 1, \\ f_k = f_{k-2} + f_{k-1}, k = 2,3,\ldots \end{cases}$$

Команди break та continue

У циклі while можуть застосовуватись команди

`break`

та

`continue`

Якщо Python зустрічає

`break`

то він перериває виконання циклу.

Якщо Python зустрічає

`continue`

то він пропускає всі команди до кінця циклу та переходить до наступного кроку циклу.

Команди break та continue.2

Команда

`break`

зокрема використовується для реалізації так званих циклів з післяумовою та з виходом.

У цих циклах питання виходу з циклу вирішується не на початку циклу, а в його кінці (або всередині циклу).

Команди break та continue.3

Такий цикл має вигляд:

```
while True:
    P
    if F: break
    Q
```

- При цьому, Q може бути і тотожною командою.

Повний синтаксис while

Повний синтаксис циклу while передбачає також можливість використання else після кінця циклу.

```
while F:
    P
else:
    Q
```

Інструкція *P* може містити, в тому числі, break та continue.

Інструкція Q буде виконуватись у випадку нормального завершення циклу. Якщо ж вихід з циклу здійснюється за допомогою break, то Q не буде виконуватись.

Наближене обчислення e^x

$$e^x = 1 + x + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!} + \ldots$$

Відомо, що загальний член цього ряду прямує до 0. Обчислимо наближено e^x як суму цього ряду.

Позначимо загальний член ряду через a_n, а суму, - через b_n.

Будемо вважати точність обчислення задовільною, якщо модуль загального члену ряду менше деякого малого ε, тобто, $|a_n| < \varepsilon$

3.2 Цикл по діапазону значень

Цикл по діапазону значень

У Python є також цикл, який виконується задану кількість разів.

При цьому, визначається спеціальна змінна, яка називається лічильником циклу і яка пробігає визначену послідовність значень.

Розглянемо цей цикл спочатку для послідовностей цілих чисел.

Така послідовність повинна бути арифметичною прогресією:

$a_1 = b,\ a_2 = a_1 + d,\ \ldots,\ a_n = a_{n-1} + d,\ \ldots$

Для обмеження кількості повторень циклу встановлюють границю c таким чином, що лічильник пробігає значення всіх елементів послідовності $\{a_n\}$ з напівінтервалу $[b,c)$, при $d > 0$ (напівінтервалу $(c,b]$ при $d < 0$).

Об’єкт range

У Python ця послідовність (прогресія) задається спеціальним об’єктом

```
range(b,c,d)
```

Якщо *d* = 1, то *d* можна опустити і писати

```
range(b,c)
```

Якщо, крім цього, *b* = 0, то *b* також можна опустити і писати

```
range(c)
```

Цикл for

Синтаксис циклу for

```
for i in range(b,c,d):
    P
```

де *b, c, d* – цілі вирази (*d* != 0), *P* – інструкція.

Цикл for.2

Правило виконання циклу for ($d > 0$)

```
for i in range(b,c,d):   ≡   if b < c:
    P                            i = b
                                 while True:
                                     P
                                     if i + d >= c: break
                                     i = i + d
```

Цикл for.3

При $d < 0$ змінюються знаки двох відношень у правилі виконання циклу for (< на >, а >= на <=):

```
for i in range(b,c,d): ≡  if b > c:
    P                         i = b
                              while True:
                                  P
                                  if i + d <= c: break
                                  i = i + d
```

Приклади

Обчислити n! За допомогою циклу for

Обчислити n!! = n(n-2)(n-4)…( )

2

1

Резюме

Ми розглянули:

1. Поняття циклу та циклічної програми
2. Цикл з умовою продовження, його властивості.
3. Обчислення елементів послідовностей та сум.
4. Повний синтаксис циклу за умовою, обчислення границь
5. Цикл по діапазону значень.

Де прочитати

1. Обвінцев О.В. Інформатика та програмування. Курс на основі Python. Матеріали лекцій. – К., Основа, 2017
2. A Byte of Python (Russian) Версия 2.01 Swaroop C H (Translated by Vladimir Smolyar), http://wombat.org.ua/AByteOfPython/AByteofPythonRussian-2.01.pdf
3. Бублик В.В., Личман В.В., Обвінцев О.В.. Інформатика та програмування. Електронний конспект лекцій, 2003 р., http://www.matfiz.univ.kiev.ua/books
4. Марк Лутц, Изучаем Python, 4-е издание, 2010, Символ-Плюс
5. Самоучитель Python. http://pythonworld.ru/samouchitel-python
6. С. Шапошникова. Основы программирования на Python. Версия 2 (2011). https://younglinux.info/python.php
7. Python 3.4.3 documentation