Universiteit van Stellenbosch
Departement Wiskundige Wetenskappe
Afdeling Rekenaarwetenskap

University of Stellenbosch
Department of Mathematical Sciences
Computer Science Division

Wetenskaplike Berekening 272 / Scientific Computing 272

Tutoriaal 6: Herhaling in Python / Tutorial 6: Choices in Python

2020-08-07/13 Opgestel deur Willem Bester Gemodereer deur Brink van der Merwe

Agtergrond

Dié tutoriaal is 'n praktiese oorsig van herhaling in Python. Ter voorbereiding, bestudeer die sesde reeks Python-skyfies. Ek stel voor dat u die gids `~/wb272/tut06` skep vir u werk aan dié tutoriaal.

Background

This tutorial is a practical overview of repetition in Python. To prepare, study the sixth set of Python slides. I suggest you create the directory `~/wb272/tut06` for your work on this tutorial.

Uitkomste

Wanneer u die tutoriaal voltooi het, behoort u in staat te wees om die volgende te doen: (1) `for`- en `while`-lusse te skryf om stellings te herhaal in Python, (2) eenvoudig algoritmes uit te dink vir probleme waarvan die oplossings herhaling behels, (3) wiskundige formules na Python te vertaal, (4) patrone te ontleed vir herhalende strukture, en (5) gegee die teoretiese agtergrond, om kode te skryf wat wiskundige funksies numeries benader.

Outcomes

When you have complete the tutorial, you should be able to do the following: (1) write `for` and `while` loops to repeat statements in Python, (2) to invent algorithms for solutions of problems that entail repetition, (3) translate mathematical formulae to Python, (4) to analyse patterns for repeating structures, and (5) given the theoretical background, write code that approximates mathematical functions numerically.

1. Gebruik 'n `for`-lus en vertoon die heelgetalle van 33 tot 47 (albei ingesluit) op 'n enkele reël.

   Use a `for` loop and display the integers from 33 to 47 (both included) on a single line.

2. Gebruik 'n `while`-lus en vertoon die heelgetalle van 33 tot 47 (albei ingesluit) op 'n enkele reël.

   Use a `while` loop and display the integers from 33 to 47 (both included) on a single line.

3. Skryf 'n Python-funksie `print_as_vector` wat een lys `L` as parameter neem en dit dan in vektorvorm met hoekhakies vertoon, met elemente van mekaar geskei deur kommapunte. Byvoorbeeld, gestel `L` is $[17, 2, 5]$; dan moet die lys soos hieronder vertoon word. Let op dat daar geen spasies na enige getal is nie en ook dat daar een minder komma as getalle is.

   Write a Python function `print_as_vector` that takes one list `L` as parameter, and then, displays it in the vector form with angular brackets and elements separated by semicolons. For example, if `L` is $[17, 2, 5]$, the list must be displayed as given below. Note that there are no spaces after any number and also that there is one fewer comma than there are numbers.

   ```
   <17, 2, 5>
   ```

4. Bereken die gemiddelde van die heelgetalle vanaf 2 tot 22 (albei ingesluit). Gebruik 'n lus om die totaal te vind en bereken daarná die gemiddelde.

   Calculate the average of the integers in the range 2 to 22 (both included). Use a loop to find the total, after which you calculate the average.

5. Wanneer ons data analiseer, is dit gewoonlik nie genoeg om slegs die gemiddeld van die steefproef te weet nie. Ons benodig ook 'n idee van hoe uitgesprei die metings is, wat beteken ons moet die variansie of sy vierkantswortel, die standaardafwyking, bereken. Alhoewel ons die variansie tipies definieer as

When analysing data, it usually not enough to know just the mean of the sample. We also require a notion of how spread out the measurements are, which means calculating the variance, or the square root thereof, the standard deviation. Although we typically define the variance as

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2\,, \tag{1}$$

waar N die steekproefgrootte, μ die steekproefgemiddeld en x_i die $i$de meting, vir $1 \le i \le N$, is, is die volgende vergelyking nuttiger wanneer ons 'n funksie wil skryf om die variansie vinnig te bereken:

where N is the sample size, μ is the sample mean, and x_i is the ith measurement, for $1 \le i \le N$, the following equation is more useful when we want to write a function to calculate the variance rapidly:

$$\sigma^2 = \frac{\sum_{i=1}^{N} x_i^2}{N} - \mu^2. \tag{2}$$

Skryf nou 'n Python-funksie `stdev` wat een lys as parameter neem en die standaardafwyking van die waardes daarin terugstuur. Wenk: Gebruik Vgl. (2) en skryf een lus wat gelyktydig die som van die waardes en die som van die kwadrate van die waardes bereken.

Now write a Python function `stdev` that takes one list as parameter and returns the standard deviation of its values. Hint: Use Eq. (2) and write one loop that simultaneously calculates the sum of the values and the sum of the squares of the values.

6. Skryf 'n Python-funksie `triangle` wat een positiewe heelgetal `n` as parameter neem en dan 'n driehoekige rangskikking van X-karakters uitdruk soos hieronder geïllustreer. Die parameter `n` spesifiseer die aantal reels in die driehoek. Gebruik een `for`-lus en die string-vermenigvuldigingsoperator. Die volgende voorbeeld is afvoer vir die roep `triangle(7)`. Let op dat die eerste "`X`" op elke reël aan die heel linkerkant van die terminaalvenster vertoon word.

Write a Python function `triangle` that takes a single, positive integer as parameter `n` and then displays a triangular arrangement of X characters on the screen, as illustrated below. The parameter `n` specifies the number of lines in the triangle. Use a single `for` loop and the string multiplication operator. The following example is output for the call `triangle(7)`. Note that the first "`X`" on each line is displayed at the very left of the terminal window.

```
X
XX
XXX
XXXX
XXXXX
XXXXXX
XXXXXXX
```

7. Doen Vraag 6 oor, maar gebruik nou geneste `for`-lusse in plaas van die string-vermenigvuldigingsoperator binnein 'n `for`-lus. Onthou, daar mag nie enige spasies tussen X'e op dieselfde reël wees nie. Wenk: Gebruik stringkatenering in die binneste lus.

Do Question 6 again, but now use a nested `for` loop instead of the string multiplication operator inside a `for` loop. Remember, there may not be any spaces between X's on the same line. Hint: Use string concatenation in the innermost loop.

8. Doen Vraag 6 weer oor, maar gebruik nou geneste `while`-lusse.

Repeat Question 6 again, but now use nested `while` loops.

9. Skryf 'n Python-funksie `parallelogram` wat 'n enkele nie-leë string as parameter neem en dié string horisontaal en veelvuldige kere vertikaal uitdruk sodat elke letter in die horisontale string ooreenstem met die vertikale posisie van daardie letter. Die horisontale string moet in die middel van die afvoer wees. Elke vertikale string moet van bo af ondertoe gelees word. Die eerste vertikale string (van links af) gebruik die eerste karakter van die horisontale string, die tweede vertikale string gebruik die tweede karakter van die horisontale string, en so voorts. By voorbeeld, die woord BILTONG moet soos volg uitgedruk word.

Write a Python function `parallelogram` that takes a single non-empty string parameter and prints this string horizontally and multiple times vertically so that each character in the horizontal string matches the position of that character vertically. The horizontal string must be in the centre of the output. Each vertical string is to be read downwards from the top. The first vertical string (from the left) uses the first character of the horizontal string, the second vertical string uses the second character of the horizontal string, and so on. For example, the word BILTONG must be output as follows.

```
      B
     BI
    BIL
   BILT
  BILTO
 BILTON
BILTONG
ILTONG
LTONG
TONG
ONG
NG
G
```

10. UITDAGING: Probeer om Vraag 9 op soveel verskillende maniere op te los. Kyk of u dit kan regkry om slegs een lus te gebruik, sonder `if`-stellings.

CHALLENGE: Try to solve Question 9 in as many different ways possible. See if you can manage to use only one loop, without any `if` statements.

11. Twee getalle a en b staan in die **sectio aurea** ("goue snee") ϕ wanneer die volgende waar is:

Two numbers a and b are said to be in the **golden ratio** ϕ if the following is true:

$$\frac{a+b}{a} = \frac{a}{b} = \phi. \tag{3}$$

Vir duisende jare al fassineer ϕ mense uit sulke diverse dissiplines soos wiskunde, musiek, biologie en argitektuur. Deur basiese algebra en die kwadratiese formule op Vgl. (3) toe te pas, kry ons

For thousands of years, ϕ has fascinated people in fields as diverse as mathematics, music, biology, and architecture. Using some basic algebra and the quadratic formula on Eq. (3) yields

$$\phi = \frac{1+\sqrt{5}}{2} = 1.6180339887498948482045868343656381177203091798058\ldots, \tag{4}$$

wat 'n irrasionale getal is. Daarom kan ons ϕ nie presies op 'n rekenaar voorstel nie. Gebruik ons egter die ewebekende Fibonacci-reeks,

which is an irrational number. Therefore, we cannot represent ϕ exactly on a computer. However, using the equally well-known Fibonacci sequence,

$$\{F_n\} = 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, \ldots, \tag{5}$$

wat volg vanuit die rekursierelasie

which is formed by the recurrence relation

$$F_n = F_{n-1} + F_{n-2}, \quad F_0 = 0, F_1 = 1, \tag{6}$$

kan ons ϕ arbitrêr akkuraat benader. Dít is moontlik omdat, soos Kepler oorspronklik getoon het,

we can approximate ϕ arbitrarily close. This is possible since, as originally shown by Kepler,

$$\lim_{n\to\infty} \frac{F_{n+1}}{F_n} = \phi. \tag{7}$$

Gebruik nou Vgl. (7) en skryf 'n Python-funksie `phi` wat 'n heeltegal `d` > 1 as parameter neem en wat ϕ akkuraat tot `d` desimale syfers na die punt terugstuur. Indien u sukkel om te begin, probeer eers om 'n lys te bou wat die Fibonacci-getalle bevat, tot 'n sekere n soos in Vgl. (5). Let dan op, volgens Vgl. (6) benodig u 'n "geskiedenis" van slegs twee getalle om die volgende getal in die reeks te bereken. Dit wil sê, dit is onnodig om die hele lys van Fibonacci-getalle aan te hou, nóg om hulle te bereken, nóg om die verhouding tussen twee opvolgende getalle uit te werk. Hou die berekening vol totdat twee opeenvolgende *verhoudings* (nie opeenvolgende getalle nie) naby genoeg aanmekaar is.

Now use Eq. (7), and write a Python function `phi` that takes an integer `d` > 1 as parameter and that returns ϕ accurate to `d` decimal digits after the point. If you struggle to decide where to begin, first try creating a list that contains the Fibonacci numbers up to some n as in Eq. (5). Then, from Eq. (6), observe that you need a "history" of only two numbers to compute the next number in the sequence. That is, it is not necessary to keep the whole list of Fibonacci numbers, either to compute them or to work out the ratio of consecutive numbers. Continue with the computation until two consecutive *ratios* (not consecutive numbers) are close enough.

12. Omskep u antwoord op Vraag 11 in 'n alleenstaande program wat direk van die bevelreël geloop kan word. Eksperimenteer met verskillende parameters en vergelyk u antwoorde met die eerste 50 beduidende syfers van ϕ wat in Vgl. (4) gegee word. Hoe vinnig (hoeveel iterasies) konvergeer u berekening tot die maksimum akkuraatheid wat deur "gewone" Python toegelaat word? Gebruik die Internet en soek vir hoe u meer akkurate wisselpuntgetalle in Python kan kry.

Turn your answer to Question 11 into a standalone program, that is, one that you can run directly from the command line. Experiment with passing different parameters, and compare the answers with the first 50 significant digits of ϕ, given in Eq. (4). How fast (how many iterations) does the calculation converge to the maximum accuracy allowed by "normal" Python? Use the Internet, and search for how to get more accurate floating-point numbers in Python.