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 1: Inleiding tot Python / Tutorial 1: Introduction to Python

2020-02-28/03-05 Opgestel deur Willem Bester Gemodereer deur Brink van der Merwe

Agtergrond

Dié tutoriaal is 'n praktiese inleiding tot die Python-programmeertaal en -interpreteerder. Ter voorbereiding, bestudeer die eerste reeks Python-skyfies. Ek stel voor dat u die gids `~/wb272/tut01` skep vir u werk van dié tutoriaal. Onthou om `CMD cd ~/wb272/tut01` uit te voer voordat u enigiets stoor.

Background

This tutorial is a practical introduction to the Python programming language and interpreter. To prepare, study the first set of Python slides. I suggest you create the directory `~/wb272/tut01` for your work for this tutorial. Remember to execute `CMD cd ~/wb272/tut01` before you save anything.

Werkopdrag

Werkopdrag 1 sal slegs uit twee praktiese vrae bestaan. Maak dus seker dat u Vrae 4, 9, 10 en 11 van dié tutoriaal goed verstaan en voltooi.

Assignment

Assignment 1 will consist of two practical questions only. Therefore, ensure that you understand and complete Questions 4, 9, 10, and 11 of this tutorial.

Uitkomste

Wanneer u die tutoriaal voltooi het, behoort u met die volgende onderwerpe gemaklik te wees: (1) evaluering van Python-uitdrukkings per hand, (2) veranderliketoekenning en -gebruik, (3) ingeboude Python-funksies, en (4) die skryf van (nie-vertakkende) Python-funksies.

Outcomes

When you have completed the tutorial, you should be comfortable with the following topics: (1) evaluating Python expressions by hand, (2) variable assignment and use, (3) built-in Python functions, and (4) writing (non-branching) Python functions

Opstelling en gebruik van gedit

Ons gebruik 'n program genaamd 'n redigeerder om Python-programme in te tik. Die verstekredigeerder op Ubuntu word **gedit** genoem. Om die lêer `program.py` oop te maak en te redigeer, tik eenvoudig die volgende op die bevelreël:

Setup and use of gedit

We use a program called an **editor** to type in Python programs. The default editor in Ubuntu is called **gedit**. To open and edit the file `program.py`, simply type the following on the command line:

```
12345678@h00:~/wb272/tut01$ gedit program.py & ⟨Enter⟩
```

Onthou: Alles tot en met die dollar is die **bevelpor**, en 'n mens tik dit nie in nie. Let ook op die ampersand ("&") wat getik moet word voordat u ⟨Enter⟩ druk. Dit forseer 'n bevel om in die agtergrond te loop: 'n Proses wat in die agtergrond loop, is nie gekoppel aan die terminaal nie, alhoewel daar in hierdie geval 'n gedit-venster op u werkskerm sal wees. Indien u nie gedit in die agtergrond loop nie, sal u nie daartoe in staat wees om enigiets verder in die terminaal uit te voer nie.

Remember: Everything up to and including the dollar is the **command prompt**, and you do not type it in. Also note the ampersand ("&") to type before you press ⟨Enter⟩. This forces a command to run in the background: A process that runs in the background is not attached to the terminal, although in this case, there will be a gedit window on your desktop. If you do not run gedit in the background, you will not be able execute anything further in the terminal.

In gedit kan u die *Edit → Preferences* menukeuse gebruik om formattering op te stel. In die oop *Preferences*-venster, kliek op die *Editor*-oortjie, en maak seker dat (1) die inkeepwydte ("tab width") op 4 gestel is, en dat die keuses (2) "Insert spaces instead of tabs" en (3) "Enable automatic indentation" gekies is. Om 'n oop lêer te stoor, gebruik die kortpad ⟨Ctrl⟩-S, en om gedit te verlaat, gebruik ⟨Ctrl⟩-Q.

Baie belangrik: Die idee is dat u u Python-kode direk in gedit intik. Kom ons gebruik die `to_celsius`-funksie op die skyfies ter illustrasie. Om hierdie funksie in 'n lêer genaamd `temperature.py` in te voer, voer eerstens CMD `gedit temperature.py &` op die bevelreël uit, en tik dan die volgende in gedit:

Inside gedit, you can use the *Edit → Preferences* menu option to set up formatting. In the open *Preferences* window, click on the *Editor* tab, and ensure that (1) the tab width is set to 4, and that (2) "Insert spaces instead of tabs" and (3) "Enable automatic indentation" are selected. To save an open file, use the shortcut ⟨Ctrl⟩-S, and to quit gedit, use ⟨Ctrl⟩-Q.

Very important: The idea is that you type your Python code directly into gedit. Let's use the `to_celsius` function on the slides as illustration. To save this function in a file called `temperature.py`, run CMD `gedit temperature.py &` on the command line, and then type the following into gedit:

```
def to_celsius(t):
    return (5 / 9) * (t - 32)
```

Vervolgens, stoor die lêer, en loop die Python-interpreteerder.

Next, save the file, and run the Python interpreter.

```
12345678@h00:~/wb272/tut01$ python3
Python 3.7.2+ (heads/3.7:7e618f3154, Feb 14 2019, 22:48:45)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import temperature          # NOTE: No .py extension
>>> temperature.to_celsius(80)
26.666666666666668
```

Indien u nou enige veranderinge aan `temperature.py` maak terwyl die Python-interpreteerder loop, sal dié veranderinge nie in die interpreteerder gereflekteer word tensy dit herbegin word nie. Wanneer ons Python-modules van nader bekyk later, gaan ons sien hoe om modules te herlaai, sonder om die interpreteerder oor te begin.

If you now make any changes to `temperature.py` while the Python interpreter is running, these changes won't be reflected in the interpreter unless it is restarted. When we consider Python modules up close later, we will see how to reload modules without restarting the interpreter.

Oefeninge

Exercises

1. Evalueer elkeen van die volgende Python-uitdrukkings *per hand*. U kan u antwoorde nagaan deur elke uitdrukking in die Python-interpreteerder te evalueer.
   (a) `9 - 3`
   (b) `8 * 2.5`
   (c) `9 / 2`
   (d) `9 / -2`
   (e) `9 % 2`
   (f) `9 % -2`
   (g) `-9 // 2`
   (h) `9 // -2.0`
   (i) `4 + 3 * 5`
   (j) `(4 + 3) * 5`

   Evaluate each of the following Python expressions *by hand*. You can verify your answers by evaluating each expression in the Python interpreter.

2. (a) Skep 'n veranderlike `temperature` (in die Python-interpreteerder) wat die waarde 24 bevat.

   Create a variable `temperature` (in the Python interpreter) that contains the value 24.

(b) Skakel nou dié temperatuur van grade Celsius na grade Fahrenheit om, en assosieer die gevolglike waarde met die veranderlike `temperature`. Wat is die nuwe waarde van laasgenoemde?

Now, convert this temperature from degrees Celsius to degrees Fahrenheit, and associate the resulting value with the variable `temperature`. What is the new value of the latter?

3. Bereken die volgende *per hand*, en gaan dan u antwoorde in die Python-interpreteerder na.

   (a) Skep 'n nuwe veranderlike `x`, en ken die waarde 10.5 daaraan toe.

   (b) Skep nog 'n nuwe veranderlike, `y`, en ken die waarde 4 daaraan toe.

   (c) Sommeer `x` en `y`, en assosieer die gevolglike waarde met `x`. Wat is die nuwe waardes van `x` en `y`?

   Compute the following *by hand*, and then verify your answers in the Python interpreter.

   Create a new variable `x`, and assign the value 10.5 to it.

   Create another new variable, `y`, and assign the value 4 to it.

   Sum `x` and `y`, and associate the resulting value with `x`. What are the new values of `x` and `y`?

4. Skryf 'n Python-funksie `celsius_to_fahrenheit` wat 'n temperatuur in grade Celsius na grade Fahrenheit omskakel. Gebruik u antwoord op Vraag 2(b) om mee te begin. Die funksie neem een wisselpuntgetal as parameter: die temperatuur in grade Celsius. Die funksie stuur een wisselpuntgetal as antwoord terug: die temperatuur in grade Fahrenheit. Stoor u funksie in 'n lêer genaamd `temperature.py`. Laai die argief van toetslêers, wat op die modulewebwerf beskikbaar is, af en onttrek die lêer `test_temperature.py` uit die argief om u werk te toets. Om die toetsprogram te loop, kopieer dit in dieselfde gids as u antwoordprogram en doen dan die volgende op die bevelreël:

   Write a Python function `celsius_to_fahrenheit` to convert a temperature in degrees Celsius to degrees Fahrenheit. Use your answer to Question 2(b) to start. The function takes one floating-point number as parameter: the temperature in degrees Celsius. It returns one floating-point number as answer: the temperature in degrees Fahrenheit. Save your function to a file called `temperature.py`. Download the test file archive from the module website, and extract the file `test_temperature.py` from the archive to test your work. To run the test program, copy it to the same directory as your answer program, and then do the following on the command line:

```
12345678@h00:~/wb272/tut01$ python3 test_temperature.py
```

5. Maak 'n lys wat puntsgewys gee wat gebeur wanneer Python die stelling `x += x - x` evalueer waar `x` aanvanklik die waarde 3 het.

   Give an itemised list of what happens when Python evaluates the statement `x += x - x` where `x` initially has the value 3.

6. Verduidelik kortliks wat die verskil tussen 'n parameter en 'n argument is.

   Briefly explain the difference between a parameter and an argument.

7. Gebruik die Python-interpreteerder se `help`-opdrag en hersien die ingeboude funksies `abs`, `round`, `pow`, `int` en `float`. Gebruik nou dié funksies en skryf nou een Python-uitdrukking vir elkeen van die volgende:

   (a) Bereken 3^7.

   (b) Skakel 34.7 na 'n heeltegal toe om deur afkapping.

   (c) Skakel 34.7 na 'n heelgetal toe om deur afronding.

   (d) Neem die absolute waarde van −86, en skakel dié dan om na 'n wisselpuntgetal.

   Use the `help` command of the Python interpreter and review the built-in functions `abs`, `round`, `pow`, `int`, and `float`. Using these functions, now write one Python expression for each of the following:

   Calculate 3^7.

   Convert 34.7 to an integer by truncating.

   Convert 34.7 to an integer by rounding.

   Take the absolute value of −86, and then convert it to a floating-point number.

8. Gebruik die Python-interpreteerder en u oortuig uself dat aftrekking **links-assosiatief** is. Byvoorbeeld, `a - b - c` beteken $(a - b) - c$; dis beslis *nie dieselfde* as $a - (b - c)$ *nie*.

   Use the Python interpreter and convince yourself that subtraction is **left associative**. For example, `a - b - c` means $(a - b) - c$; it is definitely *not the same* as $a - (b - c)$.

9. Skryf 'n Python-funksie vir die berekening van elkeen van die volgende klassieke bewegingsvergelykings:

   Write a Python function for the calculation of each of the following classical equations of motion:

$$v = v_0 + at, \tag{1}$$

$$s = s_0 + v_0 t + \frac{1}{2} a t^2, \tag{2}$$

$$v^2 = v_0^2 + 2a\Delta s. \tag{3}$$

In dié vergelykings verwys die veranderlikes v, s, a en t na die snelheid, verplasing, versnelling en tyd, onderskeidelik. Noem u funksies `velocity`, `distance` en `velocity2`, en maak seker dat u telkens die relevante parameters spesifiseer. In die geval van Vgl. (3) moet u funksie die snelheid (en nie die kwadraat van die snelheid nie) terugstuur. Voorsien u enige probleme met dié benadering?

In the equations, the variables v, s, a, and t refer to the velocity, displacement, acceleration, and time, respectively. Name your functions `velocity`, `distance`, and `velocity2`, and ensure that you specify the relevant parameters in each case. In the case of Eq. (3), your Python function must return the velocity (and not the square of the velocity). Can you foresee any problems with this approach?

10. Skryf 'n Python-funksie `elapsed_minutes` wat twee tye in militêre formaat—byvoorbeeld, 0830 is halfnege in die oggend en 1745 is kwart voor ses in die aand—as parameters neem, en die aantal minute wat tussen dié twee tye verstryk het, terugstuur. Neem aan dat albei toevoertye geldig is, dat albei tye op dieselfde dag val, en dat die tweede tyd gelyk aan of later as die tweede tyd is. *Toevoer word as stringe gespesifiseer.* Byvoorbeeld, die funksieroep `elapsed_minutes("0830","1745")` stuur die heelgetal 555 terug. Stoor u funksie in 'n lêer genaamd `duration.py`. Vanuit die toetsargieflêer, onttrek `test_duration.py` om u werk te toets. Verwys na Vraag 4 vir hoe om die toetsprogram te loop, en maak seker dat `test_duration.py` in dieselfde gids as `duration.py` is. WENKE: Die funksie `int` kan meer doen as bloot 'n wisselpuntgetal na 'n heelgetal omskakel. Verwys na Python se aanlyn-hulp deur die `help`-funksie te roep.

Write a Python function `elapsed_minutes` that takes two times in military format—for example, 0830 is half past eight in the morning, and 1745 is a quarter to six in the evening—as parameters, and returns the number of minutes that have elapsed between these two times. Assume both input times are valid and fall on the same day, and the second time is equal to or later than the first. *Input are specified as strings.* For example, the function call `elapsed_minutes("0830","1745")` returns the integer 555. Save your function to a file called `duration.py`. From the test archive file, extract `test_duration.py` to test your work. See Question 4 for how to run the test file, and ensure `test_duration.py` is in the same directory as `duration.py`. HINTS: The function `int` can do more than turn a floating-point number into an integer. For help, call Python's `help` function.

11. Skryf 'n Python-funksie `day_of_the_week` wat 'n datum as toevoer neem en die weeksdag waarop daardie datum val, terugstuur. Die funksie moet drie parameters y (jaar), m (maand) en d (dag) neem, in dié volgorde. Vir m, gebruik 1 vir Januarie, 2 vir Februarie, en so voorts. Vir die terugkeerwaarde, aanvaar 0 vir Sondag, 1 vir Maandag, en so voorts tot en met 6 vir Saterdag. Gebruik die volgende formules vir die Gregoriaanse kalender:

Write a Python function `day_of_the_week` that takes a date as input and returns the day of the week on which that date falls. The function must take three parameters y (year), m (month), and d (day), in this order. For m, use 1 for January, 2 for February, and so on. For the return value, accept 0 for Sunday, 1 for Monday, and so on up to 6 for Saturday. Use the following formulae for the Gregorian calendar:

$$y_0 = y - (14 - m)//12, \tag{4}$$

$$x = y_0 + y_0//4 - y_0//100 + y_0//400, \tag{5}$$

$$m_0 = m + 12 \times ((14 - m)//12) - 2, \tag{6}$$

$$d_0 = (d + x + (31 \times m_0)//12) \bmod 7, \tag{7}$$

waar d_0 die terugkeerwaarde is. Byvoorbeeld, die funksieroep `day_of_the_week(2012,2,29)` moet die heelgetal 3 terugstuur, wat aandui dat 29 Februarie 2012 op 'n Woensdag geval het. Stoor u funksie in 'n lêer genaamd `days.py`. Vanuit die toetsargieflêer, onttrek die lêer `test_days.py` om u werk te toets.

where d_0 is the return value. For example, the function call `day_of_the_week(2012,2,29)` must return the integer 3, indicating that 29 February 2012 fell on a Wednesday. Save your function to a file called `days.py`. From the test archive file, extract the file `test_days.py` to test your work.