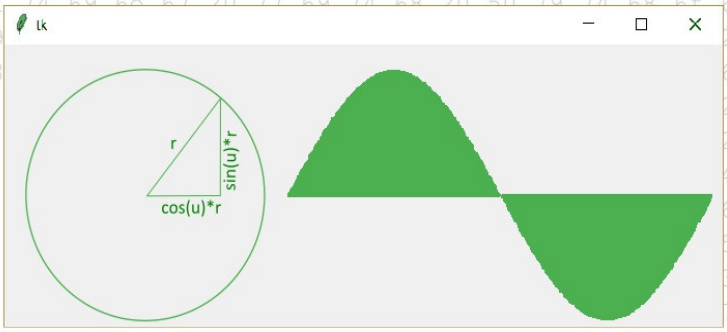
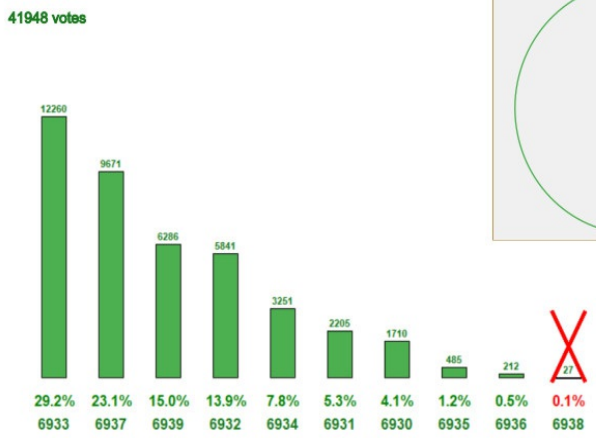
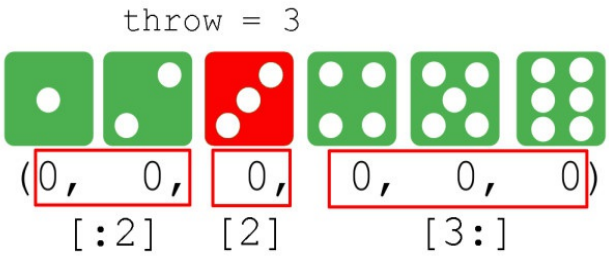
Creating with Python

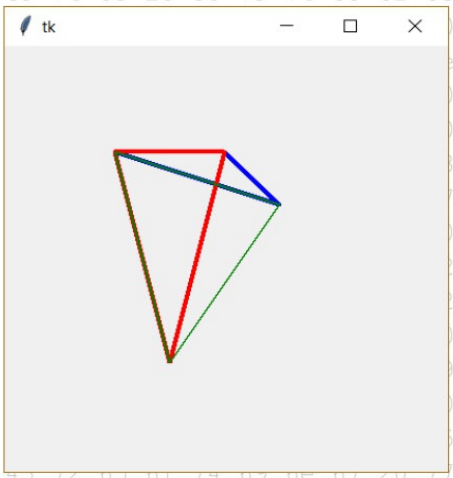
Peter Kučera, Jaroslav Výbošťok

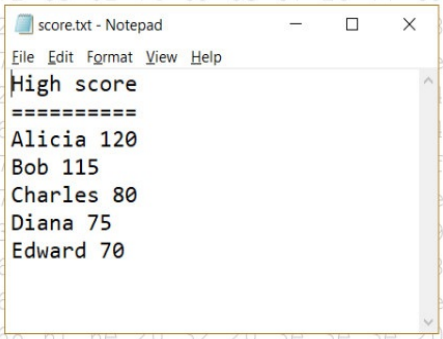
Vol. 2

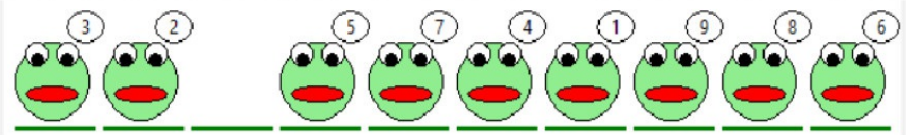












www.creatingwithpython.com

Creating with Python vol. 2

Authors © Mgr. Peter Kučera, Mgr. Jaroslav Výbošťok

Design © Mgr. Peter Kučera

Translation: Michal Čermák, Joshua Ruggiero

First published, 2019

Version number: 20191028

Publisher: Peter Kučera

eBook preview

Visit the author's website at www.creatingwithpython.com

ISBN 978-80-570-1239-9 (pdf)

ISBN 978-80-570-1240-5 (epub)

ISBN 978-80-570-1241-2 (mobi)

What others have written about our e-books ...

The textbook Creating with Python (textbook for secondary schools) quickly gained a place in the teaching programming throughout Slovakia thanks to the ease of access to a full range of programming tools introduced through the tkinter graphical environment. This approach promotes understanding of programming techniques by visualizing results in an interactive or program mode, directly encouraging students to experiment with modifying a number of graded non-mathematical tasks, and allowing an individual pace of study for both girls and boys, giving students the joy of learning. I highly recommend it to all those between the ages of 9 and 99 interested in breaking into the secrets of programming, not just high school students. For the first time in my many years as a computer science teacher, I received a methodological guide to the textbook, which will help not only teachers but also self-learners.

RNDr. Eva Hanulová,
Gymnázium Jura Hronca, Bratislava

I look forward to every new textbook, and there is just a handful of those suitable for teaching high school programming. That is why my students and I were very pleased with this new book. I also very much appreciate the teacher's guide and tests, which are absolutely perfect - there are other tasks and suggestions for teaching. I have been looking forward to the second volume since I learned that the authors were preparing it.

RNDr. Eva Stanková,
Gymnázium Ivana Horvátha, Bratislava

I greatly appreciate the Creating with Python textbook. It offers students a clear way to get into programming. Students can also work with the textbook at their own pace. Another great contribution is also the handbook for teachers, where the whole methodology from the educational plans through to the well-arranged curriculum with solved tasks and practical methodological notes on each task is elaborated. The set of tests is also valuable.

RNDr. Oľga Poliaková, SPŠ, Bardejov

Compared to other subjects, informatics is a subject for which a large part of the curriculum varies from year to year at a fairly rapid pace, putting pressure on teachers to devote much of their preparation to self-study. Based on state “support” (missing or out-of-date textbooks, old hardware, ...), an IT teacher discovers that IT is not a key subject in this society. This textbook has saved me a lot of time and energy that I can devote to other school activities. I appreciate the teacher's guide with its solved textbook assignments. Many times I have found other solutions than mine in the handbook which I could introduce to students and thus help push their thinking to the next level. I think that the textbook has improved the teaching of computer science at our school.

Mgr. Peter Nemec,
Spojená škola sv. Vincenta de Paul, Bratislava

The textbook Creating with Python is an excellent tool for computer science teachers in teaching programming. The methodology of using graphical tasks in Python is suitable for illustrating the basics of programming. A benefit of the textbook is the methodological guide for teachers and the set of tests. The textbook will be used in preparing future computer science teachers, who can verify it in their teaching practice at the training schools.

Ing. Janka Majherová, PhD., Katedra informatiky,
Pedagogická fakulta, Katolícka univerzita v Ružomberku

Contents

Introduction

We are glad that this electronic textbook of informatics for high schools, Creating with Python, has found its fans. The first volume was a basic programming course for all secondary school students and fulfilled the State Educational Program. The second volume is a continuation not only for students who want to study deeper or prepare for graduation from computer science, but also for other candidates or self-learners.

The textbook meets the requirements for graduation from computer science according to the Target Requirements for Knowledge and Skills in the area of Algorithmic problem solving, which has been valid since the 2018/2019 school year. In addition to the topics (chapters) that are needed for the school-leaving exam, you will also find topics in the textbook that complement this content and help you understand important context in Python, appropriate to the school-leaving (or intermediate) level. These topics will help you create more complex and interesting programs.

In the textbook you will find a lot of practical examples and solutions, exercises to practice, but also questions to encourage you to think, discover the context, discuss in a group, experiment, and also look for errors and intuitively search for an optimal solution.

Chapter 10 focuses on customizing the appearance of applications - the controls from the tkinter library. It contains practical demonstrations for using widgets (controls), but it is not necessary to master all of this knowledge; rather, it is a quick guide to using them. Therefore, this chapter does not include tasks and questions.

After studying this textbook and completing the hours of practical programming, you should be able to solve intermediate-level tasks using composite data, and you should also be able to program a more complex, larger-scale program, such as a seminar work.

While this e-book is in fact just a sequence of zeros and ones, we believe that the interesting things will prevail in that sequence and this textbook will become your go-to for studying programming.

We wish you pleasant moments in programming :)
The authors

1 Text strings

1.1 Numbers vs. text strings, reading input

In the past we worked with character strings and generated random sequences. In this chapter we will look at text strings more closely. So far we can:

- join strings using the operation `+`

```
word = 'Py' + 'thon'
print(word)
```

- use the `int()` function, which converts a string to a number (if it contains only characters that can be part of a number). We used it when entering input using `entry`, for example `number = int(entry1.get())`.

-
```
a = '1250'
b = int(a)
b = b + 10
print(b)
```

If we wanted to display `canvas.create_text(100, 200, text='Point count:' + points)`, we had to convert `points` (a number) to text. So far, we have been able to solve this using one command to print the text and another to print the number. In such cases, we can use the `str()` function, which converts a number to a text string.

So the aforementioned statement would look like this: `canvas.create_text(100, 200, text='Point count:' + str(points))`.

```
points = 12
announcement = 'Your point count:' + str(points)
print(announcement)
```

Questions:

1. What tells us whether a variable contains a number or a character string?
2. What will happen if we use the + operation on text and a number simultaneously?
3. What will happen if we use the * operation on text and a number simultaneously? (For example: `s = 'abc' * 4`)

If we want to find out whether a variable contains a number or a character string, we can use the function `type(variable_name)`. This function finds out the type of information stored in a variable, also known as the data type.

```
>>> a = 12
>>> type(a)
<class 'int'>
>>>
```

We can see that a number is of the data type int, which is an abbreviation of the word integer, as in a whole number.

```
>>> s = 'Python'
>>> type(s)
```

```
<class 'str'>
>>>
```

The variable **s** is of the data type **str**, which is an abbreviation of the word **string**.

Questions:

4. We set the variable `announcement` to be: `announcement = ' '`. What data type is `announcement`?
5. We executed these commands: `count = '12' result = count * 5`. What will be stored in the variable `result` and what data type will it be?
6. We executed these commands: `count = '12' result = count * 5`. What would change if we replaced `count * 5` with, e.g., `count * -1` or `count * -5`?

Up until now we have worked mostly in the graphical environment of the **tkinter** library. We have read input using the **entry** widget. However, many programs do not need a window with a graphical interface; text mode (shell window) is sufficient for them. We have already encountered such a program while generating random sentences. In text mode we can give the program input using the function **input**. The **input** function has only one parameter - containing text which will be displayed to the user, after which the program waits for input, which we signal the end of by pressing the enter button.

```
#p1-1-01
from random import *
name = input('What is your name?')
print('Hi ' + name + ', nice to meet you :)')
yearofbirth = input(name + ', what year were you born?')
name2 = choice(('Alicia', 'Barbara', 'Eve', 'Sophia'))
print('Ah , I remember, ' + name2 + ' was also born in ' + yearofbirth)
```

Questions:

7. What will the previous program do?
8. What data type are these variables: `name`, `yearofbirth` and `name2`?

Exercises:

1 Adjust the program `p1-1-01` so that the computer calculates our age (approximately, since we don't enter our exact date of birth).

2 Expand the program `p1-1-01` with further questions and answers.

3 Create a program that prompts the user to enter his/her name and age. The program will then greet the user, for example, `Hi Philip` and print how many more years the user needs to reach the age of 100.

4 Create a program where the user enters his/her name. The program then prints a "label" in this shape:

```
**********
*        *
* Andrea *
*        *
**********
```

1.2 Character indexing in a string, constructing a new string

Let's write the individual characters of a string in a column, each character on a separate line. To access the characters of a string, we can use a loop:

```
for character in 'Python':
    print(character)
```

or:

```
string = 'Python'
for character in string:
    print(character)
```

Questions:

9. What will the following program do? What will it display?

```
string = 'Python'
index = 0
for character in string:
    index += 1      #is the same as index = index + 1
    print(character)
print(index)
```

The function `len(string)` finds out the length of a string.

```
sentence = input('Write a sentence:')
length = len(sentence)
print('The number of characters in your sentence is:', length)
```

We can access individual characters of the string `string` using `string[character_number]`. Characters are numbered (indexed) incrementally from `0` to `len(s) - 1`. The number we use to access a character in a string is also called the `index`. For example:

```
string = 'Python'
print(string[0]) # prints 'P'
print(string[1]) # prints 'y'
```

We can also print out the whole string like this:

```
string = 'Python'
for i in range(len(string)):
    print(string[i])
    print(i)
```

In the shell we will see:

```
========== RESTART: write.py ==========
P
0
y
1
t
2
h
```

```
3
o
4
n
5
>>>
```

The first character of a string has the index `0` and the last has the index `len(s) - 1`. Often we need to access the last character. We can also index using negative numbers in order to access characters from the other side of the string (from the end). (see image)

P	y	t	h	o	n
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

Questions:

10. Why do we not get the last character from the string `r` by writing `r[len(r)]`, but instead by writing `r[len(r) - 1]`?
11. What happens if we use an index that is outside the length of the string? For example: `string = 'Python' string[10]` or `string[-10]`.
12. We can find and print out a specific character of a string. What will happen if we try to change it? For example, by writing `string[2] = 'i'`.

String (`string`) is an immutable type in Python, which means that we won't be able to change a character, e.g., `string[1] = '-'` (Python will then throw an error).

```
>>> string = 'Python'
>>> string[1] = '-'
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    string[1] = '-'
TypeError: 'str' object does not support item assignment
>>>
```

Instead of changing the string, we always have to construct a new string (which can still have the same name).

```
>>> name = 'Tom'
>>> name = name[0] + 'i' + name[2]
>>> name
'Tim'
>>>
```

In the following example, we create a new string in which we change all `'t'` characters to be `'*'` instead.

```
#p1-2-01
sentence1 = 'Creating with Python'
sentence2 = ''
for i in range(len(sentence1)):
    if sentence1[i] == 't':
        sentence2 = sentence2 + '*'
    else :
        sentence2 = sentence2 + sentence1[i]
print(sentence1)
print(sentence2)
```

Or we can also leave out a certain character (spaces, for example).

```
#p1-2-02
sentence1 = 'Creating with Python'
sentence2 = ''
for i in range(len(sentence1)):
    if sentence1[i] != ' ':
        sentence2 = sentence2 + sentence1[i]
print(sentence1)
print(sentence2)
```

Exercises:

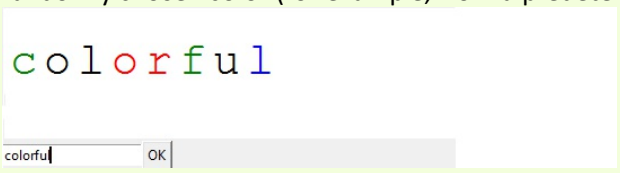
5 Create a program that receives two strings of the same length as input and joins them into one, alternating characters from the first and second string. For example, we enter `'Thomas'` and `'Andrew'` and the program creates: `'TAhnodmraesw'`.

6 Create a program which receives a string as input. This string was created by interweaving two different strings (see previous exercise). The program then prints out the original two strings. For example, we enter `'TAhnodmraesw'` and the program creates `'Thomas'` and `'Andrew'`.

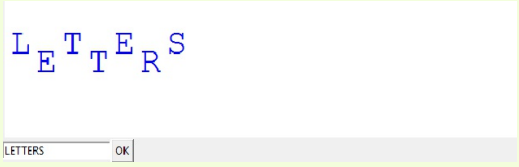
7 Create a program which receives two strings of different lengths as input and joins them into one new string, where characters from the first and second string alternate. Since one of them is shorter, at the end there will be only the remaining characters from the longer string. For example, we enter `'Diana'` and `'Ferdinand'` and the program creates: `'DFiearndainand'`.

8 Create a program which receives a sentence in English (ending with a dot, a question mark or an exclamation mark) as input. The program then proceeds to print out the type of sentence based on its ending character.

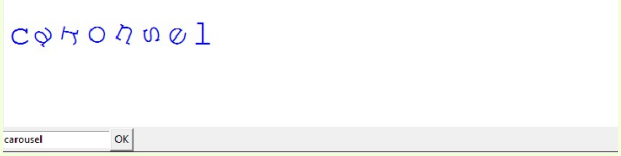
9 Create a program which receives a word as input. This word is then displayed on a graphical canvas in the form of an "advertisement", where each character of the entered word will be written in a randomly chosen color (for example, from a predetermined group of colors).



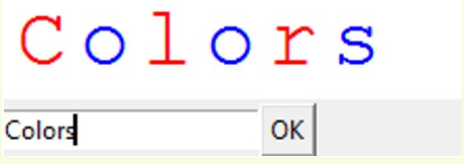
10 Create a program which receives a word as input. This word is then written onto a graphical interface so that:
a) letters in even positions will be written lower than those in odd positions,



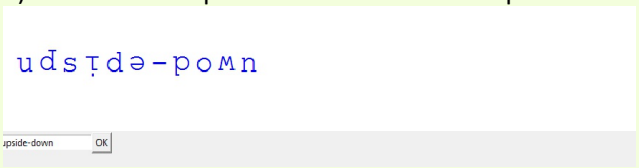
b) letters will be rotated so that the last letter is again rotated "normally",



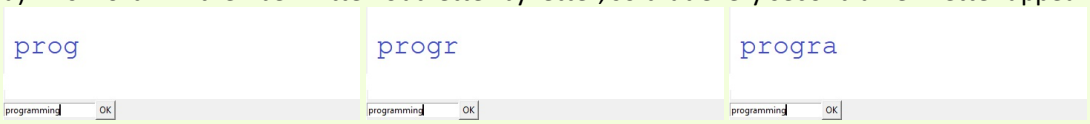
c) two colors will alternate in the written word,



d) letters in even positions will be rotated upside down.



11 Create a program which receives a word as input.
a) This word will then be written out letter by letter, so that every second a new letter appears.



b) Individual letters will be added to the word every second in a way that makes it look like they arrive from the right.
c) When the whole word is displayed, after a short while the text will disappear and the process will begin anew.

12 Create a program which receives a sentence as input. The program will write:
a) the number of words in the sentence,
b) the length and index of the longest word in the sentence – if there are more (with the same length), writing out one is enough,
c) the longest word in the sentence – if there are more, print out all of them.

13 Create a program which receives a word as input. The program will then print it out in this fashion:

```
H
 E
  L
   L
    O
```

Questions:

13. In one of the exercises, we joined (alternated characters from two strings) two strings of different lengths. Can we separate the resulting string into the two original strings? Reason your answer. How could we improve the method of interweaving characters so that we could also easily separate a string that was created from two strings of different lengths?

1.3 Substrings, slices

We can also create new strings using slices - indexing characters with multiple indices - `string [start : end]`. The character with the index `end` will no longer be in the result. We always have to extend it by one index (similar to how we set the boundaries in a for loop for `range`).

```
>>> s = 'PROGRAMMING IS AWESOME'
>>> s[3:7]
'GRAM'
>>>
```

P	R	O	G	R	A	M	M	I	N	G		I	S		A	W	E	S	O	M	E
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
-22	-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> s = 'PROGRAMMING IS AWESOME'
>>> s[15:19]
'AWES'
>>> s[15:21]
'AWESOM'
>>>
```

We can also use slices to change the format of a date. For example, if we want to rewrite the date 31-12-2017 to the format yyyy-mm-dd, which would look like 2017-12-31:

```
>>> date1 = '31-12-2017'
>>> date = date1[6:10] + '-' + date1[3:5] + '-' + date1[0:2]
>>> date
'2017-12-31'
>>>
```

Questions:

14. Which format is more useful for programming? Why?
15. We often use notations such as these. Try to find out what they mean.
    a) `s[:5] s[:2]`
    b) `s[2:] s[3:]`
    c) `s[:]`
16. Which of these notations will work and which will return a nonempty string?
    `s[-5:-2] s[5:2] s[-2:-5] s[-2:] s[:-2]`

We can also set the step of a slice (similar to the command **`range()`**, which we used in for loops). The step determines how much we move from the first index. If we don't specify the step, the default value is 1.

```
>>> s = 'Creating with Python'
>>> s[0:10:2]
'Cent'
>>> s[0:19:2]
'Cetn ihPto'
>>>  s[-1:-10:-1]
'nohtyP ht'
>>> s[::-1]
'nohtyP htiw gnitaerC'
>>> s[::-2]
'nhy twgiar'
>>>
```

Slices can also contain variables or expressions whose result is a number usable in a slice. We can also create slices like this:

```
s = 'Python'
for i in range(len(s)):
    print(s[:i+1])
```

```
P
Py
Pyt
Pyth
Pytho
Python
```

```
s = 'Python'
for i in range(len(s)):
    ns = '-' * i + s[:i+1]
```

```
    print(ns)
```

```
P
-Py
--Pyt
---Pyth
----Pytho
-----Python
```

Using slices, we can create a new string in which we can replace one or more characters. For example:

```
>>>s = 'Python'
>>>s = s[:2] + 'T' + s[3:]
>>>s
'PyThon'
>>>
```

Exercises:

14 We input a string which was created by alternating letters from two strings of the same length. For example, we enter `'TAhnodmraesw'`. Using slices (without using a loop), create the two original words from the input. For this example it would be: `'Thomas'` and `'Andrew'`.

15 Create a program which takes a string as input and creates the following output (this example is for the string 'Python'):

```
......P
.....Py
....Pyt
...Pyth
..Pytho
.Python
```

16 Create a program which takes a string as input and creates the following output (this example is for the string 'Python'):

```
......PP......
.....yPPy.....
....tyPPyt....
...htyPPyth...
..ohtyPPytho..
.nohtyPPython.
```

17 Create a program which takes a valid e-mail address as input. The program then writes:
a) the highest level domain (TLD) with the dot,
b) the mail server address,
c) the username,
d) a list of all domains in order from the first to lowest level.

```
Enter email: alexander.great@mail.pythonsoftware.net
TLD: .net
Server: mail.pythonsoftware.net
User: alexander.great
Domains:
1st level domain is: net
2nd level domain is: pythonsoftware
3rd level domain is: mail
```

18 There is a country where people receive a unique identification number when they are born, called a birth number. It contains six digits, a slash, and then four more digits. The first two digits are the last two digits of their year of birth - 80 means the year 1980. The next two digits represent their month of birth. Women have the number 50 added to their month number (e.g., a man has the number 10 for October; a woman has the number 60). The third pair of digits represent the day of birth (within that specific month). The four digits after the slash are an encoded representation of the place of birth. Create a program which takes a valid birth number from the 20th century as input. The program then writes:
a) the date of birth in the format day.month.year,
b) the person's gender.

Example:

```
Birth number: 806202/5675
Date of birth: 2.12.1980
Gender: Woman
```