PYTHON

HANDBOOK FOR BEGINNERS

A Hands-On Crash Course For Kids, Newbies and Everybody Else

12+

ROMAN GURBANOV

Roman Gurbanov

Python Handbook For Beginners

http://www.litres.ru/pages/biblio_book/?art=65147951
SelfPub; 2021
ISBN 978-5-532-96720-5

Аннотация

This book will provide you with basic knowledge and skills in Python programming, covering topics such as variables, numbers, strings, booleans, conditional statements, loops, lists, dictionaries, functions, classes and objects, modules, and packages.

Every chapter is wrapped up with a small test. Detailed explanations and practical examples accompany every topic to ensure you acquire an essential Python coding skill upon completing the book.

This book is excellent for everyone who wants to learn to code and is just starting. Other great books are available for those who have already mastered basic Python programming skills and looking to improve them.

Содержание

Roman Gurbanov
Python Handbook
For Beginners

INTRODUCTION

1 How To Work With This Book?

This book suggests 11 themes that, once mastered, will give you basic skills in Python programming. These themes are arranged according to their difficulty level, from simple to more complex. If you wish to skip chapters, you are welcome to do so but bear in mind that every chapter contains concepts given in preceding chapters.

To get the most of this book, I highly recommend creating your version of every code given in the book. Tweak the code and see how it affects the output of the program.

It is also highly recommended that you take every test given at the end of chapters. Should you stack anywhere, you can always refer to the test answers given at the end of the book. But don't go there too soon. Take your time and reattend past topics.

Finally, it's worth mentioning that you don't need to install any

software to write and run your code as you navigate through the book. You can use the free Python compiler (a program to write and run code), available here: https://online.qiber.org/code. No need to register or log in. I will be using this compiler in the book, and you may feel free to do the same. Should you decide to use other Python compilers, go ahead! There are many excellent, free compilers on the web. Here are some of them:

https://www.programiz.com/python-programming/online-compiler/

https://www.onlinegdb.com/online_python_interpreter

https://www.w3schools.com/python/trypython.asp?filename=demo_compiler

I am not affiliated with the above compilers and don't take any credits for them. I just want to give you some help diving into practical coding as soon as possible.

2 Why Python?

Python is one of the easiest to learn, yet, one of the most popular and widely used programming languages.

I would recommend Python as the first programing language to anyone who wants to learn to code. Why?

Python has a clean, minimalistic syntaxis. That almost looks like a natural language, which makes it easy to write and read the code.

It can take just a few lines of Python code to write a small script that does something. Simultaneously, it would require dozens of code lines in Java or C++, for instance, to complete the same piece of code.

Python is a high-level programing language, which means that it automates many essential tasks, such as memory management, which helps you focus on the core functionality of the program you create.

Python is extremely popular in the real world. Think of Google, Instagram, Netflix. They all use Python in areas that involve machine learning, data structuring, and processing.

Python is truly universal. Not only it works excellent for data-related scientific tasks. But you can build web apps and games too. Thanks to the respective Python frameworks.

Lastly, Python has a loyal and ever-growing community of supporters. Meaning the scale for Python application and the

level of quality and efficiency of software built with Python will only grow.

3 Brief History

In 1980, Guido van Rossum, a fellow at the Dutch CWI, set out to develop a powerful yet easy-to-read programming language.

Guido was developing Python in his spare time for one of the projects that required a scripting language.

While working on it, Guido borrowed some groundwork from another programming language that he was also developing. It was ABC language, created to teach programming.

Today Python is a popular, versatile, and mature programming language with many rich application libraries and extensions.

Professional programmers use it for various fascinating projects and support the language as a part of the global Python community.

Although professionals use Python, it is ideal for beginners. It helps schoolchildren, students, and simply novice developers take their first steps into the world of programming.

By the way, Guido named it so not in honor of the well-known snake species but the once-popular comedy TV show "Monty Python's Flying Circus." However, the language is still associated with the snake, reflecting snakes heads on its logo.

4 What Can You Create With Python?

Today, Python is used in various applications, including social media, artificial intelligence, and games like Civilization, Battlefield, World of Tanks, etc. Let's take a closer look at the examples.

Social Media

Billions of people use services like Facebook, Instagram, Reddit, Pinterest, and Quora. All of the mentioned services implement Python and its powerful features for data processing.

Search Engines

Major search engines like Google, Yahoo!, Yandex, Mail.ru use Python in their products for the same reason as mentioned above. Python is very efficient for data processing and management. It's also a dominating programming language for artificial intelligence algorithms that comprise a considerable part of search engine services.

Video Games

Popular video games like Civilization, Battlefield, and World of Tanks use this Python in their architecture. Its application is growing in game development due to the constant improvement and release of specific Python game dev frameworks and

libraries.

Streaming and Cloud Storage

Youtube, Netflix, Spotify, and Dropbox use Python. Worth mentioning that Guido van Rossum has been working in the Dropbox company from 2013 until his official retirement in 2019.

Space And Neural Network

NASA uses Python in their Workflow Automation Systems, while SpaceX uses Python for testing.

Tesla uses Python too. Its developers build Autopilot neural network initially in Python for rapid iteration; Python's syntaxis is clean and minimalistic. It allows building and testing functional prototypes fast.

CHAPTER ONE:
LET'S GET CODING!

1 Your First Line Of Code

Any, even the most advanced python program, starts with the first line of code. Here is an example of a simple program that has just one line of code. The only thing the program does is it outputs the message: "Hey! This is my first line of code!"

Open the console and write the following:

print("Hey! This is my first line of code!")

Don't worry. We'll learn what this code means in the following chapters. Now it's essential to get you going! When done, run the code by hitting the green play button. Here is the result you should get:

```
Hey! This is my first line of code!
```

Well done! You've just started and have already written your first simple program.

2 What Is a Program?

Even if it was just a single line of code, we've created a program. Like those that make computers work.

But what is a program? A program is a set of instructions and rules for a computer written in a programming language. If that makes sense, let's go on and reinforce what we have learned. Here is the code I've shuffled. It will help if you put it so that it would make it work.

"Once upon a time…!" () print

If you put it right, you should get a program that prints a sentence. Here it is below.

```
Once upon a time...
```

Are you done with the task? Great! In both programs that we have just created, we used a print function. We will use it a lot in this book. But first, let's dive a little deeper into it.

3 Print Function

Print is the function that does what it's called: it "prints" text on a screen. Programmers use this function to show messages to users. Such as "Your session is expired, please log in" or "Your password is too weak, please use a stronger password," etc.

Not only simple text messages, but the print function can also display different calculation results, presented in numeric format. We will learn all this stuff in the coming chapters. For now, we will focus purely on the print function and code some more with it.

Here is the plain text for you. Please use it with the print function, so when you run the code, it shows you the following message: Hey! I keep coding! Here is what you should get:

Have some fun. Since now you have learned how to write a program that displays messages, experiment with different messages you want your program to print. Use numbers too. Try adding or removing anything from your code. And then watch what happens; This is the best way to learn how the code

operates.

4 How Does Python Read The Code?

As soon as you run the code, the computer starts reading it line by line, from top to button. Just like you're reading this book or anything. It may not sound too important, but it's essential to consider it when building and arranging our code; This is why some elements like modules (we will learn them in the following chapters) belong at the top of the code. We import them first to use them further down.

5 Counting Program

Let's build a program that counts from 1 to 3. Here is the code we need to do that:

```
print("1")
print("2")
print("3")
```

Pretty simple, right? Now extend the code so that it could count to 10. Here is the result you should get when you run your extended program:

```
1
2
3
4
5
6
7
8
9
10
```

6 Python Challenge

Let's celebrate the completion of the first chapter with a small challenge. You need to complete the below line of code so that it could output the message: "I nailed the first chapter!"

Here is what the output should look like when you fix and run the program:

```
I nailed the first chapter!
```

7 Wrapping Up Chapter One

In the first chapter, we have accomplished the following:
1) Learned what Python is;
2) Understood what a program is;
3) Found out how does Python read the code;
4) Wrote our first line of code;
5) Created a few simple programs;
6) Finally, we learned and applied the print function.

8 Chapter One Test

1. A computer program is a:

1) Set of instructions and rules for a computer, written in a programming language.

2) Piece of code written on a computer.

3) Downloadable game.

2. How do we let a computer know that it needs to display a message on the screen?

1) With a magic word.

2) Using the print() function)

3) Using the command "Display message!"

3. In what order does a computer process (read) the code?

1) When we run the code, the computer reads that code line by line. Top-down.

2) When we run the code, the computer reads that code from bottom to top.

3) When we run the code, the computer doesn't read anything; it remembers everything by heart.

4. Arrange the parts of the code so that the program displays the message "I love Python!"

)

(

"I love Python!"

print

CHAPTER TWO: VARIABLES

1 What is a Variable?

A variable is a simple data structure that has a name and a value. Now, let's figure out what is what.

Imagine a folder on your desktop. In this case, the folder is a variable. The folder has the label "film."; This is the name of the variable. The folder contains a video called "The."; This is the value of the variable.

Here is how this variable looks in Python language:

```
film = "Super Cat"
```

Let's break it down.

1) First, we give a name to our variable – film

2) Second, we put equal sign =

3) Third, we give a value to our variable "Super Cat."

A value in a variable always goes with quotes, like in our example. Otherwise, a variable will not work.

Now, as we know what a variable is and what it contains, let's

make one!

2 How to Create a Variable?

Write our variable from the example above to the console:

```
film = "Super Cat"
```

And run the code. Nothing? That's right. There is no message to show on the screen as we did not use the print function. We will combine variables with print function in the following chapters. But, for now, hit the >_REPL button.

You should now see the input field that looks like this:

Input the name of our variable, which is the film, and hit enter.

In response to the variable name, the program returns its value, which in our case is "Super Cat."

Now, as you've created your first variable, experiment with it. Change name and value. See what happens.

3 Let's Print Variable Value

We've just learned how to create a variable. Now let's learn how to output its value.

Do you remember the print function we used earlier? Let's combine it with our variable and see what happens. Write this code in the console and hit run:

```
film = "Super Cat"
print(film)
```

Let's break it down:

1) First, we declared a variable and gave it a name – film.

2) Then we assigned a value to the variable – Super Cat.

3) Then, on a second line, we wrote a print function.

And finally, we passed the variable name to the print function, putting it in the function brackets.

Every time we create a variable and pass its name to the print function, it will output the variable's value, just like in our example.

Now you know how programmers display variables in Python. It doesn't seem too tricky. Let's have some practice. Below is the code that is missing some elements. You have to fix it so the program could create a variable and display its value.

= " "
()

By now, you should have sufficient knowledge and skills to accomplish this task. When done, iterate on it as long as you like. Change variable names and values. Hone on your new skills!

4 Wrapping Up Chapter Two

In chapter two, we have accomplished the following:
1) Learned what a variable is;
2) Created our first variable;
3) Learned how to display variable values.

5 Chapter Two Test

1. What is the purpose of variables?

1) Computers use variables to store information.

2) Computers use variables to change information.

3) Computers use variables to retrieve or delete information.

2. If a variable name consists of two or more words, you shall connect them using:

1) Underscore.

2) Dash line.

3) Write the words together, without space.

4) Write the words together with a capital letter without space.

3. We can display a variable on the screen by:

1) Using the print() function and passing the print command in its parentheses.

2) Using the print() function and passing the value of the variable in its parentheses.

3) Using the print() function and passing the name of the variable in its parentheses.

4. Arrange the parts of the code so that the program creates and displays the variable.

(name)
name
=

print
"John Doe"

CHAPTER THREE: NUMBERS

1 Integers and Fractional Numbers

You already know that integers, like 5 or 10, and fractional numbers, like 5.5 or 10.7, also called float from your school math classes. Python works with both types of numbers. It also works with relevant math operators, which are coming further.

2 Math operators

Python allows you to work with numbers using relevant math operators, that are:

1) Addition operator: +
2) Subtraction operator: -
3) Multiplication operator: *
4) Division operator: /

Let's jump into the console to work with some numbers and math operators.

3 Working with Numbers

Run empty console and hit repl button, which you already know. You should get an empty input field: input 2+2, and hit enter. What's the number you've got? Now, using the same field, subtract 5 out of 10, using subtraction operator: – Let's also multiply 5 by 5, using the multiplication operator: * Finally, let's divide 10 by 2, using the division operator: /

4 Division Without Remainder

Let's talk about division without remainder in Python. Did you notice that we've got a fraction when we divided 10 by 2, which was 5.0? But what if we need to get an integer? To get that, all we need is to use a double division operator – // Try it out. Open the input field, write 10//2, and hit enter. What number have you got now?

5 Calculation Order

Python handles calculation order, which has no difference from the order you've learned in math classes. Can you calculate (5+5)*3, then write it into the input field and check if you were right? Here is how Python will calculate it:

First, Python will calculate whatever is in brackets, following the order: first multiplication, then division, then addition, then subtraction. After that, Python will calculate whatever is around brackets, following the same order as above (multiplication, division, addition, subtraction)

Hence, Python will add 5 and 5, which will result in 10. And multiply 10 by 3, which will result in 30. Was your answer – 30? Or was it something else?)

6 Numbers and Variables

By this time, we've learned print function, variables, and numbers in Python. Let's combine them all!

Let's write an example with numbers, save its result in a variable, and then display it. Here is how it looks like:

```
result = 2+2
print(result)
```

Guess what the output will be? Please take the above code and run it in the console to check it out.

Let's break it down:

1) First, we declared a variable, giving it the name "result" and the value "1 + 1";

2) Then we went down one line, wrote the print function, and passed the name of our variable in its brackets;

3) When we ran the code, Python calculated the value of the variable by adding 2 + 2;

4) The calculation returned 4, and Python assigned this value to the variable "result";

5) Then Python saw that we want to display the result's value because of the print (result) and print it accordingly.

As you can see, numbers and variables work well together in Python.

Let's practice some more and create your examples with numbers and variables, similar to what we just saw. Here are some boilerplate examples for you:

```
_____
result = 2+2
print(result)

_____
result = 10-5
print(result)

_____
result = 5*5
print(result)

_____
result = 10/2
print(result)

_____
```

7 Wrapping Up Chapter Three

In chapter three, we have accomplished the following:

1) Refreshed on what are integers and fractional numbers;

2) Learned basic math operators used in Python;

3) Learned how to divide a number without remainder in Python;

4) Got to know the order of calculation in Python;

5) Combined numbers and variables in math examples.

8 Chapter Three Test

1. What types of numbers are there in Python?

1) There are three types of numbers in Python: Integers, almost whole numbers, and fractional.

2) There are two types of numbers in Python: Integers and half integers.

3) There are two types of numbers in Python: Integers – integer and fractional numbers.

2. How does Python divide 10 by 5? (Multiple selection)

1) 10:5
2) 10/5
3) 10-5
4) 10 * 5
5) 10 // 5

3. Arrange the parts of the code so that you get the integer 4.

five
10*2
//

4. This code should display 5. But what is missing?

result = 10/2

_____(result)

CHAPTER FOUR: STRINGS

1 Strings in Python

Do you remember the messages we displayed in chapter one, like "Hey! This is my first line of code!" or "Once upon a time.." They are all strings. In other words, a string is a data type that has a text format, or just a text, enclosed in quotes. Hence, to create a string, we have to type a text and enclose it in quotes.

2 Strings And Print Function

Let's write a code that would display the string – "Now I know what a string is in Python." You already know how to display strings in Python, right? Here is the code.

```
print("Now I know what a string is in Python.")
```

Write the above line into the console and hit run. Now it's your turn. Think of some different string, rewrite and run the above code.

3 Saving Strings in Variables

Do you remember our first variable: film = "Super Cat"? Now you know that the value of the variable – "Super Cat" had a string format, as we enclose it in quotes. Let's take our knowledge further, create a message in a string format, save it in a variable, and display the variable value on your screen using the print function. Let me do it first:

```
message = "Hello, Super Cat!"
print(message)
```

Run this code in the console. What do you see? Now it's your turn. Following the above example, think of any message. Right it down as a string. Save it in a variable and display the variable value using the print function. What you've got?

4 Strings Concatenation

We can combine strings. There is a method for that, called – concatenation. All we have to do to concatenate strings is to put addition operator + between them. Easy right? If so, let's get concatenating!

"Super Cat" + "saves the day!"

Now, let's display the two concatenated strings:

```
print("Super Cat" + "saves the day!")
```

Write this code into the console and hit run. Have you noticed something weird? Hmm. It looks like our lines stuck together. No worries. We can quickly fix that. There are several ways to do it but let's stick with the most simple – leave a space between the quote and string:

```
print("Super Cat" +" saves the day!")
```

Have you fixed that? Hit run again.

5 Strings Concatenation And Variables

Note that we can concatenate a string only with another string. Or a string-formatted value. For example, if we created a variable and assigned a string-formatted value to it, we can also concatenate such a variable with another string. In the example below, I created a variable (hero) and assigned a string-formatted value (Super Cat) to it. Then I displayed this value in concatenation with another string (VS Duper Dog).

```
hero = "Super Cat"
print(hero + " VS Duper Dog")
```

Try this code out. What message do you get? Now, change the code as you like. You can even concatenate more than two strings! Alter and run the code. See how the output changes!

6 String Formatting

Now let's talk about what string formatting is. We have already learned how to concatenate strings using the + operator. The + operator can only concatenate a string with another string. But what if we want to concatenate a string with something that doesn't have a string format? In this case, we use string formatting to turn a value that doesn't have a string format into a string-formatted value. To do so, we need two things:

The format() method to format the non-string value and insert it inside the string's placeholder.

The placeholder itself – { } for the non-string value.

Let me show you how it works in the example below:

```
print("My name is Super Cat, and I'm {} years old".format(2))
```

Please write this code into the console and run it. Here is the result you should get:

```
My name is Super Cat, and I'm 2 years old
```

Now let's break it down.

1) We created a placeholder in our string. You can identify it

by the curly brackets – {}. This placeholder is for the age of the Super Cat, which has a numeric value.

2) After the string end, we put the format() method and passed our numeric value, 2, to its brackets.

3) As a result of this operation, Python took our numeric value, formatted it into a string value, and passed it to the placeholder. So now it belongs to the entire string in the same string format.

4) After that, we displayed the entire string using the print function.

I encourage you to play with the given code and experiment with placeholders and values you pass to the format() method. Here is another way to accomplish the above exercise:

```
print("My name is {}, and I'm {} years old". format("Super Cat",2)
```

Can you explain how it works now? Can you come out with your ideas?

7 Wrapping Up Chapter Four

In chapter four, we have accomplished the following:
1) Learned what are strings in Python;
2) Learned how to store a string in a variable;
3) Learned how to concatenate strings;
4) Learned how to concatenate strings with variables values;
5) Leaned string formatting.

8 Chapter Four Test

1. What is a string?

1) A string is a line that goes through the code.
2) A string is text data enclosed in quotes.
3) A string is an integer formatted value.
4) A string is a float formatted value.

2. How do we create a string?

1) We write a text and enclose it in curly braces.
2) We write a text and enclose it in parentheses.
3) We write a text and enclose it and enclose it in quotes.

3. Arrange the code so that you get a variable with a value as a string. And then display the value of the variable on the screen.

(message)
message
"Hello, John Doe!"
print
=

CHAPTER FIVE: BOOLEANS

1 Comparison Operators

When we compare numbers to each other, we usually use the symbols like > (greater), < (less), = (equal), and so on. They work in Python too. And here's how they look:

Greater >
Less <
Equal ==
Not equal !=
Greater than or equal > =
Less than or equal <=

As you noticed, instead of the equal sign =, we need to use the double equal sign ==.

We need the double equal sign so that Python doesn't think we are trying to create a variable.

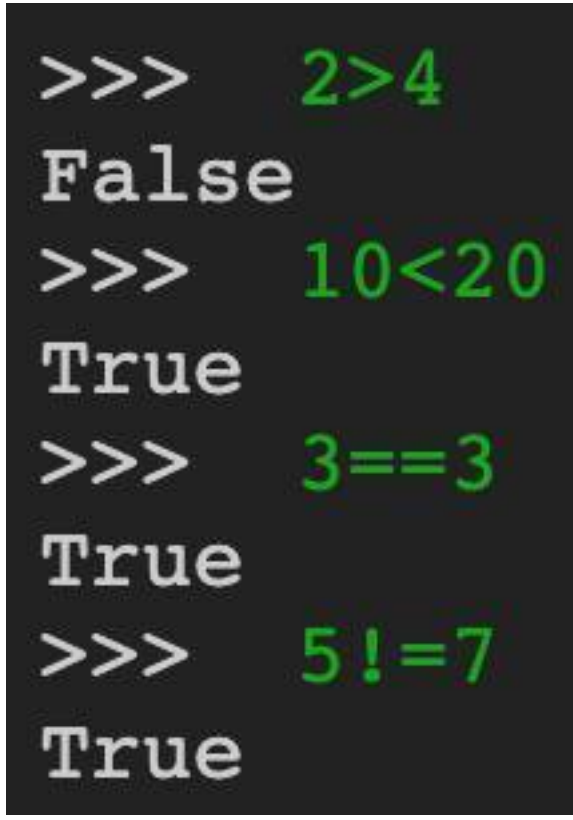
These symbols are called comparison operators in Python. And we need them to let our program compare different data, for instance: numbers, strings, variables, etc.

2 True or False

Boolean logic has Boolean values: True and False: True – when the condition is true. False – when the condition is false. Let's play! I will compare numbers, and you will answer in your head – True if it is true, or False if it is false. Let's go!

```
2 > 4
10 < 20
3 == 3
5 != 7
```

Now, let's check yourself. Run the console and hit the REPL button. Then, input the above comparisons into the input field, one by one, and hit enter. If your comparison is correct, the program will respond: True. If not, then False. Let's see how the examples above look in the console:



See that? This is how Boolean logic and values work. But what if we try comparing strings? Try comparing apples to oranges using the following:

"apples" == "oranges"

See the output? Now, try changing comparison operators and

values to your liking. See what happens? Now you know that Boolean logic works not only with numeric but with other data formats in programming.

3 True and False in Variables

Let us see how we can use True and False values in variables. We can store results from comparisons that return True or False in variables. This is how it works:

```
result = "apples" == "oranges"
print(result)
```

Run the above code in the console and see what it returns. Let's break it down.

I've created a variable named result in this code and assigned it a Boolean value from comparing the two strings "apples" and "oranges ".

Then I went down one line and printed the value of the "result" variable on the screen, passing the variable name in parentheses to the print function.

Does it make sense?) Now, alter the code by indicating that apples and oranges are not equal. What does the program return?

4 Comparing Variables

Not only can we compare numbers and strings, but the entire variables too! See how we can do it:

```
fruit = "Kiwi"
result = fruit == "Apple"
print(result)
```

Before you write the code into the console and hit enter, think of what output it will return? Let's break it down:

Конец ознакомительного фрагмента.