LEARN SMARTER

Python

Interactive Guide

1st Edition

By: Nesta Saint. Clever Parchment

Python

Interactive Study Guide - 1st Edition

By **Nesta Saint. Clever Parchment**

Copyright © 2024 by Nesta Saint. Clever Parchment
All rights reserved

First Edition: 2024
ISBN: ###-#-######-##-#

Published by Nesta Saint. Clever Parchment
New York, USA

Nesta Saint. Clever Parchment
GitHub.com/saintclever

Dedication

This book is tribute to my parents (Adina & Claude Parchment). Thank you both for everything.

Table of Contents

- Enhancements and Challenges
- Appendix

Additional Resources for Learning Python
Common Python Libraries and Frameworks
Glossary of Python Terminologies
Solutions to Chapter Exercises
Each chapter will include practical examples, exercises, and challenges to reinforce learning. The appendix will serve as a reference guide and provide further resources for readers to continue their Python journey.

Introduction

Welcome to **"Learn Smarter - Python 1st edition"** a technical beginner's guide to learning one of the most popular programming languages in the world. In this book, you'll embark on a journey to understand the fundamental concepts of Python programming and how to apply them in real-world scenarios. Within "Learn Smarter" every chapter covered you'll encounter a challenge to catapult your programming skills to the next level.

Preface

This book is designed for absolute beginners with no prior programming experience. Whether you're a student, a professional looking to switch careers, or simply curious about coding, this book will provide you with a solid foundation in Python programming.

In writing this book, my goal is to provide entry and junior level coders a with a practical and technical guide for developing their coding skills.

Acknowledgments

I would like to express my sincere gratitude to my parents for their support and encouragement throughout the writing of this book.

Chapter 1 - Introduction to Python

Welcome to Python programming! Python is a versatile, easy-to-learn language used in web development, data analysis, AI, and more. It's known for its readability and simplicity.

- **Who created Python?**
  Python was created by **Guido van Rossum, a Dutch** programmer, and it was first released in 1991.

- **Why learn Python?**
  Python is versatile, readable, boasts a large community, and is in high demand in the job market.

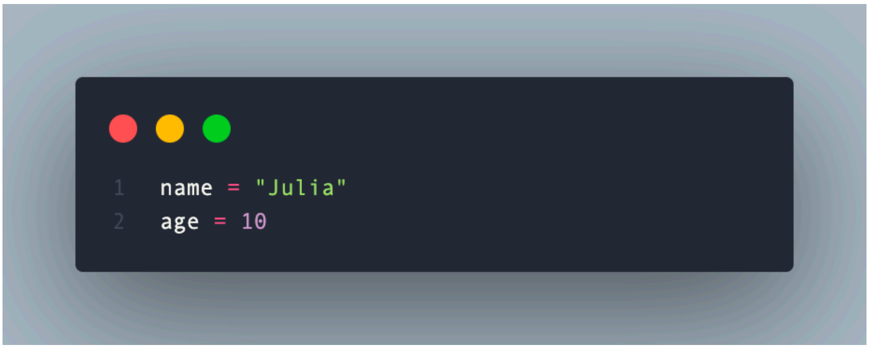
- **How to set up Python?**
  We'll install python from **python.org**, however for the first few chapters we'll use an embedded code editor.

Chapter 1 - Introduction to Python Challenge

Chapter 2 - Variables and Data Types

We'll learn about variables and basic data types. Imagine variables like boxes where you can put stuff, and data types like the different kinds of information you can put in those boxes.

- **What are Variables and Data Types?**
  Variables are like labeled containers. Within these containers we place our information to be used for later use. The information placed in these containers can be numbers, words, and lists of things, just like how you might have separate boxes for your shoes, clothes, and books. Each container holds a specific type of information, making it easier for the computer to understand what we're working with.



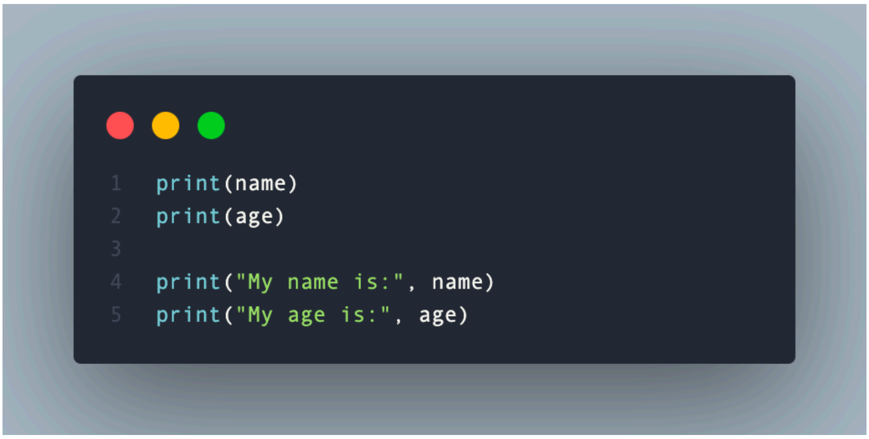
```
name = "Julia"
age = 10
```

variable = information

The image above displays two variables. A variable called name with the information of someone named Julia, and a variable called age with someones age as 10.

- **What are strings and integers?**
  Did you Notice "Julia" is in quotes and our number / integer is not? That's because strings are represented as quotes. Think of strings like colored beads placed on a string necklace. Each bead represents a letter or a symbol, while integers are plain number blocks used for counting and math.

- **What is print: print( )**
  Suppose we want to see what's stored in our variables. We can use a special Python key word called **print**.



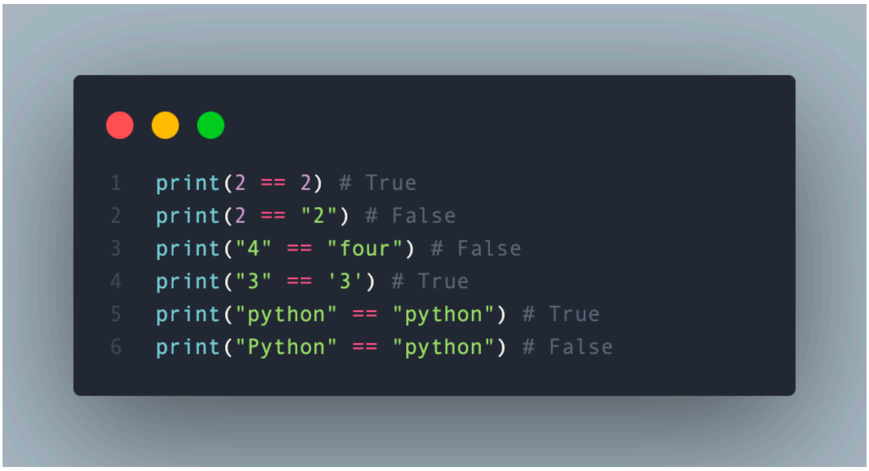
```python
print(name)
print(age)

print("My name is:", name)
print("My age is:", age)
```

Chapter 2 - Variables and Data Types Challenge

Chapter 3 - Comparison Operators

Suppose you have two numbers or pieces of information, like apples and oranges. Comparison operators in Python are like tools you can use to compare these numbers or pieces of information to see how they relate to each other. Here are some of the main comparison operators:

- **Equal to:** `==`
  Checks if two things are exactly the same.



```python
print(2 == 2) # True
print(2 == "2") # False
print("4" == "four") # False
print("3" == '3') # True
print("python" == "python") # True
print("Python" == "python") # False
```

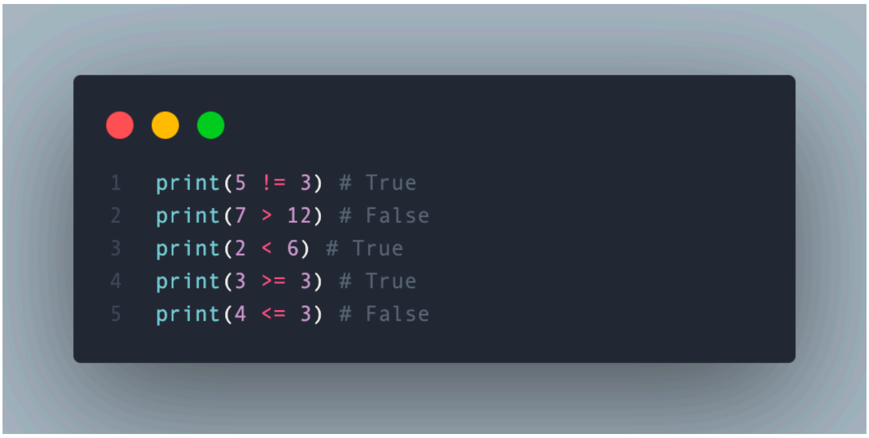
- **Not equal to:** `!=`
  Checks if two things are not the same.

- **Greater than:** `>`
  Checks if one thing is bigger than another.

- **Less than:** <
  Checks if one thing is smaller than another.

- **Greater than or equal to:** >=
  Checks if one thing is either bigger than or equal to another.

- **Less than or equal to:** <=
  Checks if one thing is either smaller than or equal to another.



```python
print(5 != 3) # True
print(7 > 12) # False
print(2 < 6) # True
print(3 >= 3) # True
print(4 <= 3) # False
```

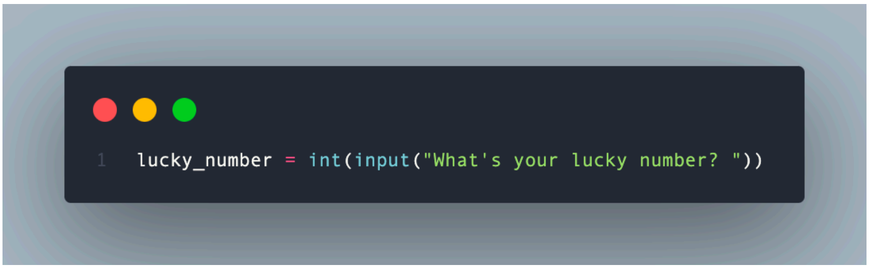
- **Bonus:**
  In addition to comparison operators, Python features logical operators: **and**, **or**, and **not**. Alongside these logical operators, Python also has **True** and **False** values.

Chapter 3 - Comparison Operators Challenge

Chapter 4 - User Input and Output

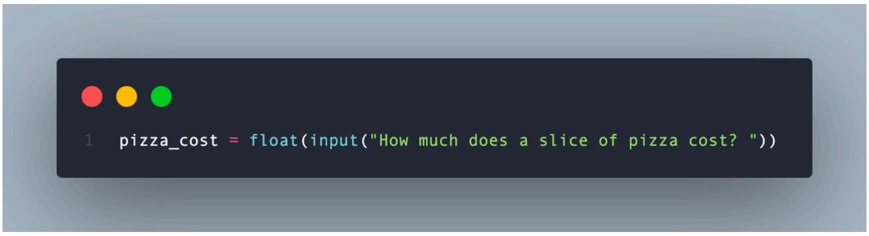
Imagine you're playing a game on your computer, and the game wants to ask you something, like your name or age. Python's **input( )** is like a way for the game to talk to you and ask for information. So, when the game asks, "What's your name?" and you type "Jane" and hit Enter, Python's input helps the game understand what you typed. It's like having a conversation with the computer!

- **What if the users input is a integer: int( )**
  Let's say the question is "What's your lucky number?" and you answer "15". Even-though it appears to be a integer the input method converts everything to a string. To transform a stringed integer to a whole number use the int method as followed.



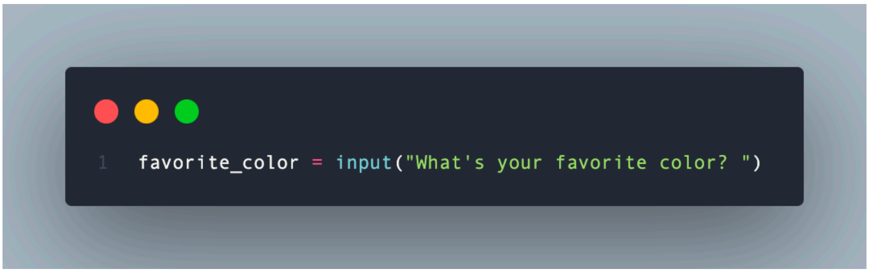
```python
lucky_number = int(input("What's your lucky number? "))
```

- **What if the users input is a float: float( )**
  Now, imagine the question is "How much does a slice of pizza cost?" and you answer "2.75", that's also a string. We would have to use the float method to convert our answer if we want to probably use it in a math equation.



```python
pizza_cost = float(input("How much does a slice of pizza cost? "))
```

- **What if the users input is a string: str( )**
  Finally, if the question is "What's your favorite color?" and you answer "blue", no converting is needed. However if you feel the need to convert a int or a float into a string



```python
favorite_color = input("What's your favorite color? ")
```

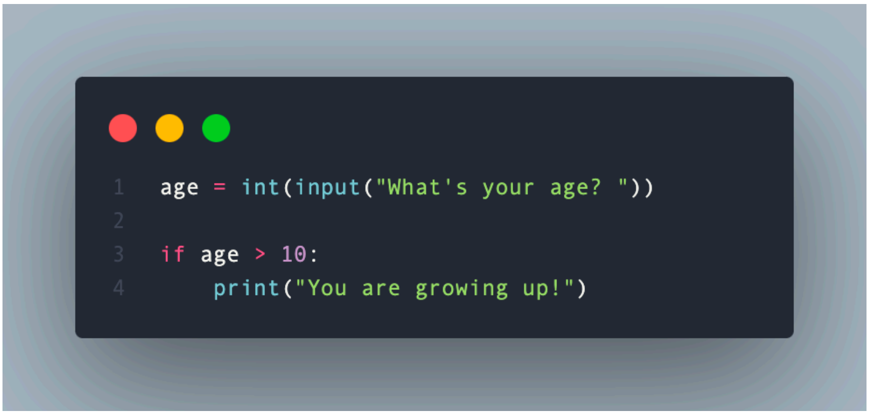
- **Bonus:**
  When creating a variable with more then one word use snake_case, an underscore after every word. When we have an input as a stringed int / float we can convert the input to a int or float.

Chapter 4 - User Input and Output Challenge

Chapter 5 - Conditional Statements

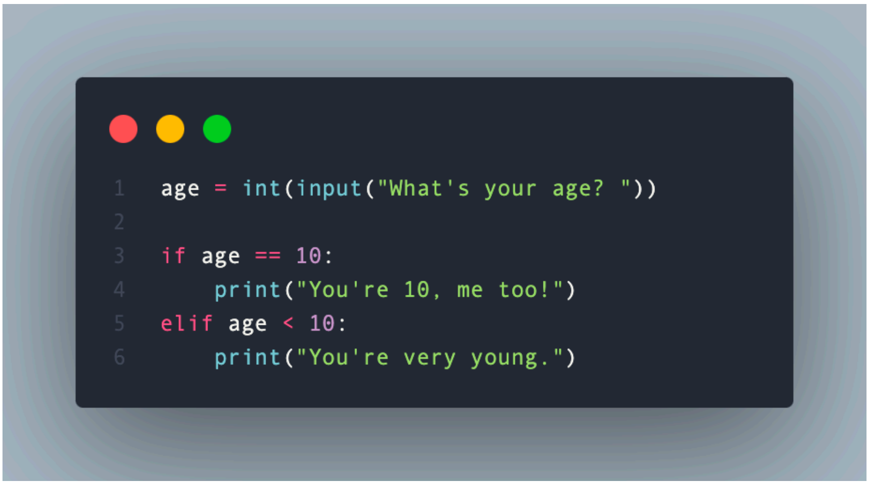
Just like choosing between paths in a game, we sometimes need to make decisions in our code based on certain conditions.

- **The if Statement:**
  It's like asking a question. If a condition is true, it does something.

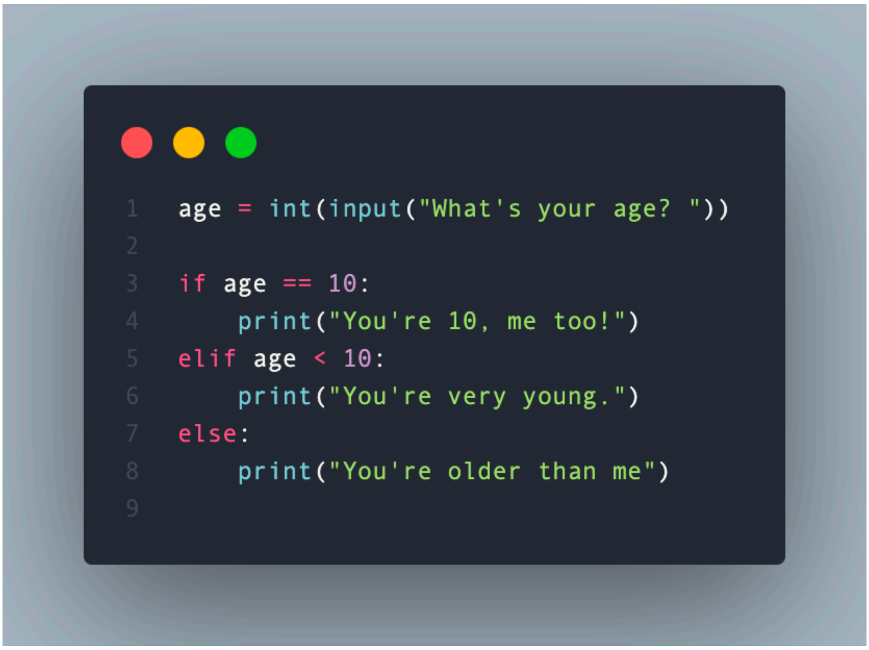


```python
age = int(input("What's your age? "))

if age > 10:
    print("You are growing up!")
```

- **The elif Statement:**
  When we have more than one condition, we use "elif" (short for "else if”).


```python
age = int(input("What's your age? "))

if age == 10:
    print("You're 10, me too!")
elif age < 10:
    print("You're very young.")
```

- **The else Statement:**
  The final result when your if and or elif don’t pass as true.


```python
age = int(input("What's your age? "))

if age == 10:
    print("You're 10, me too!")
elif age < 10:
    print("You're very young.")
else:
    print("You're older than me")

```

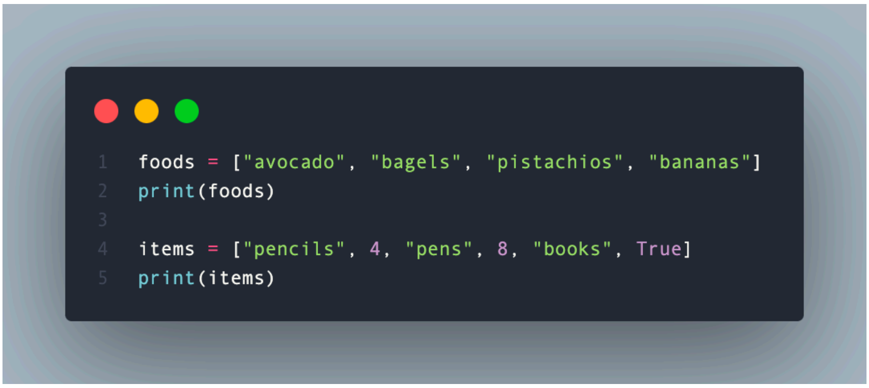
- **Bonus:**
  You can have multiple single if or elif statements however you can only have one else in a single block of code.

Chapter 5 - Conditional Statement Challenge

Chapter 6 - Lists

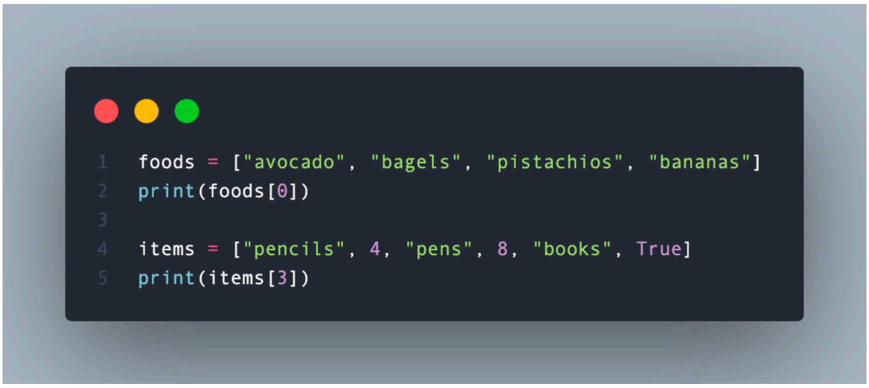
We learned about variables and how they can contain a string, integer, and a boolean but we also have a list.

- **List:**
  Although a variable can technically hold only one item at a time, a list is capable of containing a plethora of items with diverse data types. The distinction lies in its presentation: lists are denoted by square brackets enclosing multiple items separated by commas.



```python
foods = ["avocado", "bagels", "pistachios", "bananas"]
print(foods)

items = ["pencils", 4, "pens", 8, "books", True]
print(items)
```

- **Indexing:**
  Although a variable can technically hold only one item at a time, a list is capable of containing a plethora of items with diverse data types. The distinction lies in its presentation: lists are denoted by square brackets enclosing multiple items separated by commas.



```python
foods = ["avocado", "bagels", "pistachios", "bananas"]
print(foods[0])

items = ["pencils", 4, "pens", 8, "books", True]
print(items[3])
```

Chapter 7 - Loops

Think of a loop as an assistant that handles a repetitive task for you.

- **for loop:**
  Imagine we're at a grocery store with a shopping cart full of items. When we go to checkout, each item gets scanned and added to our bill. This is similar to how a for loop works. In a for loop, we go through a group of items (like numbers or words) one by one. It's like scanning each item at the checkout. Just as we see the item number or name on our bill, during the loop, we can see each item's value or name as we go through them one by one.

About the Author

Nesta Saint. Clever Parchment – A learner / explorer of many programmer