合肥大學
HEFEI UNIVERSITY

Programming with Python

Introduction

Thomas Weise (汤卫思)
tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn/5

Institute of Applied Optimization (IAO)
School of Artificial Intelligence and Big Data
Hefei University
Hefei, Anhui, China

应用优化研究所
人工智能与大数据学院
合肥大学
中国安徽省合肥市

Version: 2025-02-24

Programming with Python

This is a course on programming with the Python language at Hefei University (合肥大学).

The website with the teaching material of this course is https://thomasweise.github.io/programmingWithPython (see also the QR-code at the bottom right). There, you can find the course book and these slides. The repository with the example Python programs can be found at https://github.com/thomasWeise/programmingWithPythonCode.

Outline

Introduction

Introduction

- This course aims to teach you programming using the programming language Python.

Introduction

- This course aims to teach you programming using the programming language Python.
- What does *programming* mean?

Introduction

- This course aims to teach you programming using the programming language Python.
- What does *programming* mean?
- Programming means that we delegate a task to the computer.

Introduction

- This course aims to teach you programming using the programming language Python.
- What does *programming* mean?
- Programming means that we delegate a task to the computer.
- We have this job to do, this thing.

Introduction

- This course aims to teach you programming using the programming language Python.
- What does *programming* mean?
- Programming means that we delegate a task to the computer.
- We have this job to do, this thing.
- Maybe it is too complicated and time consuming to do.

Introduction

- This course aims to teach you programming using the programming language Python.
- What does *programming* mean?
- Programming means that we delegate a task to the computer.
- We have this job to do, this thing.
- Maybe it is too complicated and time consuming to do.
- Maybe it is something that we have to do very often.

Introduction

- This course aims to teach you programming using the programming language Python.
- What does *programming* mean?
- Programming means that we delegate a task to the computer.
- We have this job to do, this thing.
- Maybe it is too complicated and time consuming to do.
- Maybe it is something that we have to do very often.
- Maybe it is something that we cannot, physically, do.

Introduction

- This course aims to teach you programming using the programming language Python.
- What does *programming* mean?
- Programming means that we delegate a task to the computer.
- We have this job to do, this thing.
- Maybe it is too complicated and time consuming to do.
- Maybe it is something that we have to do very often.
- Maybe it is something that we cannot, physically, do.
- Maybe we are just lazy.

Introduction

- This course aims to teach you programming using the programming language Python.
- What does *programming* mean?
- Programming means that we delegate a task to the computer.
- We have this job to do, this thing.
- Maybe it is too complicated and time consuming to do.
- Maybe it is something that we have to do very often.
- Maybe it is something that we cannot, physically, do.
- Maybe we are just lazy.
- So we want that the computer does it for us.

Introduction

- Whenever we delegate a task to another person, we need to explain it.

Introduction

- Whenever we delegate a task to another person, we need to explain it.
- If you are a chef in a kitchen, you have to tell the junior trainee chef: "First you wash the potatoes, then peel the potato skin, then you wash the potatoes again, and then you cook them."

Introduction

- Whenever we delegate a task to another person, we need to explain it.
- If you are a chef in a kitchen, you have to tell the junior trainee chef: "First you wash the potatoes, then peel the potato skin, then you wash the potatoes again, and then you cook them."
- If you are visiting the hairdresser to get your hair done, you would say something like: "Wash my hair, then cut it down to 1cm on the top, trim the sides, then color it green."

Introduction

- Whenever we delegate a task to another person, we need to explain it.
- If you are a chef in a kitchen, you have to tell the junior trainee chef: "First you wash the potatoes, then peel the potato skin, then you wash the potatoes again, and then you cook them."
- If you are visiting the hairdresser to get your hair done, you would say something like: "Wash my hair, then cut it down to 1cm on the top, trim the sides, then color it green."
- You provide the other person with a clear and unambiguous sequence of instructions in a language they can understand.

Introduction

- Whenever we delegate a task to another person, we need to explain it.
- If you are a chef in a kitchen, you have to tell the junior trainee chef: "First you wash the potatoes, then peel the potato skin, then you wash the potatoes again, and then you cook them."
- If you are visiting the hairdresser to get your hair done, you would say something like: "Wash my hair, then cut it down to 1cm on the top, trim the sides, then color it green."
- You provide the other person with a clear and unambiguous sequence of instructions in a language they can understand.
- In this book, you will learn to do the same — with computers.

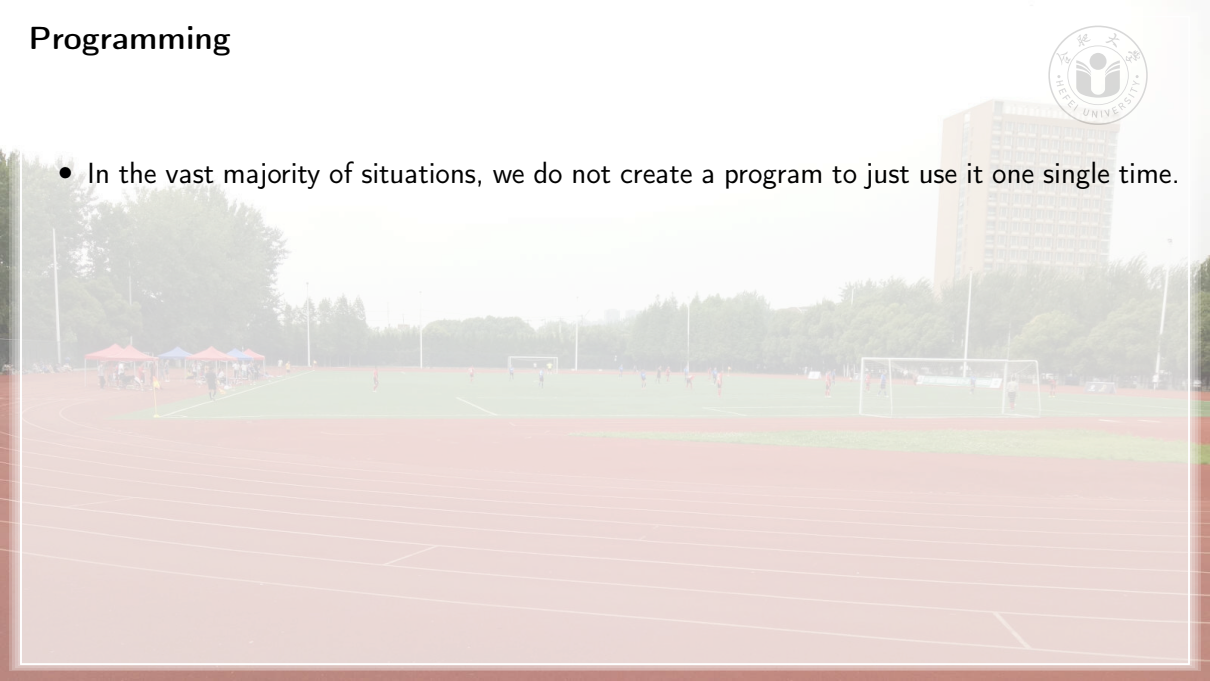
Programming vs. Software Development

Programming

Definition (Computer Program) A *computer program* is an unambiguous sequence of computational instructions for a computer to achieve a specific goal.

Definition (Programming) *Programming* is the activity or job of writing computer programs[19].

Programming



- In the vast majority of situations, we do not create a program to just use it one single time.

Programming

- In the vast majority of situations, we do not create a program to just use it one single time.
- This is similar to the real life situation of work delegation again.

Programming

- In the vast majority of situations, we do not create a program to just use it one single time.
- This is similar to the real life situation of work delegation again.
- If you were a chef, you basically "input" the "program" *cook potatoes* into the junior trainee once.

Programming

- In the vast majority of situations, we do not create a program to just use it one single time.
- This is similar to the real life situation of work delegation again.
- If you were a chef, you basically "input" the "program" *cook potatoes* into the junior trainee once.
- In the future, you would like to be able go to them and invoke this program again by saying: "Please cook 2kg of potatoes."

Programming

- In the vast majority of situations, we do not create a program to just use it one single time.
- This is similar to the real life situation of work delegation again.
- If you were a chef, you basically "input" the "program" *cook potatoes* into the junior trainee once.
- In the future, you would like to be able go to them and invoke this program again by saying: "Please cook 2kg of potatoes."
- Our "programs" often even have implicit parameters, like the quantity of 2kg mentioned above.

Programming

- In the vast majority of situations, we do not create a program to just use it one single time.
- This is similar to the real life situation of work delegation again.
- If you were a chef, you basically "input" the "program" *cook potatoes* into the junior trainee once.
- In the future, you would like to be able go to them and invoke this program again by saying: "Please cook 2kg of potatoes."
- Our "programs" often even have implicit parameters, like the quantity of 2kg mentioned above.
- Maybe you go to the hairdresser again and want to say: "Same as usual, but today color it blue."

Programming

- In our day-to-day interactions, creating reusable and parameterized programs happens very often and very implicitly.

Programming

- In our day-to-day interactions, creating reusable and parameterized programs happens very often and very implicitly.
- We usually do not think about this in any explicit terms.

Programming

- In our day-to-day interactions, creating reusable and parameterized programs happens very often and very implicitly.
- We usually do not think about this in any explicit terms.
- But when we program computers, we do think about this explicitly.

Programming

- In our day-to-day interactions, creating reusable and parameterized programs happens very often and very implicitly.
- We usually do not think about this in any explicit terms.
- But when we program computers, we do think about this explicitly.
- Right from the start.

Programming

- In our day-to-day interactions, creating reusable and parameterized programs happens very often and very implicitly.
- We usually do not think about this in any explicit terms.
- But when we program computers, we do think about this explicitly.
- Right from the start.
- Therefore programming is only one part of software development.

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.
- Is it that easy?

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.
- Is it that easy? No.

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.
- Is it that easy? No.
  1. You may wonder whether you made any mistake.

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.
- Is it that easy? No.
  1. You may wonder whether you made any mistake. People make mistakes.

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.
- Is it that easy? No.
  1. You may wonder whether you made any mistake. People make mistakes. The more complex the task we tackle, the more (program code) we write, the more likely it is that we make some small error somewhere.

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.
- Is it that easy? No.
  1. You may wonder whether you made any mistake. People make mistakes. The more complex the task we tackle, the more (program code) we write, the more likely it is that we make some small error somewhere. You must test your program.

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.
- Is it that easy? No.
  1. You may wonder whether you made any mistake. People make mistakes. The more complex the task we tackle, the more (program code) we write, the more likely it is that we make some small error somewhere. You must test your program.
  2. What if someone else is going to use your program later?

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.
- Is it that easy? No.
  1. You may wonder whether you made any mistake. People make mistakes. The more complex the task we tackle, the more (program code) we write, the more likely it is that we make some small error somewhere. You must test your program.
  2. What if someone else is going to use your program later? You need to write clear documentation.

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.
- Is it that easy? No.
  1. You may wonder whether you made any mistake. People make mistakes. The more complex the task we tackle, the more (program code) we write, the more likely it is that we make some small error somewhere. You must test your program.
  2. What if someone else is going to use your program later? You need to write clear documentation.
  3. What if your program or packages provides functions that others can use?

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.
- Is it that easy? No.
  1. You may wonder whether you made any mistake. People make mistakes. The more complex the task we tackle, the more (program code) we write, the more likely it is that we make some small error somewhere. You must test your program.
  2. What if someone else is going to use your program later? You need to write clear documentation.
  3. What if your program or packages provides functions that others can use? The input and output datatypes must be clearly specified.

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.
- Is it that easy? No.
  1. You may wonder whether you made any mistake. People make mistakes. The more complex the task we tackle, the more (program code) we write, the more likely it is that we make some small error somewhere. You must test your program.
  2. What if someone else is going to use your program later? You need to write clear documentation.
  3. What if your program or packages provides functions that others can use? The input and output datatypes must be clearly specified.
  4. What if someone else is supposed to read your code and work with it?

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.
- Is it that easy? No.
  1. You may wonder whether you made any mistake. People make mistakes. The more complex the task we tackle, the more (program code) we write, the more likely it is that we make some small error somewhere. You must test your program.
  2. What if someone else is going to use your program later? You need to write clear documentation.
  3. What if your program or packages provides functions that others can use? The input and output datatypes must be clearly specified.
  4. What if someone else is supposed to read your code and work with it? Your code must be readable, clear, and follow common coding styles[24].

Developing Software

- Later in your job, you want to develop a program that can be used to solve a specific task.
  1. You write the program.
  2. You now have the file with the program code.
  3. The problem is solved.
- Is it that easy? No.
  1. You may wonder whether you made any mistake. People make mistakes. The more complex the task we tackle, the more (program code) we write, the more likely it is that we make some small error somewhere. You must test your program.
  2. What if someone else is going to use your program later? You need to write clear documentation.
  3. What if your program or packages provides functions that others can use? The input and output datatypes must be clearly specified.
  4. What if someone else is supposed to read your code and work with it? Your code must be readable, clear, and follow common coding styles[24].
- All of these things must be considered!

Developing Software

- Developing software is more than writing programs.

Developing Software

- Developing software is more than writing programs.
- Most jobs are more than just the associated “main work”

Developing Software

- Developing software is more than writing programs.
- Most jobs are more than just the associated "main work"
  - Let's say that you need to go to a doctor to undergo some procedure.

Developing Software

- Developing software is more than writing programs.
- Most jobs are more than just the associated “main work”
  - Let’s say that you need to go to a doctor to undergo some procedure.
  - You hope that they have been trained well in doing operations.

Developing Software

- Developing software is more than writing programs.
- Most jobs are more than just the associated "main work"
  - Let's say that you need to go to a doctor to undergo some procedure.
  - You hope that they have been trained well in doing operations.
  - But you simply expect that they were also trained to wash their hands before surgery.

Developing Software

- Developing software is more than writing programs.
- Most jobs are more than just the associated "main work"
  - Let's say that you need to go to a doctor to undergo some procedure.
  - You hope that they have been trained well in doing operations.
  - But you simply expect that they were also trained to wash their hands before surgery.
- It is the same for programmers!

Developing Software

- Developing software is more than writing programs.
- Most jobs are more than just the associated "main work"
  - Let's say that you need to go to a doctor to undergo some procedure.
  - You hope that they have been trained well in doing operations.
  - But you simply expect that they were also trained to wash their hands before surgery.
- It is the same for programmers!
  - Let's say that your boss asks you to write a program.

Developing Software

- Developing software is more than writing programs.
- Most jobs are more than just the associated "main work"
  - Let's say that you need to go to a doctor to undergo some procedure.
  - You hope that they have been trained well in doing operations.
  - But you simply expect that they were also trained to wash their hands before surgery.
- It is the same for programmers!
  - Let's say that your boss asks you to write a program.
  - They hope that you can write a program that "works."

Developing Software

- Developing software is more than writing programs.
- Most jobs are more than just the associated "main work"
  - Let's say that you need to go to a doctor to undergo some procedure.
  - You hope that they have been trained well in doing operations.
  - But you simply expect that they were also trained to wash their hands before surgery.
- It is the same for programmers!
  - Let's say that your boss asks you to write a program.
  - They hope that you can write a program that "works."
  - But they expect that the code that you produced is readable, was tested, and is documented.

Developing Software

- Developing software is more than writing programs.
- Most jobs are more than just the associated "main work"
  - Let's say that you need to go to a doctor to undergo some procedure.
  - You hope that they have been trained well in doing operations.
  - But you simply expect that they were also trained to wash their hands before surgery.
- It is the same for programmers!
  - Let's say that your boss asks you to write a program.
  - They hope that you can write a program that "works."
  - But they expect that the code that you produced is readable, was tested, and is documented.
- I do not want to go to a surgeon who does not wash their hands before operating on me.

Developing Software

- Developing software is more than writing programs.
- Most jobs are more than just the associated "main work"
  - Let's say that you need to go to a doctor to undergo some procedure.
  - You hope that they have been trained well in doing operations.
  - But you simply expect that they were also trained to wash their hands before surgery.
- It is the same for programmers!
  - Let's say that your boss asks you to write a program.
  - They hope that you can write a program that "works."
  - But they expect that the code that you produced is readable, was tested, and is documented.
- I do not want to go to a surgeon who does not wash their hands before operating on me.
- And I will not teach you programming without emphasizing code cleanliness.

Developing Software

- Programmers don't just write programs, they *develop software*.

Developing Software

- Programmers don't just write programs, they *develop software*.
- A good share of programmers usually spend only about 50% of their time with programming[6,14].

Developing Software

- Programmers don't just write programs, they *develop software*.
- A good share of programmers usually spend only about 50% of their time with programming[6,14].
- Other studies even suggest that less than 20% of the working time spent with coding, maybe with another 15% of bug fixing[15].

Developing Software

- Programmers don't just write programs, they *develop software*.
- A good share of programmers usually spend only about 50% of their time with programming[6,14].
- Other studies even suggest that less than 20% of the working time spent with coding, maybe with another 15% of bug fixing[15].
- Of course, we will focus on programming in this class on, well, programming.

Developing Software

- Programmers don't just write programs, they *develop software*.
- A good share of programmers usually spend only about 50% of their time with programming[6,14].
- Other studies even suggest that less than 20% of the working time spent with coding, maybe with another 15% of bug fixing[15].
- Of course, we will focus on programming in this class on, well, programming.
- But we will at discuss several issues beyond that, things that belong into your tool belt, that can make you a *good* programmer.

Developing Software

- Programmers don't just write programs, they *develop software*.
- A good share of programmers usually spend only about 50% of their time with programming[6,14].
- Other studies even suggest that less than 20% of the working time spent with coding, maybe with another 15% of bug fixing[15].
- Of course, we will focus on programming in this class on, well, programming.
- But we will at discuss several issues beyond that, things that belong into your tool belt, that can make you a *good* programmer.
- Our course is on developing good software with Python.

Why Python?

Why Python?

1. Because Python is a very widely-used programming language[3,4].

Why Python?

1. Because Python is a very widely-used programming language[3,4].

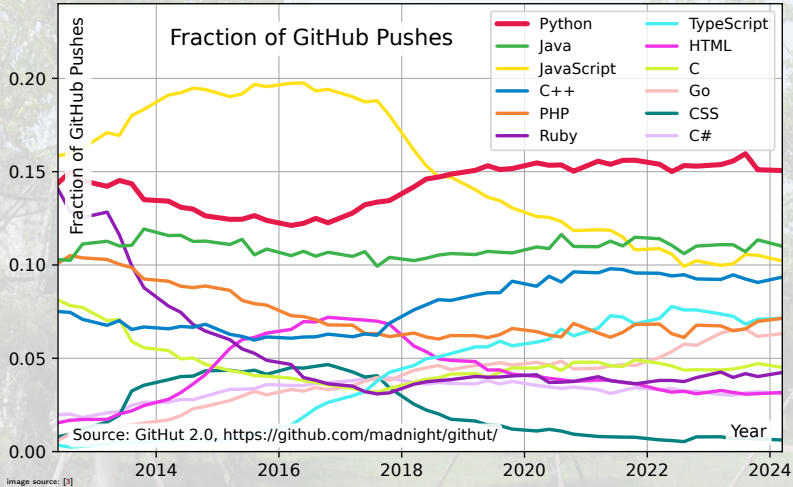


image source: [3]

Why Python?

1. Because Python is a very widely-used programming language[3,4].
2. Python is intensely used in AI[21], ML[22], and Data Science[8] as well as optimization, which are among the most important areas of future technology.

Why Python?

1. Because Python is a very widely-used programming language[3,4].
2. Python is intensely used in AI[21], ML[22], and Data Science[8] as well as optimization, which are among the most important areas of future technology.
3. There exists a very large set of powerful libraries supporting both research and application development in these fields, including NumPy[5,9,11], Pandas[2,13], Scikit-learn[18,20], SciPy[11,25], TensorFlow[1,12], PyTorch[17,20], Matplotlib[10,11,16], SimPy[27], and moptipy[26], just to name a few.

Why Python?

1. Because Python is a very widely-used programming language[3,4].
2. Python is intensely used in AI[21], ML[22], and Data Science[8] as well as optimization, which are among the most important areas of future technology.
3. There exists a very large set of powerful libraries supporting both research and application development in these fields, including NumPy[5,9,11], Pandas[2,13], Scikit-learn[18,20], SciPy[11,25], TensorFlow[1,12], PyTorch[17,20], Matplotlib[10,11,16], SimPy[27], and moptipy[26], just to name a few.
4. Python is very easy to learn[7,23]. It has a simple and clean syntax and enforces a readable structure of programs. Python has expressive built-in types likes lists, tuples, and dictionaries.

Python is an interpreted language

- Most programming languages require code to be compiled.

Python is an interpreted language

- Most programming languages require code to be compiled.

Programming in a compiled language like C

Python is an interpreted language

- Most programming languages require code to be compiled.

Programming in a compiled language like C

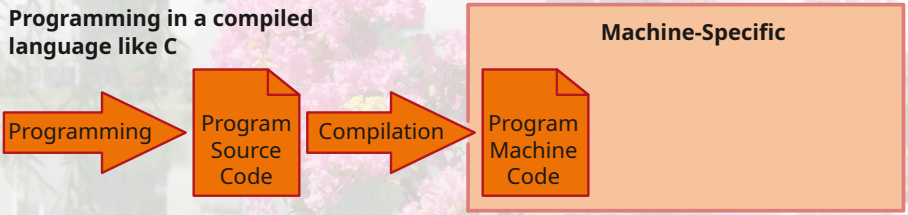
Python is an interpreted language

- Most programming languages require code to be compiled.

Programming in a compiled language like C

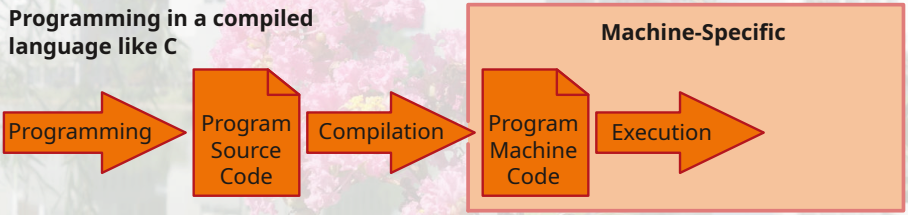
Python is an interpreted language

- Most programming languages require code to be compiled.



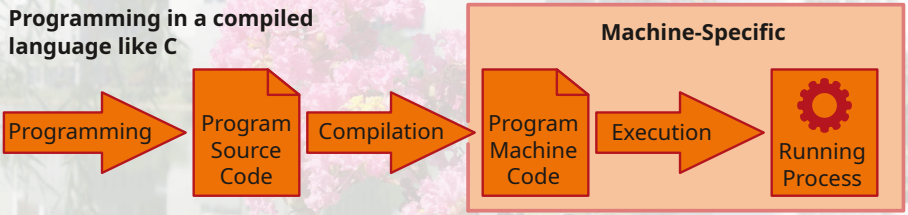
Python is an interpreted language

- Most programming languages require code to be compiled.



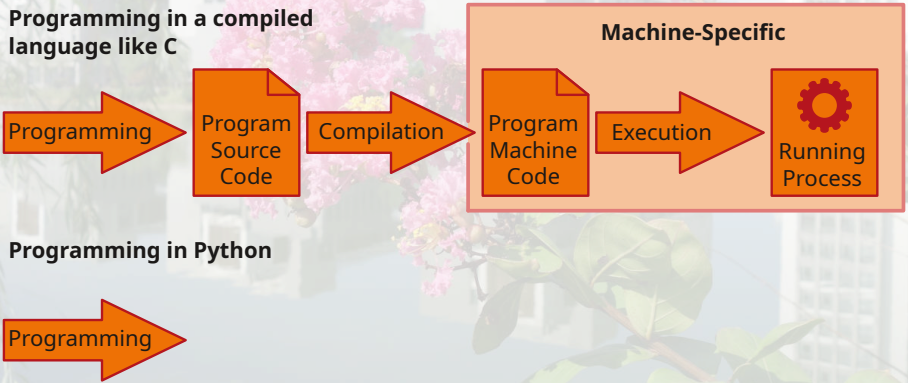
Python is an interpreted language

- Most programming languages require code to be compiled.



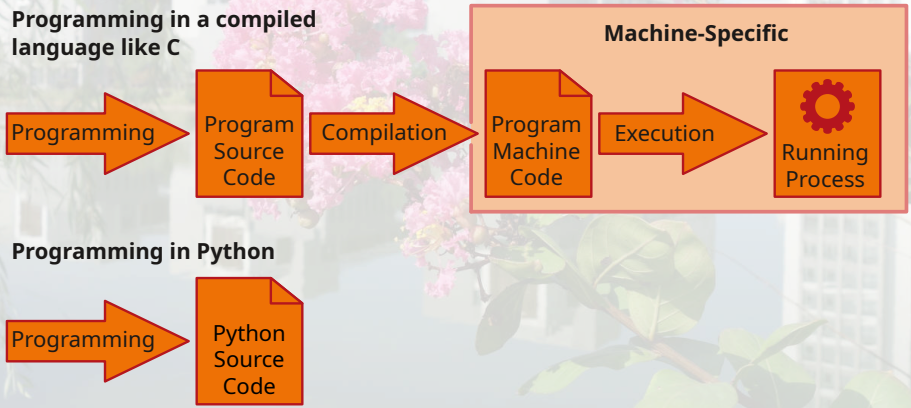
Python is an interpreted language

- Most programming languages require code to be compiled.
- Python is interpreted.



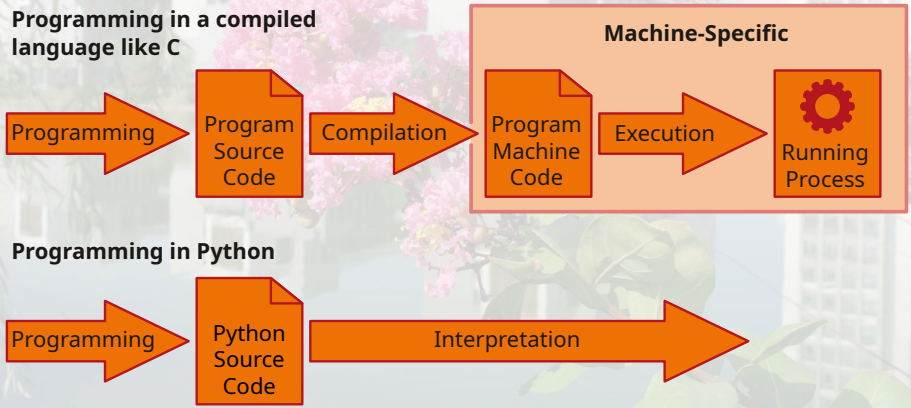
Python is an interpreted language

- Most programming languages require code to be compiled.
- Python is interpreted.



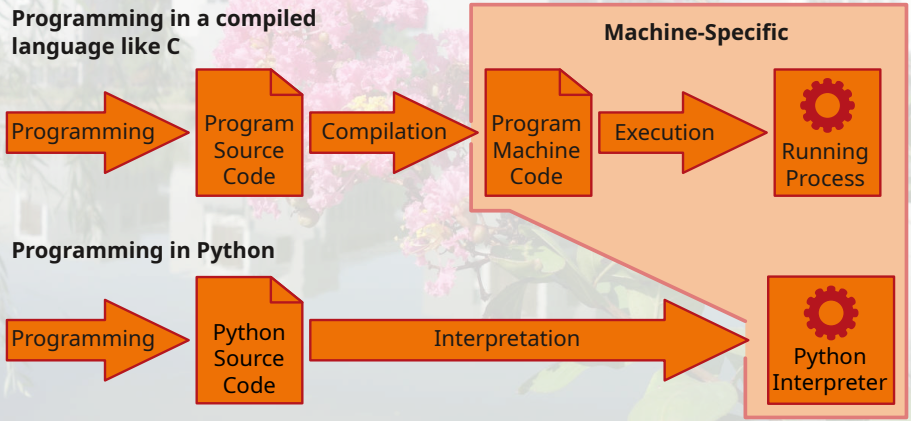
Python is an interpreted language

- Most programming languages require code to be compiled.
- Python is interpreted.



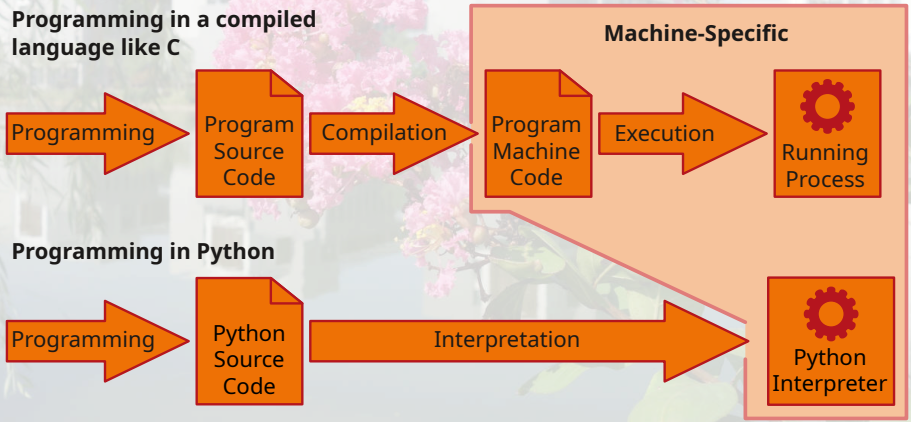
Python is an interpreted language

- Most programming languages require code to be compiled.
- Python is interpreted.



Python is an interpreted language

- Most programming languages require code to be compiled.
- Python is interpreted.
- So there are fewer steps in the build process.



Summary

Summary

- Programming means to write the source code of computer programs.

Summary

- Programming means to write the source code of computer programs.
- We can use a programming language like Python for that.

Summary

- Programming means to write the source code of computer programs.
- We can use a programming language like Python for that.
- To be able to create good, useful, and maintainable programs, it is not enough to just learn a programming language.

Summary

- Programming means to write the source code of computer programs.
- We can use a programming language like Python for that.
- To be able to create good, useful, and maintainable programs, it is not enough to just learn a programming language.
- You also have to understand the tools surrounding it, the best practices, the coding guidelines, how to test programs, how to document programs, and so on.

Summary

- Programming means to write the source code of computer programs.
- We can use a programming language like Python for that.
- To be able to create good, useful, and maintainable programs, it is not enough to just learn a programming language.
- You also have to understand the tools surrounding it, the best practices, the coding guidelines, how to test programs, how to document programs, and so on.
- You need a good understanding of the most important components of *software development*.

Summary

- Programming means to write the source code of computer programs.
- We can use a programming language like Python for that.
- To be able to create good, useful, and maintainable programs, it is not enough to just learn a programming language.
- You also have to understand the tools surrounding it, the best practices, the coding guidelines, how to test programs, how to document programs, and so on.
- You need a good understanding of the most important components of *software development*.
- I will try to teach you programming together with several of such aspects.

Summary

- Programming means to write the source code of computer programs.
- We can use a programming language like Python for that.
- To be able to create good, useful, and maintainable programs, it is not enough to just learn a programming language.
- You also have to understand the tools surrounding it, the best practices, the coding guidelines, how to test programs, how to document programs, and so on.
- You need a good understanding of the most important components of *software development*.
- I will try to teach you programming together with several of such aspects.
- We will use the Python programming language

Summary

- Programming means to write the source code of computer programs.
- We can use a programming language like Python for that.
- To be able to create good, useful, and maintainable programs, it is not enough to just learn a programming language.
- You also have to understand the tools surrounding it, the best practices, the coding guidelines, how to test programs, how to document programs, and so on.
- You need a good understanding of the most important components of *software development*.
- I will try to teach you programming together with several of such aspects.
- We will use the Python programming language, because it is easy to learn

Summary

- Programming means to write the source code of computer programs.
- We can use a programming language like Python for that.
- To be able to create good, useful, and maintainable programs, it is not enough to just learn a programming language.
- You also have to understand the tools surrounding it, the best practices, the coding guidelines, how to test programs, how to document programs, and so on.
- You need a good understanding of the most important components of *software development*.
- I will try to teach you programming together with several of such aspects.
- We will use the Python programming language, because it is easy to learn, widely used

Summary

- Programming means to write the source code of computer programs.
- We can use a programming language like Python for that.
- To be able to create good, useful, and maintainable programs, it is not enough to just learn a programming language.
- You also have to understand the tools surrounding it, the best practices, the coding guidelines, how to test programs, how to document programs, and so on.
- You need a good understanding of the most important components of *software development*.
- I will try to teach you programming together with several of such aspects.
- We will use the Python programming language, because it is easy to learn, widely used, has a rich environment of useful packages

Summary

- Programming means to write the source code of computer programs.
- We can use a programming language like Python for that.
- To be able to create good, useful, and maintainable programs, it is not enough to just learn a programming language.
- You also have to understand the tools surrounding it, the best practices, the coding guidelines, how to test programs, how to document programs, and so on.
- You need a good understanding of the most important components of *software development*.
- I will try to teach you programming together with several of such aspects.
- We will use the Python programming language, because it is easy to learn, widely used, has a rich environment of useful packages, and has a simple build process.

谢谢您们

Thank you

References I

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. "TensorFlow: A System for Large-Scale Machine Learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)*. Nov. 2–4, 2016, Savannah, GA, USA. Ed. by Kimberly Keeton and Timothy Roscoe. Berkeley, CA, USA: USENIX Association. Berkeley, CA, USA: USENIX Association: USENIX Association, 2016, pp. 265–283. ISBN: 978-1-931971-33-1. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi (visited on 2024-06-26) (cit. on pp. 67–71).

[2] David M. Beazley. "Data Processing with Pandas". *;login: Usenix Magazin* 37(6), Dec. 2012. Berkeley, CA, USA: USENIX Association. ISSN: 1044-6397. URL: https://www.usenix.org/publications/login/december-2012-volume-37-number-6/data-processing-pandas (visited on 2024-06-25) (cit. on pp. 67–71).

[3] Fabian Beuke. *GitHut 2.0: GitHub Language Statistics*. San Francisco, CA, USA: GitHub Inc, 2023. URL: https://madnight.github.io/githut (visited on 2024-06-24) (cit. on pp. 67–71).

[4] Oscar Castro, Pierrick Bruneau, Jean-Sébastien Sottet, and Dario Torregrossa. "Landscape of High-Performance Python to Develop Data Science and Machine Learning Applications". *ACM Computing Surveys (CSUR)* 56(3):65:1–65:30, 2024. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0360-0300. doi:10.1145/3617588 (cit. on pp. 67–71).

[5] Justin Dennison, Cherokee Boose, and Peter van Rysdam. *Intro to NumPy*. Centennial, CO, USA: ACI Learning. Birmingham, England, UK: Packt Publishing Ltd, June 2024. ISBN: 978-1-83620-863-1 (cit. on pp. 67–71).

[6] *Developer Survey 2019: Open Source Runtime Pains*. Vancouver, BC, Canada: ActiveState Software Inc., Apr. 30, 2019. URL: https://cdn.activestate.com/wp-content/uploads/2019/05/ActiveState-Developer-Survey-2019-Open-Source-Runtime-Pains.pdf (visited on 2024-12-29) (cit. on pp. 60–65).

[7] Linda Grandell, Mia Peltomäki, Ralph-Johan Back, and Tapio Salakoski. "Why complicate things? Introducing Programming in High School using Python". In: *Proceedings of the 8th Australasian Conference on Computing Education (ACE'06)*. Jan. 16–19, 2006, Hobart, TAS, Australia. Ed. by Denise Tolhurst and Samuel Mann. Vol. 52. New York, NY, USA: Association for Computing Machinery (ACM), 2006, pp. 71–80. ISBN: 978-1-920682-34-7. doi:10.5555/1151869.1151880 (cit. on pp. 67–71).

References II

[8] Joel Grus. *Data Science from Scratch: First Principles with Python*. 2nd ed. Sebastopol, CA, USA: O'Reilly Media, Inc., May 2019. ISBN: 978-1-4920-4113-9 (cit. on pp. 67–71).

[9] Charles R. Harris, K. Jarrod Millman, Stéfan van der Walt, Ralf Gommers, Pauli "pv" Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. "Array programming with NumPy". *Nature* 585:357–362, 2020. London, England, UK: Springer Nature Limited. ISSN: 0028-0836. doi:10.1038/S41586-020-2649-2 (cit. on pp. 67–71).

[10] John D. Hunter. "Matplotlib: A 2D Graphics Environment". *Computing in Science & Engineering* 9(3):90–95, May–June 2007. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1521-9615. doi:10.1109/MCSE.2007.55 (cit. on pp. 67–71).

[11] Robert Johansson. *Numerical Python: Scientific Computing and Data Science Applications with NumPy, SciPy and Matplotlib*. New York, NY, USA: Apress Media, LLC, Dec. 2018. ISBN: 978-1-4842-4246-9 (cit. on pp. 67–71).

[12] Charles Landau. *TensorFlow Deep Dive: Build, Train, and Deploy Machine Learning Models with TensorFlow*. Sebastopol, CA, USA: O'Reilly Media, Inc., Dec. 2023 (cit. on pp. 67–71).

[13] Reuven M. Lerner. *Pandas Workout*. Shelter Island, NY, USA: Manning Publications, June 2024. ISBN: 978-1-61729-972-8 (cit. on pp. 67–71).

[14] *Making Open Source Work Better for Developers: Highlights of the 2019 Tidelift Managed Open Source Survey*. Boston, MA, USA: Tidelift, Inc., June 2019. URL: https://tidelift.com/subscription/managed-open-source-survey (visited on 2024-12-29) (cit. on pp. 60–65).

[15] Pedro Mejia Alvarez, Raul E. Gonzalez Torres, and Susana Ortega Cisneros. *Exception Handling – Fundamentals and Programming*. SpringerBriefs in Computer Science. Cham, Switzerland: Springer. Cham, Switzerland: Springer: Springer, Feb. 2024. ISSN: 2191-5768. ISBN: 978-3-031-50680-2. doi:10.1007/978-3-031-50681-9 (cit. on pp. 60–65).

[16] Ashwin Pajankar. *Hands-on Matplotlib: Learn Plotting and Visualizations with Python 3*. New York, NY, USA: Apress Media, LLC, Nov. 2021. ISBN: 978-1-4842-7410-1 (cit. on pp. 67–71).

References III

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019 (NeurIPS'19)*. Dec. 8–14, 2019, Vancouver, BC, Canada. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. San Diego, CA, USA: The Neural Information Processing Systems Foundation (NeurIPS), 2019, pp. 8024–8035. ISBN: 978-1-7138-0793-3. URL: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html (visited on 2024-07-18) (cit. on pp. 67–71).

[18] Fabian Pedregos, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research (JMLR)* 12:2825–2830, Oct. 2011. Cambridge, MA, USA: MIT Press. ISSN: 1532-4435. doi:10.5555/1953048.2078195 (cit. on pp. 67–71).

[19] "programming: Meaning of *programming* in English". In: *Cambridge Dictionary English (UK)*. Cambridge, England, UK: Cambridge University Press & Assessment, June 2024. URL: https://dictionary.cambridge.org/dictionary/english/programming (visited on 2024-06-17) (cit. on p. 20).

[20] Sebastian Raschka, Yuxi Liu, and Vahid Mirjalili. *Machine Learning with PyTorch and Scikit-learn*. Birmingham, England, UK: Packt Publishing Ltd, Feb. 2022. ISBN: 978-1-80181-931-2 (cit. on pp. 67–71).

[21] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (AIMA)*. 4th ed. Hoboken, NJ, USA: Pearson Education, Inc. ISBN: 978-1-292-40113-3. URL: https://aima.cs.berkeley.edu (visited on 2024-06-27) (cit. on pp. 67–71).

[22] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, England, UK: Cambridge University Press & Assessment, July 2014. ISBN: 978-1-107-05713-5. URL: http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning (visited on 2024-06-27) (cit. on pp. 67–71).

References IV

[23] Guido van Rossum. *Computer Programming for Everybody (Revised Proposal). A Scouting Expedition for the Programmers of Tomorrow*. CNRI Proposal 90120-1a. Reston, VA, USA: Corporation for National Research Initiatives (CNRI), July 1999. URL: https://www.python.org/doc/essays/cp4e (visited on 2024-06-27) (cit. on pp. 67–71).

[24] Guido van Rossum, Barry Warsaw, and Alyssa Coghlan. *Style Guide for Python Code*. Python Enhancement Proposal (PEP) 8. Beaverton, OR, USA: Python Software Foundation (PSF), July 5, 2001. URL: https://peps.python.org/pep-0008 (visited on 2024-07-27) (cit. on pp. 32–48).

[25] Pauli "pv" Virtanen, Ralf Gommers, Travis E. Oliphant, Matt "mdhaber" Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, Ilhan "ilayn" Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregos, Paul van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". *Nature Methods* 17:261–272, Mar. 2, 2020. London, England, UK: Springer Nature Limited. ISSN: 1548-7091. doi:10.1038/s41592-019-0686-2. URL: http://arxiv.org/abs/1907.10121 (visited on 2024-06-26). See also arXiv:1907.10121v1 [cs.MS] 23 Jul 2019. (Cit. on pp. 67–71).

[26] Thomas Weise (汤卫思) and Zhize Wu (吴志泽). "Replicable Self-Documenting Experiments with Arbitrary Search Spaces and Algorithms". In: *Companion Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'23), Companion Volume*. July 15–19, 2023, Lisbon, Portugal. Ed. by Sara Silva and Luís Paquete. New York, NY, USA: Association for Computing Machinery (ACM). New York, NY, USA: Association for Computing Machinery (ACM): Association for Computing Machinery (ACM), 2023, pp. 1891–1899. ISBN: 979-8-4007-0120-7. doi:10.1145/3583133.3596306 (cit. on pp. 67–71).

[27] Dmitry Zinoviev. *Discrete Event Simulation: It's Easy with SimPy!* arXiv.org: Computing Research Repository (CoRR) abs/2405.01562. Ithaca, NY, USA: Cornell Universiy Library, Apr. 3, 2024. doi:10.48550/ARXIV.2405.01562. URL: https://arxiv.org/abs/2405.01562 (visited on 2024-06-27). arXiv:2405.01562v1 [cs.MS] 3 Apr 2024 (cit. on pp. 67–71).