

Functions, Modules, and Data Manipulation in Python

In this presentation, we will explore the power and versatility of functions, modules, and data manipulation techniques in Python programming.

 by **Vikas Mishra**



Introduction to Functions

Functions are reusable blocks of code that perform specific tasks. They make code more modular and easier to read and maintain.

1 Benefits of Functions

Functions allow us to break down complex problems into smaller, more manageable tasks. They also promote code reuse and modularity.

2 Syntax for Creating Functions

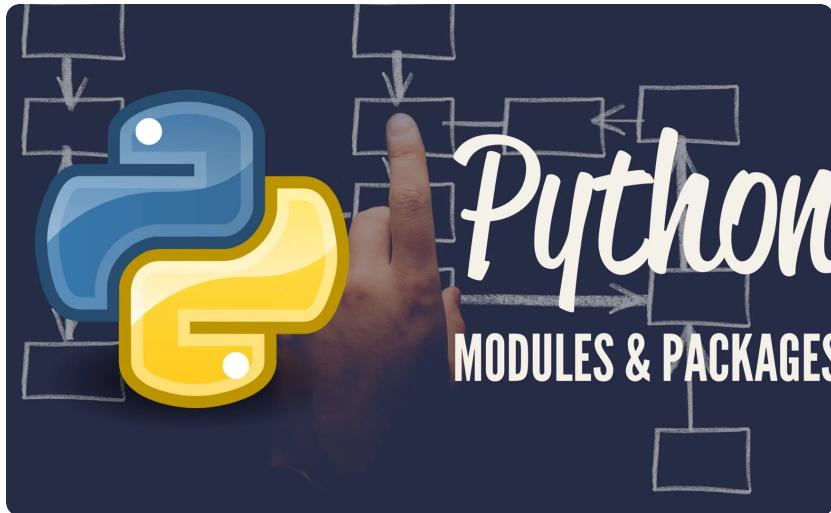
A function is created using the 'def' keyword, followed by the function name and any parameters it requires.

3 Example Functions

Some example uses of Python functions include calculating areas, printing outputs, and sorting data.

Working with Modules

Modules are collections of reusable code that can be imported into our programs. They save time and effort by providing pre-written functionality that we can use.

A screenshot of a terminal window in a dark-themed code editor. The code editor shows a file named "Hello World.py" with the line `1 print("Hello, World!")` highlighted with a red box. The terminal window below shows the command `PS C:\Users\suppo> & C:/Users/suppo/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/suppo/OneDrive/Documents/Python Sources/Hello World.py"`, followed by the output "Hello, World!". The terminal window has a red border around the command and output lines.

Standard Modules

Python comes with a rich selection of built-in modules that we can use to extend our programming capabilities.

Third-Party Modules

Python's extensive library of third-party modules gives us access to a wealth of powerful tools and features.

Data Manipulation Techniques

Data manipulation involves transforming data to extract meaningful insights and make better-informed decisions.

Common Data Types in Python

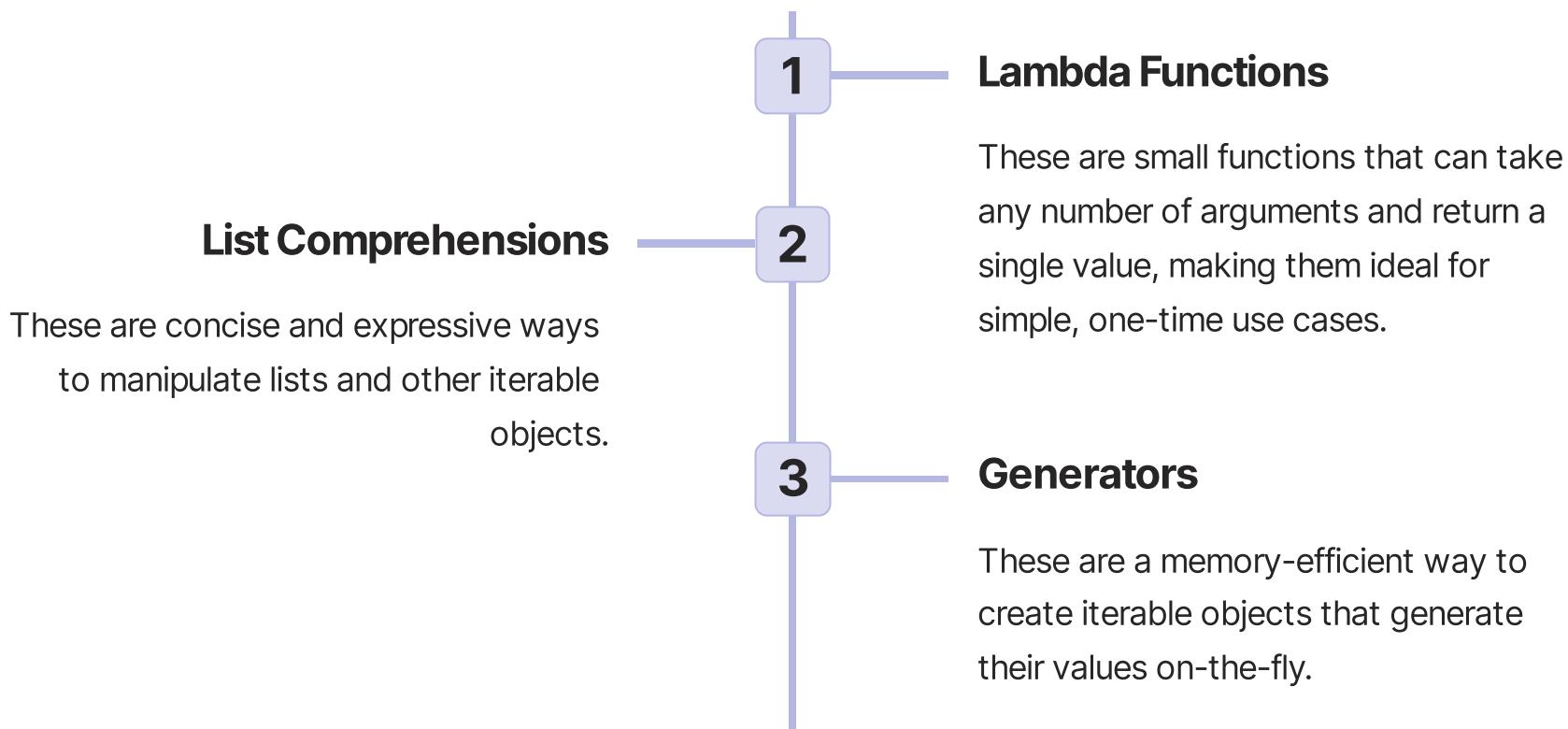
Some common data types include strings, numbers, and lists, each with its own unique properties and uses.

Built-in Functions for Manipulating Data

Python's rich library of built-in functions makes it easy to manipulate data in a variety of ways, from sorting and filtering to performing calculations and statistical analyses.

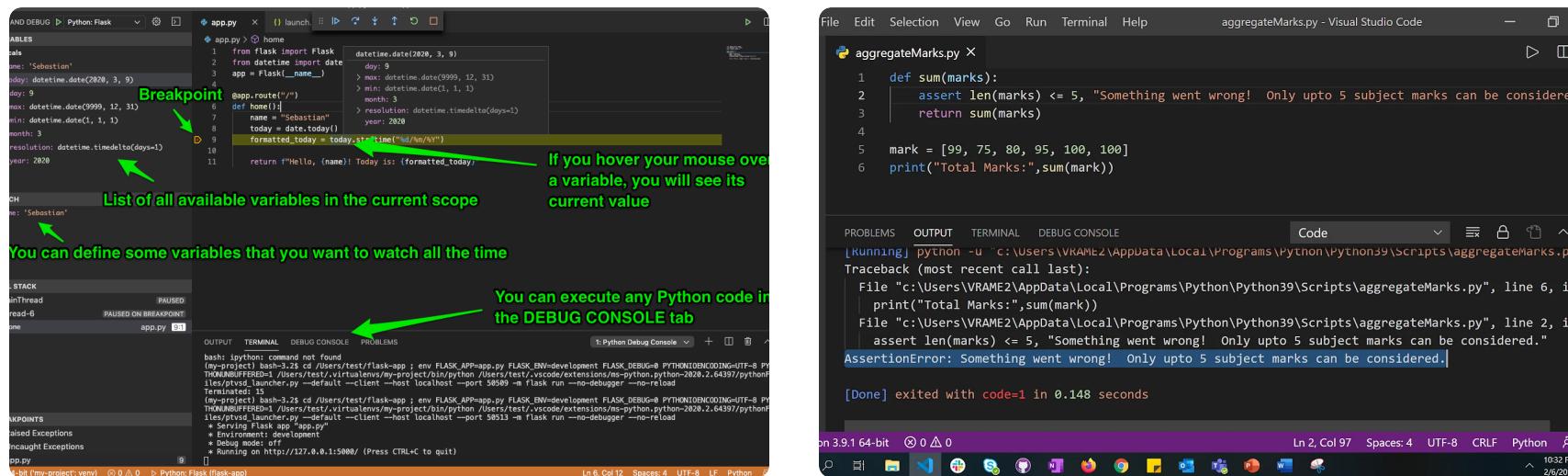
Functional Programming in Python

Functional programming is a powerful paradigm for building scalable, efficient, and maintainable software systems. In Python, this includes lambda functions, comprehensions, and generators.



Error Handling and Debugging

Python provides several powerful tools for detecting and resolving errors in our code, including try/except statements, assertions, and debugging tools like pdb.



The screenshot shows the Visual Studio Code interface with the Python extension. A breakpoint is set on line 8 of `app.py`. The code defines a Flask application with a `home` route that prints the current date. The `Variables` sidebar shows the current scope variables: `name` (Sebastian), `today` (datetime.date(2028, 3, 9)), `year` (2028), and `formatted_today` (the formatted date string). The `Breakpoints` sidebar shows the breakpoint at line 8. The `Stack` sidebar shows the current stack frame. The `DEBUG CONSOLE` tab shows the output of the Python debugger, including the command to start the Flask app and the resulting URL. A tooltip explains that hovering over a variable shows its current value.

File Edit Selection View Go Run Terminal Help aggregateMarks.py - Visual Studio Code

Variables

```
name: 'Sebastian'  
today: datetime.date(2028, 3, 9)  
year: 2028  
month: 3  
resolution: datetime.timedelta(days=1)  
formatted_today: datetime.datetime(2028, 3, 9, 0, 0, 0)  
Breakpoint  
List of all available variables in the current scope
```

Stack

```
inThread  
read-6 PAUSED ON BREAKPOINT  
None app.py [81]
```

Breakpoints

```
raised Exceptions  
caught Exceptions  
app.py [81]
```

DEBUG CONSOLE

```
hash: python: command not found  
(my-project) bash-3.2$ cd /Users/test/flask-app ; env FLASK_APP=app.py FLASK_ENV=development FLASK_DEBUG=0 PYTHONUNBUFFERED=1 /Users/test/virtualenvs/my-project/bin/python /Users/test/vscode/extensions/python.python-2028.2.44397/python -m flask run --no-debugger --no-reload  
Terminated: 15  
(my-project) bash-3.2$ cd /Users/test/flask-app ; env FLASK_APP=app.py FLASK_ENV=development FLASK_DEBUG=0 PYTHONUNBUFFERED=1 /Users/test/virtualenvs/my-project/bin/python /Users/test/vscode/extensions/python.python-2028.2.44397/python -m flask run --no-debugger --no-reload  
* Serving Flask app "app:app"  
* Development mode: off  
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
[Running] Python -u "c:\Users\VRAME2\AppData\Local\Programs\Python\Python39\Scripts\aggregateMarks.py"  
Traceback (most recent call last):  
  File "c:\Users\VRAME2\AppData\Local\Programs\Python\Python39\Scripts\aggregateMarks.py", line 6, in   
    print("Total Marks:",sum(mark))  
  File "c:\Users\VRAME2\AppData\Local\Programs\Python\Python39\Scripts\aggregateMarks.py", line 2, in   
    assert len(marks) <= 5, "Something went wrong! Only upto 5 subject marks can be considered."  
AssertionError: Something went wrong! Only upto 5 subject marks can be considered.  
[Done] exited with code=1 in 0.148 seconds
```

Ln 6, Col 12 Spaces: 4 UTF-8 CRLF Python

Try/Except Statements

These allow us to gracefully handle errors in our code by anticipating potential problems and taking steps to correct them.

Assertions

These are ways to ensure that key pieces of our code are functioning correctly, by testing and confirming assumptions about our data and functions.

Advanced Techniques

Python is a versatile language that can be used for a wide range of applications, from web development and machine learning to data analysis and game design.

1

Web Development with Python

Python's Flask and Django frameworks provide powerful tools for creating web applications and APIs.

2

Data Science and Machine Learning

Python offers extensive libraries for data analysis and machine learning, such as Pandas, NumPy, and TensorFlow.

3

Game Design with Python

Python's Pygame library provides tools for developing 2D games with intuitive interfaces and smooth mechanics.



Made with Gamma

Conclusion

Functions, modules, and data manipulation techniques are fundamental building blocks of modern software development. Mastery of these powerful tools is essential for creating robust, scalable, and efficient applications.

Thank you for attending our presentation, we hope you found it informative and engaging.