## 📘 Beginner Level

### 1. ****Hello World****

* **Task:** Write a Python program that prints "Hello, World!" to the console.
* **Concepts Covered:** Basic syntax, print function.

### 2. ****Basic Arithmetic Operations****

* **Task:** Create a program that takes two numbers as input and prints their sum, difference, product, and quotient.
* **Concepts Covered:** Variables, data types, input/output, arithmetic operators.

### 3. ****Variables and Data Types****

* **Task:** Define variables of different data types (integer, float, string, boolean) and print their types using the type() function.
* **Concepts Covered:** Data types, type checking.

### 4. ****Control Structures****

* **Task:** Write a program that takes a number from the user and prints whether it's positive, negative, or zero.
* **Concepts Covered:** if, elif, else statements.

### 5. ****Loops****

* **Task:** Print all even numbers from 1 to 100 using a for loop.
* **Concepts Covered:** for loops, range function, conditional statements.

### 6. ****Lists****

* **Task:** Create a list of your favorite fruits and print each fruit using a loop.
* **Concepts Covered:** Lists, looping through lists.

### 7. ****Functions****

* **Task:** Write a function that takes a list of numbers and returns the largest number.
* **Concepts Covered:** Function definition, return values, list manipulation.

### 8. ****String Manipulation****

* **Task:** Write a program that takes a string and returns it reversed.
* **Concepts Covered:** String slicing, functions.

### 9. ****Simple Calculator****

* **Task:** Create a simple calculator that can perform addition, subtraction, multiplication, and division based on user input.
* **Concepts Covered:** Functions, user input, control flow.

## 📗 Intermediate Level

### 1. ****Dictionaries****

* **Task:** Create a dictionary to store student names and their grades. Write a program to print each student's name and grade.
* **Concepts Covered:** Dictionaries, looping through key-value pairs.

### 2. ****File I/O****

* **Task:** Write a program to read a text file and count the number of words in it.
* **Concepts Covered:** File handling, reading files, string processing.

### 3. ****Exception Handling****

* **Task:** Write a program that prompts the user to enter two numbers and divides them, handling division by zero errors gracefully.
* **Concepts Covered:** try, except blocks, error handling.

### 4. ****List Comprehensions****

* **Task:** Use a list comprehension to create a list of squares of numbers from 1 to 20.
* **Concepts Covered:** List comprehensions, loops.

### 5. ****Modules and Packages****

* **Task:** Write a program that uses the math module to calculate the square root of a given number.
* **Concepts Covered:** Importing modules, using built-in modules.

### 6. ****Object-Oriented Programming (OOP)****

* **Task:** Define a Car class with attributes like make, model, and year. Create instances and print their details.
* **Concepts Covered:** Classes, objects, attributes, methods.

### 7. ****Regular Expressions****

* **Task:** Write a program to validate email addresses using regular expressions.
* **Concepts Covered:** re module, pattern matching.

### 8. ****Lambda and Higher-Order Functions****

* **Task:** Use map and filter functions with lambda expressions to process a list of numbers (e.g., square even numbers).
* **Concepts Covered:** Lambda functions, map, filter.

### 9. ****Basic Data Structures****

* **Task:** Implement a stack and a queue using Python lists.
* **Concepts Covered:** Data structures, list operations.

## 📕 Advanced Level

### 1. ****Decorators****

* **Task:** Write a decorator that logs the execution time of a function.
* **Concepts Covered:** Decorators, higher-order functions, time module.

### 2. ****Generators****

* **Task:** Create a generator function that yields Fibonacci numbers up to a given limit.
* **Concepts Covered:** Generators, yield statement.

### 3. ****Multithreading****

* **Task:** Write a program that downloads multiple files concurrently using threading.
* **Concepts Covered:** threading module, concurrency.

### 4. ****Asynchronous Programming****

* **Task:** Implement an asynchronous function to fetch data from a web API using asyncio and aiohttp.
* **Concepts Covered:** Async/Await, asynchronous I/O.

### 5. ****Data Analysis with Pandas****

* **Task:** Load a CSV file using pandas, perform data cleaning, and generate summary statistics.
* **Concepts Covered:** Pandas library, data manipulation.

### 6. ****Web Development with Flask/Django****

* **Task:** Build a simple web application with user authentication.
* **Concepts Covered:** Web frameworks, routing, templates, databases.

### 7. ****Unit Testing****

* **Task:** Write unit tests for a Python module using unittest or pytest.
* **Concepts Covered:** Testing frameworks, test cases, assertions.

### 8. ****Design Patterns****

* **Task:** Implement the Singleton and Factory design patterns in Python.
* **Concepts Covered:** Design patterns, class methods, object creation.

### 9. ****Machine Learning with scikit-learn****

* **Task:** Build and evaluate a machine learning model on a dataset of your choice.
* **Concepts Covered:** Data preprocessing, model training, evaluation metrics.

### 10. ****Advanced Data Structures and Algorithms****

* **Task:** Implement a balanced binary search tree and perform insertions, deletions, and traversals.
* **Concepts Covered:** Data structures, algorithms, recursion.

### 11. ****Database Integration****

* **Task:** Create a Python application that interacts with a SQL database using sqlite3 or SQLAlchemy.
* **Concepts Covered:** SQL, database connections, CRUD operations.

### 12. ****RESTful API Development****

* **Task:** Develop a RESTful API using Flask or Django REST Framework.
* **Concepts Covered:** API design, HTTP methods, JSON handling.

## 🎯 Project Ideas

Applying your skills through projects is a great way to solidify your understanding. Here are some project ideas:

**To-Do List App**

* + **Description:** A command-line or web-based application to manage tasks.
  + **Skills Applied:** CRUD operations, user input, data storage.

**Web Scraper**

* + **Description:** Scrape data from websites and store it in a database or CSV file.
  + **Skills Applied:** requests, BeautifulSoup or Scrapy, data parsing.

**Chat Application**

* + **Description:** Build a real-time chat application using sockets.
  + **Skills Applied:** Networking, multithreading, GUI (optional).

**Blog Platform**

* + **Description:** Develop a blog site with user authentication, post creation, and comments.
  + **Skills Applied:** Web frameworks, databases, authentication.

**Game Development**

* + **Description:** Create a simple game using Pygame (e.g., Snake, Tic-Tac-Toe).
  + **Skills Applied:** Game logic, graphics, event handling.

**Expense Tracker**

* + **Description:** An application to track income and expenses with data visualization.
  + **Skills Applied:** Data storage, data analysis, plotting libraries like matplotlib or seaborn.

**Machine Learning Project**

* + **Description:** Choose a dataset (e.g., Titanic survival, Iris classification) and build a predictive model.
  + **Skills Applied:** Data preprocessing, feature engineering, model evaluation.

**Personal Portfolio Website**

* + **Description:** Develop a personal website to showcase your projects and skills.
  + **Skills Applied:** Web development, deployment.

## 📚 Additional Resources

To support your learning journey, here are some valuable resources:

* **Official Python Documentation:** [docs.python.org](https://docs.python.org/3/" \t "_new)
* **Python Exercises and Solutions:**
  + W3Resource Python Exercises
  + HackerRank Python Challenges
  + LeetCode Python Problems
* **Interactive Learning:**
  + Codecademy Python Course
  + freeCodeCamp Python Curriculum
* **Books:**
  + Automate the Boring Stuff with Python by Al Sweigart
  + Python Crash Course by Eric Matthes
  + Fluent Python by Luciano Ramalho (for advanced topics)

## 🛠️ Tips for Effective Learning

1. **Consistent Practice:** Dedicate regular time to coding to build and retain your skills.
2. **Understand the Concepts:** Don't just memorize code; strive to understand how and why it works.
3. **Work on Projects:** Apply what you learn in real-world projects to reinforce your knowledge.
4. **Read Code:** Explore open-source projects on GitHub to see how others write Python code.
5. **Seek Feedback:** Share your code with peers or mentors to receive constructive feedback.
6. **Stay Curious:** Continuously explore new libraries, frameworks, and tools in the Python ecosystem.