Complete Python Beginner Course

Welcome to your journey into programming with Python! This course is designed specifically for complete beginners. We'll take things one step at a time, using simple language and plenty of examples. You can do this — millions of people have learned Python before you, and you absolutely can too. Let's begin!

1: Introduction to Python and Setting Up Your Environment

Why This Matters

Python is one of the most popular and beginner-friendly programming languages in the world. It's used for web development, data analysis, automation, artificial intelligence, and more. Starting with a proper setup ensures you can practice smoothly without technical frustrations.

Explanation

Think of Python as a helpful assistant that follows your instructions exactly. To talk to it, you need two things:

1. Install Python on your computer.
2. Have a place to write and run your code (like a notebook for Python).

The easiest way for beginners is to use an online platform — no installation needed!

Recommended Setup (Zero Installation)

Go to https://replit.com and create a free account. Click “Create Repl” → choose “Python” → start coding instantly.
Alternative: https://www.pythonanywhere.com (also free console).

If you prefer installing locally:

- Download Python from https://www.python.org/downloads/ (choose the latest version).
- During installation, check “Add Python to PATH”.
- Open your computer's terminal/command prompt and type python --version to confirm.

Code Examples

No code yet — just open your chosen environment and type print("Hello, World!") to test it works!

Practice Exercises

No exercises for this section — just get your environment ready!

2: Print Statement and Basic Output

Why This Matters

The print() function is how your program “speaks” to you. It's the fastest way to see if your code is working and to display results.

Explanation

Imagine you want to show a message on the screen — like a sign saying “Welcome!”. In Python, you use print() and put what you want to show inside parentheses and quotes.

Code Examples

Python

```python
# Example 1: Simple message
print("Hello, World!")

# Example 2: Multiple things
print("My name is", "Alex", "and I am learning Python!")

# Example 3: Numbers
print(42)
print("The answer is", 42)

# Example 4: New lines
print("Line one\nLine two")  # \n means new line
```

Practice Exercises

- Exercise 1: Print your full name on the screen.
- Exercise 2: Print three separate lines: your favorite color, food, and hobby.
- Exercise 3: Print a small banner like this:

  text

  ```
  ***
  WELCOME TO PYTHON
  ***
  ```

Solutions

Solution to Exercise 1:

Python

```python
print("Your Name Here")  # Replace with your actual name
```

Solution to Exercise 2:

Python

```python
print("Blue")
print("Pizza")
print("Reading")
```

Solution to Exercise 3:

Python

```python
print("***")
print("WELCOME TO PYTHON")
print("***")
```

3: Variables and Data Types (int, float, str, bool)

Why This Matters

Variables are like labeled boxes where you store information. Without them, your program can't remember anything.

Explanation

A variable has a name and a value. You create one using =.
Common data types:

- int → whole numbers (e.g., 5, -10)
- float → decimal numbers (e.g., 3.14, 0.5)
- str → text (must be in quotes)
- bool → True or False only

Code Examples

Python

```python
# Example 1
name = "Anna"          # str
age = 28               # int
height = 1.65          # float
is_student = True      # bool

print(name, age, height, is_student)

# Example 2: Changing a variable
score = 0
print(score)
score = 100
print(score)

# Example 3: Type checking
print(type(name))      # <class 'str'>
print(type(age))       # <class 'int'>
```

Practice Exercises

- Exercise 1: Create variables for your name (str) and age (int), then print them.
- Exercise 2: Create variables: price = 19.99 (float), in_stock = True (bool), and product = "book" (str). Print all.
- Exercise 3: Create a variable called temperature with value 98.6, then change it to 37.0 (Celsius equivalent). Print both.

Solutions

Solution to Exercise 1:

Python

```python
name = "Your Name"
age = 25  # your age
print(name, age)
```

Solution to Exercise 2:

Python

```python
price = 19.99
in_stock = True
product = "book"
print(price, in_stock, product)
```

Solution to Exercise 3:

Python

```python
temperature = 98.6
print("Fahrenheit:", temperature)
temperature = 37.0
print("Celsius:", temperature)
```

4: Basic Operations (+, -, *, /, //, %, **)

Why This Matters

Math is everywhere in programming — calculating totals, averages, discounts, game scores, and more.

Explanation

Python can do math just like a calculator:

- + add
- - subtract
- * multiply
- / divide (gives float)
- // floor division (whole number result)
- % remainder
- ** exponent (power)

Code Examples

Python

```python
# Example 1: Basic math
print(10 + 3)    # 13
print(10 - 3)    # 7
print(10 * 3)    # 30
print(10 / 3)     # 3.333...

# Example 2: Special ones
print(10 // 3)   # 3 (whole part)
print(10 % 3)     # 1 (remainder)
print(2 ** 3)     # 8 (2 to the power 3)

# Example 3: With variables
price = 100
discount = 20
final = price - discount
print(final)
```

Practice Exercises

- Exercise 1: Calculate and print 15 + 27, then 48 - 19.
- Exercise 2: Calculate the area of a rectangle: length = 12, width = 8. Print the result.
- Exercise 3: You have 100 dollars. A game costs 29 dollars. How many games can you buy completely (//), and how much money is left (%)?

Solutions

Solution to Exercise 1:

Python

```
print(15 + 27)
print(48 - 19)
```

Solution to Exercise 2:

Python

```
length = 12
width = 8
area = length * width
print("Area:", area)
```

Solution to Exercise 3:

Python

```
money = 100
cost = 29
games = money // cost
left = money % cost
print("You can buy", games, "games")
print("Money left:", left, "dollars")
```

5: Getting User Input with input()

Why This Matters

Programs become interactive when they can ask the user for information — like names, numbers, choices.

Explanation

input() pauses the program and waits for the user to type something and press Enter. It always returns a string.

Code Examples

Python

```python
# Example 1
name = input("What is your name? ")
print("Hello,", name)

# Example 2: Getting numbers (need to convert)
age = int(input("How old are you? "))
print("Next year you will be", age + 1)

# Example 3: Multiple inputs
color = input("Favorite color? ")
food = input("Favorite food? ")
print("You like", color, "and", food)
```

Practice Exercises

- Exercise 1: Ask the user for their name and print “Welcome, [name]!”
- Exercise 2: Ask for two numbers, add them, and print the sum.
- Exercise 3: Ask for the user's name and age, then print “Hello [name], you will be [age+10] in ten years.”

Solutions

Solution to Exercise 1:

Python

```python
name = input("What is your name? ")
print("Welcome,", name + "!")
```

Solution to Exercise 2:

Python

```python
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
print("Sum:", num1 + num2)
```

Solution to Exercise 3:

Python

```python
name = input("Your name: ")
age = int(input("Your age: "))
print("Hello", name + ",", "you will be", age + 10, "in ten years.")
```

6: Strings and String Methods

Why This Matters

Text is everywhere — names, messages, emails, websites. String methods help you manipulate text easily.

Explanation

Strings are text in quotes. Methods are built-in tools you can use on them with dot notation.

Code Examples

Python

```python
# Example 1: Common methods
message = "  hello world  "
print(message.upper())        # HELLO WORLD
print(message.lower())        # hello world
print(message.strip())        # hello world (removes spaces)
print(message.replace("world", "Python"))

# Example 2: Length and indexing
text = "Python"
print(len(text))              # 6
print(text[0])                # P (first letter)
print(text[-1])               # n (last letter)

# Example 3: Concatenation and f-strings
name = "Sam"
age = 12
print(f"{name} is {age} years old.")  # modern way
```

Practice Exercises

- Exercise 1: Take a string "python is fun" and print it in all uppercase.
- Exercise 2: Ask the user for their full name, then print it stripped and title-cased (First Letters Capital).
- Exercise 3: Ask for a sentence, then print its length and the first and last characters.

Solutions

Solution to Exercise 1:

Python

```python
text = "python is fun"
print(text.upper())
```

Solution to Exercise 2:

Python

```
full_name = input("Enter your full name: ")
print(full_name.strip().title())
```

Solution to Exercise 3:

Python

```
sentence = input("Enter a sentence: ")
print("Length:", len(sentence))
print("First:", sentence[0])
print("Last:", sentence[-1])
```

7: Conditionals (if, elif, else)

Why This Matters

Programs need to make decisions — “if it's raining, take umbrella; else, wear sunglasses.”

Explanation

if checks a condition. If true → run that block. elif checks another. else runs if nothing else is true. Use : and indentation.

Code Examples

Python

```
# Example 1: Simple if
age = 18
if age >= 18:
    print("You can vote!")

# Example 2: if-else
temperature = 15
if temperature > 25:
    print("It's hot!")
else:
    print("It's cool.")

# Example 3: if-elif-else
score = 85
if score >= 90:
    print("A")
elif score >= 80:
    print("B")
elif score >= 70:
    print("C")
else:
    print("Try harder")
```

Practice Exercises

- Exercise 1: Ask for a number. If it's even, print “Even”; else print “Odd”.
- Exercise 2: Ask for age. If under 13 → “Child”, 13-19 → “Teen”, else → “Adult”.
- Exercise 3: Ask for a password. If it's exactly “python123”, print “Access granted”; else “Access denied”.

Solutions

Solution to Exercise 1:

Python

```python
num = int(input("Enter a number: "))
if num % 2 == 0:
    print("Even")
else:
    print("Odd")
```

Solution to Exercise 2:

Python

```python
age = int(input("Enter your age: "))
if age < 13:
    print("Child")
elif age <= 19:
    print("Teen")
else:
    print("Adult")
```

Solution to Exercise 3:

Python

```python
password = input("Enter password: ")
if password == "python123":
    print("Access granted")
else:
    print("Access denied")
```

8: Loops – while loop

Why This Matters

Loops let you repeat tasks without writing the same code over and over — perfect for games, counters, menus.

Explanation

A while loop keeps running as long as its condition is True. Be careful — it can run forever if the condition never becomes False!

Code Examples

Python

```python
# Example 1: Countdown
count = 5
while count > 0:
    print(count)
    count = count - 1
print("Blast off!")

# Example 2: Keep asking until correct
password = ""
while password != "secret":
    password = input("Enter password: ")
print("Welcome!")

# Example 3: Simple counter
total = 0
num = 1
while num <= 10:
    total += num
    num += 1
print("Sum 1 to 10:", total)
```

Practice Exercises

- Exercise 1: Print numbers from 1 to 10 using while.
- Exercise 2: Keep asking for a number until the user enters a negative number, then stop.
- Exercise 3: Create a simple multiplication game: ask random questions (e.g., 4×7) until the user gets one wrong.

Solutions

Solution to Exercise 1:

Python

```python
i = 1
while i <= 10:
    print(i)
    i += 1
```

Solution to Exercise 2:

Python

```python
num = 0
while num >= 0:
    num = int(input("Enter a number (negative to stop): "))
```

```
print("Stopped.")
```

Solution to Exercise 3:

Python

```
import random
correct = True
while correct:
    a = random.randint(1, 10)
    b = random.randint(1, 10)
    answer = int(input(f"What is {a} x {b}? "))
    if answer == a * b:
        print("Correct!")
    else:
        print("Wrong! Game over.")
        correct = False
```

9: Loops – for loop and range()

Why This Matters

for loops are perfect when you know exactly how many times you want to repeat something.

Explanation

for loops work great with range(start, stop, step). range(10) gives 0 to 9.

Code Examples

Python

```
# Example 1: Basic for
for i in range(5):
    print("Hello", i)

# Example 2: Custom range
for num in range(1, 11):     # 1 to 10
    print(num)

# Example 3: Step
for even in range(2, 11, 2):  # 2,4,6,8,10
    print(even)

# Example 4: Over a string
name = "Python"
for letter in name:
    print(letter)
```

Practice Exercises

- Exercise 1: Print all even numbers from 0 to 20.
- Exercise 2: Print a multiplication table for 5 (5×1=5 up to 5×10=50).
- Exercise 3: Print a triangle of stars:

text

```
*
**
***
****
*****
```

Solutions

Solution to Exercise 1:

Python

```python
for i in range(0, 21, 2):
    print(i)
```

Solution to Exercise 2:

Python

```python
for i in range(1, 11):
    print(f"5 x {i} = {5 * i}")
```

Solution to Exercise 3:

Python

```python
for i in range(1, 6):
    print("*" * i)
```

10: Lists

Why This Matters

Lists store multiple items in order — like a shopping list or top 10 songs.

Explanation

Lists are created with square brackets [ ]. You can add, remove, change items.

Code Examples

Python

```python
# Example 1: Creating and accessing
fruits = ["apple", "banana", "orange"]
print(fruits[0])      # apple
print(fruits[-1])     # orange

# Example 2: Modifying
fruits.append("grape")
```

```
fruits[1] = "kiwi"
print(fruits)

# Example 3: Looping
for fruit in fruits:
    print("I like", fruit)

# Example 4: Length and slicing
print(len(fruits))
print(fruits[1:3])    # ['kiwi', 'orange']
```

Practice Exercises

- Exercise 1: Create a list with 5 colors, print the whole list and the third color.
- Exercise 2: Start with empty list, ask user for 3 favorite movies, append them, then print the list.
- Exercise 3: Create a list of numbers 1–10, then print only the even ones using a for loop.

Solutions

Solution to Exercise 1:

Python

```
colors = ["red", "blue", "green", "yellow", "purple"]
print(colors)
print(colors[2])  # green (index 2)
```

Solution to Exercise 2:

Python

```
movies = []
movies.append(input("Movie 1: "))
movies.append(input("Movie 2: "))
movies.append(input("Movie 3: "))
print("Your favorites:", movies)
```

Solution to Exercise 3:

Python

```
numbers = [1,2,3,4,5,6,7,8,9,10]
for n in numbers:
    if n % 2 == 0:
        print(n)
```

11: Tuples and Sets

Why This Matters

Tuples are like lists but can't be changed — great for fixed data. Sets automatically remove duplicates and are fast for membership testing.

Explanation

- Tuple: ( ) — immutable
- Set: { } — unordered, no duplicates

Code Examples

Python

```python
# Tuple
coordinates = (10, 20)
print(coordinates[0])

# Set
unique = {1, 2, 2, 3, 3}  # duplicates removed
print(unique)            # {1, 2, 3}
unique.add(4)
print(unique)

# Membership
print(3 in unique)       # True
```

Practice Exercises

- Exercise 1: Create a tuple with days of the week, print the 4th day.
- Exercise 2: Create a set from list [1,2,2,3,3,4], print the set.
- Exercise 3: Create two sets: {1,2,3,4} and {3,4,5,6}. Print their intersection.

Solutions

Solution to Exercise 1:

Python

```python
days = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
print(days[3])  # Thursday (index 3)
```

Solution to Exercise 2:

Python

```python
nums = [1,2,2,3,3,4]
unique = set(nums)
print(unique)
```

Solution to Exercise 3:

Python

```
set1 = {1,2,3,4}
set2 = {3,4,5,6}
print(set1.intersection(set2))  # or set1 & set2
```

12: Dictionaries

Why This Matters

Dictionaries store data as key-value pairs — like a real dictionary: word → definition.

Explanation

Created with curly braces and colon: {"key": value}

Code Examples

Python

```
# Example 1
person = {"name": "Emma", "age": 22, "city": "Paris"}
print(person["name"])

# Example 2: Modify and add
person["age"] = 23
person["job"] = "Designer"
print(person)

# Example 3: Looping
for key, value in person.items():
    print(key, ":", value)
```

Practice Exercises

- Exercise 1: Create a dictionary for a student with name, grade, subject. Print the grade.
- Exercise 2: Create an empty dictionary, ask user for 3 key-value pairs (e.g., fruit-color), add them.
- Exercise 3: Create a phonebook dictionary with 3 names and numbers, then look up one name.

Solutions

Solution to Exercise 1:

Python

```
student = {"name": "Tom", "grade": 95, "subject": "Math"}
print("Grade:", student["grade"])
```

Solution to Exercise 2:

Python

```
book = {}
```

```
for i in range(3):
    key = input("Key: ")
    value = input("Value: ")
    book[key] = value
print(book)
```

Solution to Exercise 3:

Python

```
phonebook = {"Alice": "123-456", "Bob": "789-000", "Cara": "111-222"}
name = input("Who to look up? ")
if name in phonebook:
    print(phonebook[name])
else:
    print("Not found")
```

13: Functions – Defining and Calling

Why This Matters

Functions let you reuse code and organize your program into logical blocks.

Explanation

Define with def, call by name. Like a recipe you can use many times.

Code Examples

Python

```
# Example 1
def greet():
    print("Hello!")

greet()          # call it
greet()

# Example 2
def say_hello(name):
    print("Hello,", name)

say_hello("Mia")
say_hello("Leo")
```

Practice Exercises

- Exercise 1: Define a function welcome() that prints “Welcome to Python!”. Call it twice.
- Exercise 2: Define add_numbers() that prints the sum of 5 and 10. Call it.
- Exercise 3: Define print_square(num) that prints the square of the given number. Call it with 7.

Solutions

Solution to Exercise 1:

Python

```python
def welcome():
    print("Welcome to Python!")

welcome()
welcome()
```

Solution to Exercise 2:

Python

```python
def add_numbers():
    print(5 + 10)

add_numbers()
```

Solution to Exercise 3:

Python

```python
def print_square(num):
    print(num ** 2)

print_square(7)
```

14: Functions – Parameters and Return

Why This Matters

return lets a function give back a result that you can store or use later.

Explanation

Parameters go in parentheses when defining. return sends back a value.

Code Examples

Python

```python
# Example 1
def add(a, b):
    return a + b

result = add(10, 20)
print(result)

# Example 2
def is_even(num):
    return num % 2 == 0
```

```
print(is_even(4))   # True
```

Practice Exercises

- Exercise 1: Write multiply(x, y) that returns the product. Use it to print 6×7.
- Exercise 2: Write get_full_name(first, last) that returns the full name as one string.
- Exercise 3: Write max_of_three(a, b, c) that returns the largest number.

Solutions

Solution to Exercise 1:

Python

```
def multiply(x, y):
    return x * y

print(multiply(6, 7))
```

Solution to Exercise 2:

Python

```
def get_full_name(first, last):
    return first + " " + last

print(get_full_name("Ada", "Lovelace"))
```

Solution to Exercise 3:

Python

```
def max_of_three(a, b, c):
    if a >= b and a >= c:
        return a
    elif b >= a and b >= c:
        return b
    else:
        return c

print(max_of_three(10, 25, 15))
```

15: Modules and Importing

Why This Matters

Modules are pre-written code you can use — no need to reinvent the wheel.

Explanation

import brings in a module. Popular ones: random, math, datetime.

Code Examples

Python

```python
# Example 1: random
import random

print(random.randint(1, 100))
print(random.choice(["apple", "banana", "orange"]))

# Example 2: math
import math

print(math.sqrt(16))
print(math.pi)

# Example 3: from syntax
from datetime import datetime
print(datetime.now())
```

Practice Exercises

- Exercise 1: Import random and print a random number between 1 and 10 five times.
- Exercise 2: Import math, calculate and print the area of a circle with radius 5.
- Exercise 3: Use random to make a simple coin flip: print “Heads” or “Tails”.

Solutions

Solution to Exercise 1:

Python

```python
import random
for i in range(5):
    print(random.randint(1, 10))
```

Solution to Exercise 2:

Python

```python
import math
radius = 5
area = math.pi * radius ** 2
print("Area:", area)
```

Solution to Exercise 3:

Python

```python
import random
if random.randint(0, 1) == 0:
    print("Heads")
```

```
else:
    print("Tails")
```

16: Basic File Handling (reading and writing text files)

Why This Matters

Programs often need to save data permanently — scores, notes, logs.

Explanation

Use open() with mode 'w' to write, 'r' to read. Always close the file or use with.

Code Examples

Python

```python
# Example 1: Writing
with open("note.txt", "w") as file:
    file.write("Hello file!\n")
    file.write("This is line 2")

# Example 2: Reading
with open("note.txt", "r") as file:
    content = file.read()
    print(content)

# Example 3: Append
with open("note.txt", "a") as file:
    file.write("\nNew line added")
```

Practice Exercises

- Exercise 1: Write your name and age to a file called “info.txt”.
- Exercise 2: Read and print the content of “info.txt”.
- Exercise 3: Create a to-do list: ask user for 3 tasks, write each on a new line in “todo.txt”.

Solutions

Solution to Exercise 1:

Python

```python
with open("info.txt", "w") as f:
    f.write("Name: Alice\n")
    f.write("Age: 30")
```

Solution to Exercise 2:

Python

```python
with open("info.txt", "r") as f:
    print(f.read())
```

Solution to Exercise 3:

Python

```
with open("todo.txt", "w") as f:
    for i in range(3):
        task = input(f"Task {i+1}: ")
        f.write(task + "\n")
```

17: Error Handling with try-except

Why This Matters

Programs can crash when something unexpected happens (wrong input, file not found). try-except prevents crashes.

Explanation

Put risky code in try. If error occurs, run except instead of crashing.

Code Examples

Python

```
# Example 1
try:
    num = int(input("Enter a number: "))
    print(100 / num)
except:
    print("Something went wrong!")

# Example 2: Specific errors
try:
    age = int(input("Age: "))
except ValueError:
    print("Please enter a valid number")
```

Practice Exercises

- Exercise 1: Ask for a number and divide 100 by it, with basic try-except.
- Exercise 2: Ask for two numbers, divide first by second with try-except for division by zero.
- Exercise 3: Try to open a file “missing.txt”. If not found, print friendly message.

Solutions

Solution to Exercise 1:

Python

```
try:
    num = int(input("Number: "))
    print(100 / num)
except:
    print("Invalid input or division by zero")
```

Solution to Exercise 2:

Python

```
try:
    a = int(input("First: "))
    b = int(input("Second: "))
    print(a / b)
except ZeroDivisionError:
    print("Cannot divide by zero")
except ValueError:
    print("Enter numbers only")
```

Solution to Exercise 3:

Python

```
try:
    with open("missing.txt", "r") as f:
        print(f.read())
except FileNotFoundError:
    print("File not found – creating a new one.")
    with open("missing.txt", "w") as f:
        f.write("Hello")
```

18: Introduction to Simple Projects

Why This Matters

Now you have all the basic tools — time to combine them into real programs!

Explanation

Here are three classic beginner projects you can build right now using what you've learned.

Project Ideas

1. **Number Guessing Game**
    - Computer picks random number 1–100
    - User guesses, program says too high/low
    - Count attempts
2. **Simple To-Do List**
    - Menu: add task, view tasks, remove task, quit
    - Save tasks to a file
3. **Quiz Game**
    - Ask 5 multiple-choice questions
    - Keep score
    - Show final result

Code Example – Number Guessing Game

Python

```
import random

secret = random.randint(1, 100)
guess = 0
attempts = 0

print("Guess the number between 1 and 100!")

while guess != secret:
    guess = int(input("Your guess: "))
    attempts += 1
    if guess < secret:
        print("Too low!")
    elif guess > secret:
        print("Too high!")
    else:
        print(f"Correct! You got it in {attempts} attempts!")
```

Practice
Choose one project above and build it step by step. Break it into small pieces, test often, and celebrate when it works!

Final Words

Congratulations on completing this Python beginner course! You've just learned the core building blocks that professional programmers use every day. Remember: every expert was once a beginner. The secret to mastery is consistent practice.

Keep coding a little every day. Build small projects that interest you — a calculator, a journal app, a game. When you get stuck, read error messages, search online, and ask questions. You now have the foundation to create almost anything with Python.

You're capable of amazing things. Keep going — the coding world is waiting for you!

Happy coding!
— Your Python Instructor