PROGRAMMING ON PYTHON

By
Kulakov Denis

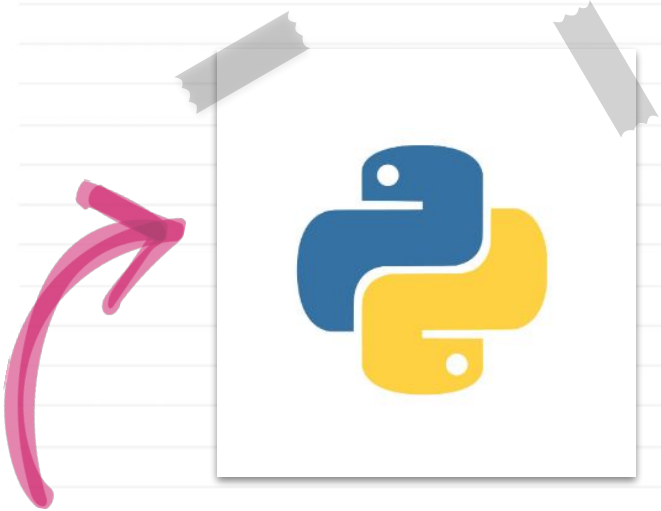
Introduction

Overview of Python:

- High-level, interpreted language
- Created by Guido van Rossum
- First released in 1991

Key Characteristics:

- Easy to read and write
- Versatile and widely used
- Supports multiple programming paradigms



Applications of Python

Real-world Applications:

- Web development (Django, Flask)
- Data science (Pandas, NumPy, Matplotlib)
- Automation (scripting, task automation)
- Artificial intelligence (TensorFlow, scikit-learn)

Companies Using Python:

Google Яндекс Microsoft NVIDIA

And many more!

Basic Python Concepts

```python
age = 21 # int
temperature = 98.6 # float
name = "Alice" # str
is_student = True # bool

# Basic Operations

addition = a + b
subtraction = a - b
multiplication = a * b
division = a / b

# Comparison Operations
equal = (a == b) # False
not_equal = (a != b) # True
greater = (a > b) # True
less = (a < b) # False
```

Simple Calculator on Python

```python
operation = input("Choose operation (+, -, *, /): ")
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

if operation == '+':
    print("Result:", num1 + num2)
elif operation == '-':
    print("Result:", num1 - num2)
elif operation == '*':
    print("Result:", num1 * num2)
elif operation == '/':
    if num2 != 0:
        print("Result:", num1 / num2)
    else:
        print("Error: Division by zero")
else:
    print("Invalid operation")
```

Data Structures in Python

```python
# List of student names
students = ["Alice", "Bob", "Charlie", "Diana"]

# Add a new student to the list
students.append("Eve")
# Output: List of students: ['Alice', 'Bob', 'Charlie',
'Diana', 'Eve']

# Tuple of coordinates (immutable)
coordinates = (10.0, 20.0)
# Output: Coordinates: (10.0, 20.0)

# Dictionary of student grades
grades = {
    "Alice": 85,
    "Bob": 90,
    "Charlie": 78,
    "Diana": 92
}
```

```python
# Add a new student's grade
grades["Eve"] = 88
# Expected output: Student grades: {'Alice': 85,
'Bob': 90, 'Charlie': 78, 'Diana': 92, 'Eve': 88}

# Update a student's grade
grades["Alice"] = 95
# Expected output: Updated grades: {'Alice': 95,
'Bob': 90, 'Charlie': 78, 'Diana': 92, 'Eve': 88}

# Set of unique student IDs
student_ids = {101, 102, 103, 104}

# Add a new ID
student_ids.add(105)
# Output: Student IDs: {101, 102, 103, 104, 105}

# Try to add a duplicate ID
student_ids.add(101)
# Output: Student IDs after attempting to add
duplicate: {101, 102, 103, 104, 105}
```

Functions and Modules

Functions are defined using the **def** keyword, allowing you to create reusable code blocks. They can take parameters as inputs and return values.

Modules are collections of related functions and variables. Python's standard library includes modules like **math** for mathematical operations and **random** for generating random numbers.

```
numbers = [10, 20, 30, 40, 50]
total = sum(numbers)
count = len(numbers)
average = total / count
print(f"The average is: {average}")
# Output: The average is: 30.0
```

```
import math

number = 16
square_root = math.sqrt(number)
print(f"The square root of {number} is:
{square_root}")
# Output: The square root of 16 is: 4.0

# Importing the random module and generating
a random number
import random

random_number = random.randint(1, 100)
print(f"Random number between 1 and 100:
{random_number}")
# Output: Random number between 1 and 100:
<random_number>
```

Error Handling

Error handling allows programs to manage unexpected situations gracefully.

The **try** block lets Python attempt to execute code, and if an error occurs

The **except** block handles it.

- For instance, catching a division by zero error prevents the program from crashing and allows displaying a user-friendly message.

The **finally** block ensures certain code runs regardless of an exception

- Such as closing a file or releasing resources.

```python
def divide_numbers(a, b):
    try:
        result = a / b
    except ZeroDivisionError:
        return "Error: Cannot divide by zero."
    finally:
        print("Execution completed.")
    return result

num1 = 10
num2 = 0
print(divide_numbers(num1, num2))
# Output: Error: Cannot divide by zero.
```

Exercises

These hands-on tasks will help you apply the concepts we've discussed and give you a better understanding of how to use Python in real-world scenarios.

1. Temperature Converter:

Write a program to convert temperatures between Fahrenheit and Celsius. Create functions for the conversions and handle user input for the temperatures.

2. Contact Book:

Develop a program to manage a contact book. Allow users to add, search, update, and delete contacts, with each contact containing a name, phone number, and email address.

3. Number Guessing Game:

Implement a game where the user guesses a randomly selected number within a range. Provide feedback for each guess and track the number of attempts.

Self Check.
Temperature Converter

```python
def celsius_to_fahrenheit(celsius):
    return (celsius * 9/5) + 32

def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5/9

def main():
    print("Temperature Converter")
    choice = input("Enter 1 to convert Celsius to Fahrenheit or 2 to convert Fahrenheit to Celsius: ")

    if choice == '1':
        celsius = float(input("Enter temperature in Celsius: "))
        print(f"{celsius}°C is {celsius_to_fahrenheit(celsius):.2f}°F")
    elif choice == '2':
        fahrenheit = float(input("Enter temperature in Fahrenheit: "))
        print(f"{fahrenheit}°F is {fahrenheit_to_celsius(fahrenheit):.2f}°C")
    else:
        print("Invalid choice. Please enter 1 or 2.")

if __name__ == "__main__":
    main()
```

Self Check. Contact Book

```python
contacts = {}
def add_contact(name, phone, email):
    contacts[name] = {"phone": phone, "email": email}
    print(f"Contact {name} added successfully.")
def search_contact(name):
    contact = contacts.get(name)
    if contact:
        print(f"Name: {name}, Phone: {contact['phone']}, Email:
{contact['email']}")
    else:
        print(f"Contact {name} not found.")
def update_contact(name, phone=None, email=None):
    if name in contacts:
        if phone:
            contacts[name]['phone'] = phone
        if email:
            contacts[name]['email'] = email
        print(f"Contact {name} updated successfully.")
    else:
        print(f"Contact {name} not found.")
def delete_contact(name):
    if name in contacts:
        del contacts[name]
        print(f"Contact {name} deleted successfully.")
    else:
        print(f"Contact {name} not found.")
```

```python
def main():
    while True:
        print("\nContact Book Menu")
        print("1. Add Contact")
        print("2. Search Contact")
        print("3. Update Contact")
        print("4. Delete Contact")
        print("5. Exit")

        choice = input("Choose an option (1-5): ")

        if choice == '1':
            name = input("Enter name: ")
            phone = input("Enter phone number: ")
            email = input("Enter email address: ")
            add_contact(name, phone, email)
        elif choice == '2':
            name = input("Enter name to search: ")
            search_contact(name)
        elif choice == '3':
            name = input("Enter name to update: ")
            phone = input("Enter new phone number (leave blank to keep current): ")
            email = input("Enter new email address (leave blank to keep current): ")
            update_contact(name, phone if phone else None, email if email else None)
        elif choice == '4':
            name = input("Enter name to delete: ")
            delete_contact(name)
        elif choice == '5':
            print("Exiting Contact Book.")
            break
        else:
            print("Invalid choice. Please choose a valid option.")

if __name__ == "__main__":
    main()
```

Self Check. Number Guessing Game

```python
import random

def number_guessing_game():
    lower_bound = int(input("Enter the lower bound of the range: "))
    upper_bound = int(input("Enter the upper bound of the range: "))

    # Ensure that the upper bound is greater than the lower bound
    while upper_bound <= lower_bound:
        print("Upper bound must be greater than lower bound. Please try again.")
        upper_bound = int(input("Enter the upper bound of the range: "))

    secret_number = random.randint(lower_bound, upper_bound)
    attempts = 0

    print(f"Guess the number between {lower_bound} and {upper_bound}")

    while True:
        guess = int(input("Enter your guess: "))
        attempts += 1

        if guess < secret_number:
            print("Too low! Try again.")
        elif guess > secret_number:
            print("Too high! Try again.")
        else:
            print(f"Congratulations! You guessed the number in {attempts} attempts.")
            break

if __name__ == "__main__":
    number_guessing_game()
```

Brief summary of what we have discussed today

Python is one of the most readable and writable programming languages

Python can be used for an incredibly large set of tasks ranging from AI to web applications

Python is an <u>object-oriented</u> and <u>functional</u> language

Python can handle errors and exceptions

And we also had some practice!

Q&A

Thank you for your attention!

Find us online:

Tg: @fcs_hse
Vk: @cshse
cs.hse.ru