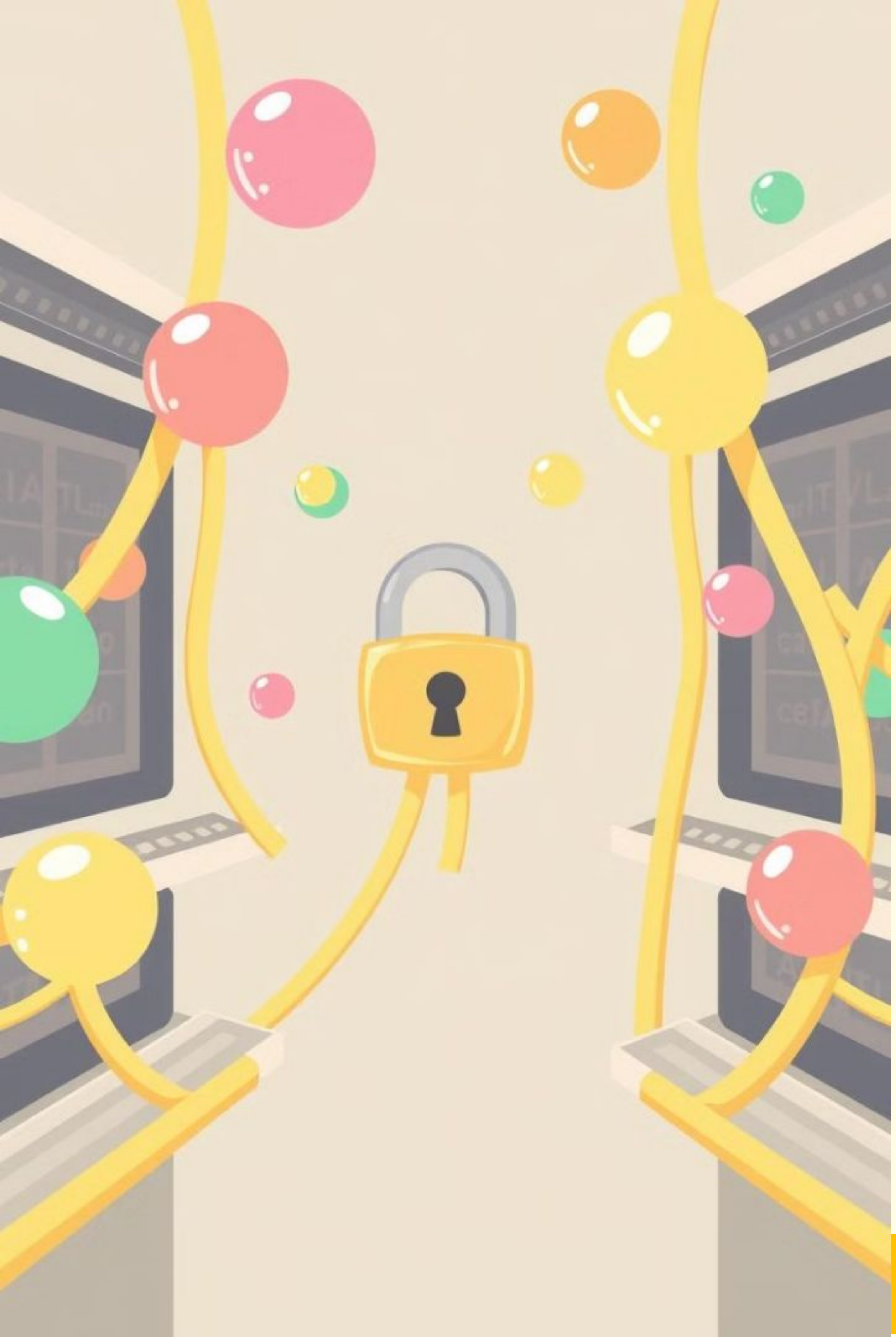


OTP System: A Python-Based Implementation

Presented by
Twinkle Devwanshi
Student id- S9104

Project Overview: Purpose and Features

1. **OTP Generation**

   The system generates random and unique OTPs for each user request, ensuring security and preventing reuse.

2. **Email Delivery**

   OTPs are securely sent to the user's email address, providing a convenient and reliable delivery method.

3. **OTP Verification**

   The system verifies the entered OTP against the generated one, ensuring that the user is the legitimate owner of the account.

Libraries Used

- `smtplib`: For sending emails.
- `random`: For generating random OTPs.
- `re`: For validating email format using regex.
- `tkinter`: For creating the GUI.
- `email.mime`: For formatting email content.

OTP Generation

```python
def generate_otp(length=6):
    """
    Generates a random OTP of the specified length.
    """
    return ''.join(random.choices('0123456789', k=length))
```

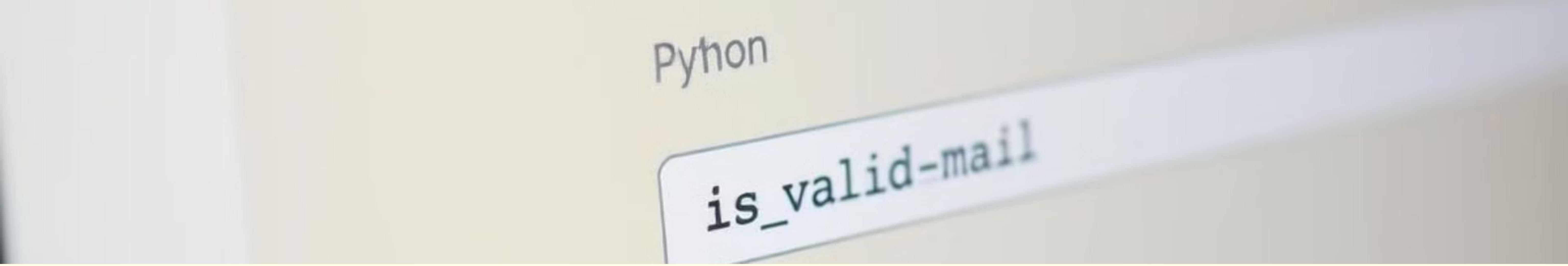
Email Sending

```
def send_otp_email(receiver_email, otp):

    # Email credentials and settings
    sender_email = "your_email@example.com"
    sender_password = "your_password"
    subject = "Your OTP Code"

    # Email content
    message = MIMEMultipart()
    message['From'] = sender_email
    message['To'] = receiver_email
    message['Subject'] = subject
    body = f"Your One-Time Password (OTP) is: {otp}\n\nPlease use this 
code to complete your action."
    message.attach(MIMEText(body, 'plain'))

    # Sending the email via SMTP
    try:
        with smtplib.SMTP('smtp.gmail.com', 587) as server:
            server.starttls()
            server.login(sender_email, sender_password)
            server.sendmail(sender_email, receiver_email, 
message.as_string())
            print(f"OTP sent to {receiver_email}")
    except Exception as e:
        print(f"Failed to send OTP. Error: {e}")
```



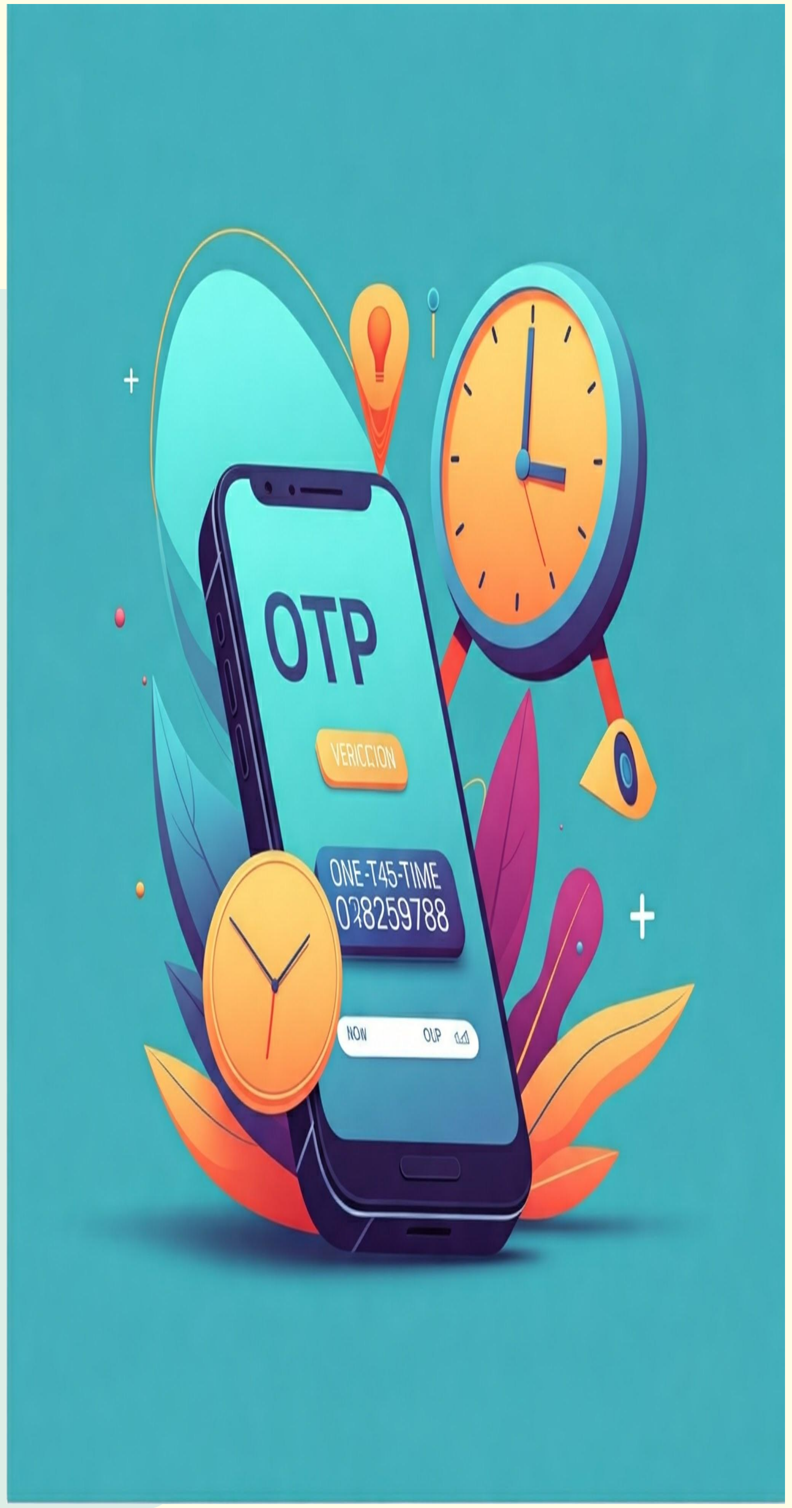
Email Validation

```
def is_valid_email(email):
    """
    Validates the email address format.
    """
    email_regex = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return re.match(email_regex, email) is not None
```

GUI Functions : Send OTP

```
def send_otp():
    email = email_entry.get()
    if not is_valid_email(email):
        messagebox.showerror("Error", "Invalid email address.")
        return

    otp = generate_otp()
    otp_store[email] = otp
    send_otp_email(email, otp)
    messagebox.showinfo("Success", "OTP sent successfully!")
```



GUI Functions: Verify OTP

```python
def verify_otp():
    email = email_entry.get()
    otp = otp_entry.get()

    if email not in otp_store:
        messagebox.showerror("Error", "No OTP generated for this email.")
        return

    if otp_store[email] == otp:
        del otp_store[email]  # Clear OTP after successful verification
        messagebox.showinfo("Success", "OTP verified successfully!")
    else:
        messagebox.showerror("Error", "Invalid OTP.")
```

How it Works: The OTP Verification Process

1. User enters their email address in the application.
2. The system generates a unique OTP and sends it to the user's email.
3. User receives the OTP via email and enters it into the application.
4. The system verifies the entered OTP against the generated one.
5. If the OTP matches, access is granted to the account or service.

Key Takeaways and Next Steps

OTP Benefits

OTPs provide a significant improvement in security and are essential for various applications.

Python Implementation

This project demonstrates a foundational understanding of OTP implementation using Python.

Future Enhancements

Consider database integration, SMS delivery, and more sophisticated GUI designs for future enhancements.

Thank You