# Birthday Notification Script Documentation

## Overview

This Python script automates the process of sending birthday greetings to employees by fetching employee birthday data from an Oracle database and sending customized birthday emails using SMTP. It randomly selects a background color for the email and includes HTML-based birthday cards for each employee.

## Libraries Used

The following Python libraries are used in the script:

* oracledb: For interacting with the Oracle database.
* smtplib: For sending emails via SMTP.
* email.mime.multipart and email.mime.text: For constructing and sending MIME emails.
* datetime: For handling date-related operations (e.g., retrieving today's date).
* random: For randomly selecting a background color from a predefined list.

## Classes and Methods

### 1. Custom Exception: DatabaseConnectionError

A custom exception for handling errors related to the database connection.

#### Usage:

* Raised when there is an issue connecting to the Oracle database.

### 2. Class: BirthdayNotifier

The main class responsible for managing the birthday notification process.

#### Initialization: \_\_init\_\_(self, db\_config, email\_config)

Parameters:

* db\_config: A dictionary containing the database connection details such as username, password, host, port, and service name.
* email\_config: A dictionary containing the email configuration details such as sender’s email, recipient’s email, SMTP server, port, and sender’s password.

Purpose: Initializes the BirthdayNotifier object with database and email configuration.

#### Method: connect\_to\_db(self)

* Purpose: Establishes a connection to the Oracle database.
* Returns: None
* Raises: DatabaseConnectionError if the connection fails.

#### Method: get\_birthday\_data(self)

* Purpose: Retrieves a list of employees whose birthdays fall on the current date from the database.
* Returns: A list of employee birthday records (tuples) containing NAME, DEPARTMENT, DESIGNATION, and DOB.
* Raises: DatabaseConnectionError if there is an issue during the query execution.

#### Method: get\_random\_dark\_background\_color(self)

* Purpose: Returns a randomly selected dark color hex code from a predefined list.
* Returns: A string representing a dark hex color code.

#### Method: send\_birthday\_email(self, birthday\_data)

Parameters:

* birthday\_data: A list of tuples containing employee details for those with birthdays today.
* Purpose: Constructs and sends a birthday greeting email with a visually appealing layout.
* Returns: None
* Side Effect: Sends an email to the configured recipient(s).

#### Method: generate\_birthday\_cards(self, birthday\_data)

Parameters:

* birthday\_data: A list of tuples containing employee details for those with birthdays today.
* Purpose: Generates HTML code for birthday cards that include the employee’s name, department, and designation. Cards also have a "Wish Birthday!" button that opens a mail client with a prefilled birthday message.
* Returns: A string of HTML content representing the birthday cards.

#### Method: send\_email(self, subject, body)

Parameters:

* subject: The subject of the email.
* body: The HTML body of the email.
* Purpose: Sends an email using the SMTP protocol.
* Returns: None
* Side Effect: Sends an email to the recipient(s).

#### Method: close\_connection(self)

* Purpose: Closes the connection to the Oracle database when all operations are complete.
* Returns: None
* Side Effect: Closes the database connection.

### Main Execution Block

Purpose: The script will connect to the database, fetch the birthday data, send the birthday email, and close the connection.

Flow:

1. The script initializes the BirthdayNotifier object with the given database and email configuration.
2. It then attempts to connect to the database and fetch birthday data for today.
3. If birthdays are found, an email is composed and sent to the recipient(s).
4. Finally, the database connection is closed.

## Configuration

### Database Configuration (db\_config)

This dictionary contains the credentials needed to connect to the Oracle database.

python

Copy code

db\_config = {

'user': 'jnl',

'password': 'jnl123',

'host' : '80.0.1.81',

'port': 1521,

'service\_name' : "jnil"

}

### Email Configuration (email\_config)

This dictionary contains the configuration for sending emails.

python

Copy code

email\_config = {

'from': "myneco@necoindia.com",

'to': "saket.verma@necoindia.com",

# 'bcc': "saket.verma@necoindia.com",

'smtp\_server': 'smtp.gmail.com',

'smtp\_port': 587,

'password': "ukzu matf aupi kesv"

}

## Error Handling

The script raises a custom DatabaseConnectionError if there is any issue connecting to the database or executing the query to retrieve birthday data. This is handled by the try/except block in the \_\_main\_\_ execution section.

## Logging

* The script prints messages to the console at various stages (connecting to the database, sending emails, etc.) to help with debugging and to track progress.

## Usage Instructions

### 1. Setup

Before running the script, you need to set up the following configurations:

1. Database Configuration: Replace the values in db\_config with the correct details for your Oracle database.
2. Email Configuration: Replace the values in email\_config with your sender and recipient email details. Ensure the SMTP server and port are correctly configured.
3. Oracle Database: The script assumes that the database contains a view or table V\_EMPDATA with the following columns: NAME, DEPARTMENT, DESIGNATION, and DOB.

### 2. Install Dependencies

Make sure the required libraries are installed:

bash

Copy code

pip install oracledb

pip install smtplib

### 3. Running the Script

Once the configuration is complete, run the script. The script will:

1. Connect to the Oracle database.
2. Fetch birthday data for today.
3. Send a personalized birthday email to the specified recipient.

### 4. Email Output

The email will include:

* A visually appealing design with a randomly selected background color.
* A card for each employee whose birthday is today.
* A “Wish Birthday!” button that links to a prefilled email, allowing others to send their own birthday wishes.

## Example of Output Email

The email will look like this:

* A header saying "🎉 Happy Birthday! 🎉"
* A section listing the employees with birthdays today, with each employee having a personalized card displaying:
  + Name
  + Department and Designation
  + A "Wish Birthday!" button that links to a prefilled email

The background color of the email will be randomly selected from a set of dark tones, providing an elegant and cheerful visual effect.

## Future Enhancements

* Multiple Recipients: You could expand the email functionality to send the email to multiple recipients by modifying the to field in the email\_config.
* Database Query Optimization: If the database has many records, consider optimizing the query with better indexing or pagination.
* Customizable Card Layout: Allow more flexibility in the design and layout of birthday cards (e.g., adding images, more styling options).

## Conclusion

This script automates the process of wishing employees a happy birthday and sending them personalized greetings, making it a useful tool for maintaining employee morale and engagement. The combination of database querying, dynamic email generation, and a visually appealing design makes this a powerful and flexible solution.

**CODE:**

import oracledb

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from datetime import datetime ,timedelta

import random

class DatabaseConnectionError(Exception):

"""Custom exception for database connection errors."""

pass

class BirthdayNotifier:

def \_\_init\_\_(self, db\_config, email\_config):

self.db\_config = db\_config

self.email\_config = email\_config

self.connection = None

def connect\_to\_db(self):

try:

print("Connecting to the database...")

self.connection = oracledb.connect(

user=self.db\_config['user'],

password=self.db\_config['password'],

host = self.db\_config['host'],

port = self.db\_config["port"],

service\_name = self.db\_config["service\_name"]

)

print("Connection successful.")

except oracledb.DatabaseError as e:

error, = e.args

print(f"Error code: {error.code}, Error message: {error.message}")

raise DatabaseConnectionError(f"Error connecting to the database: {e}")

def get\_birthday\_data(self):

cursor = None

try:

cursor = self.connection.cursor()

current\_date = datetime.now()

# Use timedelta to adjust the date (e.g., adding 5 days to the current date)

adjusted\_date = current\_date - timedelta(days=0)

# Format the adjusted date as 'DD-MMM' (e.g., '09-NOV')

formatted\_date = adjusted\_date.strftime('%d-%b').upper()

day, month = formatted\_date.split('-')

# day= str(int(day)+1)

# print(day,type(day))

query = """

SELECT NAME, DEPARTMENT, DESIGNATION, DOB

FROM V\_EMPDATA

WHERE TO\_CHAR(DOB, 'MON') = :month AND TO\_CHAR(DOB, 'DD') = :day

"""

cursor.execute(query, {'month': month, 'day': day})

results = cursor.fetchall()

if not results:

print("No birthday records found for today.")

else:

print(f"Found {len(results)} birthday(s) for today.")

return results

except Exception as e:

print(f"Unexpected error: {e}")

raise DatabaseConnectionError(f"Unexpected error: {e}")

finally:

if cursor:

cursor.close()

print("Cursor closed.")

# Function to randomly pick a background color

def get\_random\_dark\_background\_color(self):

return random.choice(dark\_colors\_hex)

def send\_birthday\_email(self, birthday\_data):

if not birthday\_data:

print("No birthdays found for today.")

return

formatted\_today = datetime.now().strftime("%B %d, %Y")

subject = "ðŸŽ‰ Happy Birthday to Our Team Members! ðŸŽ‰"

background\_color = self.get\_random\_dark\_background\_color()

birthday\_cards = self.generate\_birthday\_cards(birthday\_data)

body = f"""

<div style="font-family: 'Arial', sans-serif; margin: 0; padding: 20px; background-color: {background\_color}; color: #ECF0F1; text-align: center; max-width: 800px; margin: auto;">

<h1 style="color: #FFD700; font-size: 2.5em; margin-bottom: 20px;">ðŸŽ‰ Happy Birthday! ðŸŽ‰</h1>

<p style="font-size: 1.5em; color: #ECF0F1;">Today is <strong style="color: #FFD700;">{formatted\_today}</strong> and we celebrate:</p>

<div style='display: flex; flex-wrap: wrap; justify-content: center;align-items: center; text-align: center;'>{birthday\_cards}</div>

<p style="font-size: 1.5em; margin-top: 20px; color: #ECF0F1;">Wishing everyone a wonderful day filled with love and happiness!</p>

<p style="font-size: 1.5em; font-weight: bold; color: #FFD700;">ðŸ’– NECO FAMILY ðŸ’–</p>

</div>

"""

self.send\_email(subject, body)

def generate\_birthday\_cards(self, birthday\_data):

birthday\_cards = ""

for index, person in enumerate(birthday\_data):

card\_html = f"""

<div style="position: relative; margin: 10px; padding: 10px; background-color: rgba(255, 255, 255, 0.2); border-radius: 10px; width: 250px; max-width: 250px; height: 200px; text-align: center;">

<h3 style="color: #FFD700; margin: 0; padding: 0;">SHRI {person[0].replace("SHRI", "")}</h3>

<p style="color: #ECF0F1;">{person[2]}<br><br>-</p>

<!-- Button styled as fixed at the bottom -->

<a href="mailto:?subject=Happy Birthday,{person[0].replace('SHRI','')}&body=Dear%20{person[0].replace('SHRI','')},%0D%0A%0D%0AðŸŽ‰%20Wishing%20you%20a%20wonderful%20birthday!ðŸŽ‚%0D%0A%0D%0A%20I%20hope%20this%20year%20brings%20you%20good%20health,%20joy,%20and%20everything%20you%20wish%20for.%20May%20your%20special%20day%20be%20filled%20with%20love,%20laughter,%20and%20memories%20to%20treasure.I%E2%80%99m%20truly%20thankful%20to%20have%20you%20in%20my%20life%20and%20for%20the%20friendship%20we%20share.%20Enjoy%20every%20moment%20today,%20and%20may%20this%20year%20bring%20you%20even%20more%20success%20and%20happiness.%20Take%20care%20and%20celebrate%20your%20day%20in%20the%20most%20special%20way.%20Happy%20birthday%20again%2C%20and%20here's%20to%20another%20great%20year%20ahead."

style="position: absolute; bottom: 10px; left: 50%; transform: translateX(-50%); padding: 10px 20px; background-color: #0066CC; color: white; font-size: 14px; text-decoration: none; border-radius: 6px; font-weight: bold; font-style: italic; font-family: cursive; transition: background-color 0.3s ease, transform 0.3s ease; text-align: center;">

<span style="font-family: Arial, sans-serif;">Wish Birthday! ðŸŽ‰</span>

</a>

</div>

"""

birthday\_cards += card\_html

if (index + 1) % 3 == 0 and index + 1 != len(birthday\_data):

birthday\_cards += "</div><div style='display: flex; flex-wrap: wrap; justify-content: center; align-items: center; text-align: center;'>"

return birthday\_cards

def send\_email(self, subject, body):

msg = MIMEMultipart()

msg['From'] = self.email\_config['from']

msg['To'] = self.email\_config['to']

# msg['bcc'] = self.email\_config['bcc']

msg['Subject'] = subject

msg.attach(MIMEText(body, 'html'))

try:

with smtplib.SMTP(self.email\_config['smtp\_server'], self.email\_config['smtp\_port']) as server:

server.starttls()

server.login(self.email\_config['from'], self.email\_config['password'])

server.send\_message(msg)

print(f"Birthday email sent to {msg['To']}")

except Exception as e:

print(f"Failed to send email: {e}")

def close\_connection(self):

if self.connection:

self.connection.close()

print("Database connection closed.")

if \_\_name\_\_ == "\_\_main\_\_":

db\_config = {

'user': 'jnl',

'password': 'jnl123',

'host' : '80.0.1.81',

'port': 1521,

'service\_name' : "jnil"

}

dark\_colors\_hex = [

"#2c4850",

"#2c5046",

"#27381f",

"#2d2e19",

"#3b2c20",

"#2a3439", # Gunmetal

"#0b0b36", # Midnight Blue

"#011c01", # Dark Green

"#250533", # Dark Violet (Still a bit bold, but kept for its dark tone)

"#2C3E50"

]

email\_config = {

'from': "myneco@necoindia.com",

'to': "saket.verma@necoindia.com",

# 'bcc': "saket.verma@necoindia.com",

'smtp\_server': 'smtp.gmail.com',

'smtp\_port': 587,

'password': "ukzu matf aupi kesv"

}

notifier = BirthdayNotifier(db\_config, email\_config)

try:

notifier.connect\_to\_db()

birthday\_data = notifier.get\_birthday\_data()

notifier.send\_birthday\_email(birthday\_data)

except DatabaseConnectionError as e:

print(e)

finally:

notifier.close\_connection()