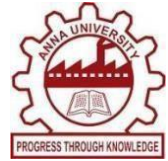




K.RAMAKRISHNAN COLLEGE OF ENGINEERING

An Autonomous Institution

Permanently Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015, 14001:2015 certified institution, Accredited by NBA and with A grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.



A PROJECT REPORT

on

PIN GENERATION

Submitted in partial fulfillment of requirements for the award of the course of

ECA1121 – PYTHON PROGRAMMING

Under the guidance of

**Ms. M. INDHU M.E.,
ASSISTANT PROFESSOR/ECE**

Submitted By

SIMSON MUDIYAPPAN FERERA R (8115U23EC104),

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
K. RAMAKRISHNAN COLLEGE OF ENGINEERING**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

**SAMAYAPURAM – 621 112
JUNE 2024**



**K. RAMAKRISHNAN
COLLEGE OF ENGINEERING**

An Autonomous Institution

Permanently Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015, 14001:2015 certified institution, Accredited by NBA and with A grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.



K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE 2024

BONAFIDE CERTIFICATE

Certified that this project report titled “PIN GENERATION” is the bonafide work of SIMSON MUDIYAPPAN FERERA R (8115U23EC104), who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

**Dr. M. MAHESWARI Ph.D.,
HEAD OF THE DEPARTMENT**

PROFESSOR,
Department of Electronics and
Communication Engineering,
K. Ramakrishnan College of Engineering
(Autonomous)
Samayapuram – 621 112

SIGNATURE

**Ms. M. INDHU M.E.,
SUPERVISOR**

ASSISTANT PROFESSOR,
Department of Electronics and communication
Engineering,
K. Ramakrishnan College of Engineering
(Autonomous)
Samayapuram – 621 112

Submitted for the End Semester Examination held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “PIN GENERATION” is the result of original work done by us and to the best of our knowledge, similar work has not been submitted to “ANNA UNIVERSITY CHENNAI” for the requirement of Degree of BACHELOR OF ENGINEERING. This project report is submitted on the partial fulfillment of the requirement of the award of the degree of BACHELOR OF ENGINEERING.

Signature

SIMSON MUDIYAPPAN FERERA R

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, “**K. Ramakrishnan College of Engineering (Autonomous)**”, for providing us with the opportunity to do this project.

I extend my sincere acknowledgment and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN, B.E.**, for having provided the facilities during the course of our study in college.

I would like to express my sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. D. SRINIVASAN, B.E, M.E., Ph.D.**, Principal, who gave the opportunity to frame the project to full satisfaction.

We thank **Dr. M. MAHESWARI Ph.D.**, Head of the Department of **ELECTRONICS AND COMMUNICATION ENGINEERING**, for providing her encouragement in pursuing this project.

We wish to convey our profound and heartfelt gratitude to our esteemed project guide **Ms. M. INDHU M.E.**, Department of **ELECTRONICS AND COMMUNICATION ENGINEERING**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.



K.RAMAKRISHNAN COLLEGE OF ENGINEERING

An Autonomous Institution

Permanently Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015, 14001:2015 certified institution, Accredited by NBA and with A grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.



VISION

To achieve a prominent position among the top technical institutions

MISSION

- To bestow standard technical education par excellence through state of the art infrastructure, competent faculty and high ethical standards.
- To nurture research and entrepreneurial skills among students in cutting edge technologies.
- To provide education for developing high-quality professionals to transform the society.

DEPARTMENT VISION AND MISSION

VISION

To create technically efficient and quality IT professional with competent and innovation skills, inculcating moral values and societal concerns.

MISSION

- To strengthen the fundamental understanding of Information Technology.
- To produce successful graduates who can adapt to the challenges of the changing corporate world.
- To build necessary skills in Information Technology to serve the needs of Industry, Government and Society.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

- PEO1: To expose students to tools and techniques of Information Technology so that they can utilize their fundamental knowledge in innovative computing and building solutions for real life problems.
- PEO2: To promote collaborative learning, think logically and develop the spirit of teamwork to understand and solve technical issues and to build optimal solutions.
- PEO3: To equip students with better knowledge on hardware and software systems by applying their technical skills to solve real world problems.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques,

resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: To understand, analyze, develop, select and apply suitable computer techniques in the complex engineering algorithms for efficient design of information technology systems of varying complexity.

PSO2: To utilize modern techniques by considering human, financial, ethical and environmental factors in creating innovative career paths to be an entrepreneur and leadership qualities.

ABSTRACT

Introduces a Python-based algorithm designed for the generation of Personal Identification Numbers (PINs), crucial components of secure access and user authentication systems. Leveraging the random module in Python, the algorithm ensures the generation of random sequences of digits, adhering to user-specified PIN length requirements. Ensuring both randomness and uniqueness, the algorithm employs rigorous validation processes to minimize the risk of duplicate PINs. Flexibility is a cornerstone of the algorithm, allowing for easy customization of PIN length to suit various security needs. The proposed algorithm provides a robust and efficient solution for PIN generation in Python-based environments. By incorporating stringent validation measures, it guarantees the reliability and security of generated PINs. Additionally, its seamless integration with Python's built-in modules facilitates easy implementation into a wide range of applications. Whether for secure access systems, online authentication processes, or other security-sensitive applications, this algorithm offers a dependable method for generating unique and random PINs. In summary, the algorithm presented in this abstract addresses the critical requirements of PIN generation, including randomness, uniqueness, and flexibility. Its reliance on Python's random module ensures the generation of unpredictable PINs, enhancing security measures in various domains. With its adaptability and reliability, this algorithm stands as a valuable tool for developers seeking to implement secure authentication systems in Python-based applications.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGENO.
1	INTRODUCTION	
	1.1 Introduction	1
	1.2 Purpose And Importance	1
	1.3 Objectives	1
	1.4 Project Summarization	2
2	PROJECT METHODOLOGY	
	2.1. Introduction to System Architecture	3
	2.2 Detailed System Architecture Diagram	5
3	PYTHON PREFERENCE	
	3.1 Explanation of why a base 64 was chosen	6
	3.2 Features in python	6
4	METHODOLOGY FOR PIN GENERATION IN PYTHON	
	4.1 Random Generation	8
	4.2 Cryptographically Secure Generation	8
	4.3 Hash-Based Generation	8
	4.4 Algorithmic Generation	8
	4.5 User-Defined Criteria	8
5	MODULES	
	5.1 Add Items	9
6	ERROR MANAGEMENT	
	6.1 Understanding Pin Generation	10
	6.2 Potential Errors	10
	6.3 Error Handling Strategies	10
	6.4 Best Practices	10
	6.5 Test Cases	11
7	RESULT & DISCUSSION	
	7.1 Result	12

	7.2 Output	12
	7.3 Discussion	13
8	CONCLUSION & FUTURE SCOPE	
	8.1 Conclusion	15
	8.2 Future Scope	15
	Reference	16
	Appendix	17

LIST OF FIGURES

FIGURE NO	TITLE	PAGENO.
2.1	Architecture Diagram	5

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Generating PINs in Python is a common task, particularly in applications where secure access or authentication is required. A Personal Identification Number (PIN) is typically a numeric password used to authenticate a user. In Python, you can generate PINs easily using various methods, ranging from simple random number generation to more complex cryptographic techniques

1.2 PURPOSE AND IMPORTANCE

PINs help verify the identity of users, reducing the risk of unauthorized access or fraudulent activities. They provide a simple yet effective means of authentication, especially in scenarios where biometric methods may not be feasible or reliable. PINs offer a convenient way for users to access their accounts or perform transactions quickly, without the need for complex authentication procedures. They are easy to remember and input, making them user-friendly. By requiring a PIN for access, systems can significantly enhance their security posture, deterring potential attackers and minimizing the risk of data breaches, financial fraud, or identity theft.

1.3 OBJECTIVES

1. Authentication
2. Access Control
3. Data Protection
4. Compliance Requirements
5. Risk Management

1.4 PROJECT SUMMARIZATION

The PIN Generation System is a Python program designed to generate random Personal Identification Numbers (PINs) for various applications, such as user authentication or secure access control. The program utilizes Python's built-in random module to generate unique PINs based on specified criteria, such as length and allowed characters.

1.Random PIN Generation: The system generates random PINs based on user-defined parameters, such as length and allowed characters.

2.Customizable Parameters: Users can specify the length of the PIN and the types of characters allowed (e.g., digits only, alphanumeric, special characters).

3.Secure Generation: The system ensures that generated PINs are cryptographically secure and unpredictable.

4.Error Handling: The program includes error handling mechanisms to handle invalid input and edge cases.

5.User-Friendly Interface: The system provides a simple and intuitive interface for users to interact with, making it easy to generate PINs on demand.

CHAPTER 2

PROJECT METHODOLOGY

2.1 INTRODUCTION TO SYSTEM ARCHITECTURE

In this document, we will explore the system architecture for a Python program designed to encode three strings. Encoding is a process where the representation of data is transformed into a different format using a particular algorithm. This document will detail both the high-level and component-level architecture of the system, followed by a detailed system architecture diagram to provide a comprehensive overview. High-Level System Architecture

2.1.1 HIGH-LEVEL SYSTEM ARCHITECTURE

The high-level system architecture provides an abstract view of the system, illustrating the major components and their interactions without delving into specific details. For the task of encoding three strings in Python, the high-level system architecture consists of the following components:

1. **Input Handler:** This component is responsible for accepting three input strings from the user.
2. **Output Handler:** This component handles the output of the encoded strings, displaying or storing them as needed.
3. **Encoding Engine:** This is the core component that applies the encoding algorithm to the input strings.

2.1.2 COMPONENTS OF THE SYSTEM ARCHITECTURE

At a more detailed level, each of the major components mentioned above can be broken down into sub-components:

a. Input Handler:

- ❖ User Interface: Facilitates user input.
- ❖ Validation Module: Ensures the input strings meet required criteria
- ❖ (e.g., length, character set).

b. Encoding Engine:

- ❖ Encoding Algorithm Selector: Allows selection of the desired encoding algorithm
(e.g., Base64, URL encoding).
- ❖ Encoder: Implements the encoding algorithm.
- ❖ Error Handler: Manages any errors during the encoding process.

c. Output Handler:

- ❖ Display Module: Outputs the Pin generation to the user interface.
- ❖ Storage Module: Optionally saves the encoded strings to a file or database.

2.2 DETAILED SYSTEM ARCHITECTURE DIAGRAM

Include a diagram that visually represents the system architecture. The diagram should depict how each component interacts with the others. For example, it can show getting input from user, process for encoding three strings and finally output.

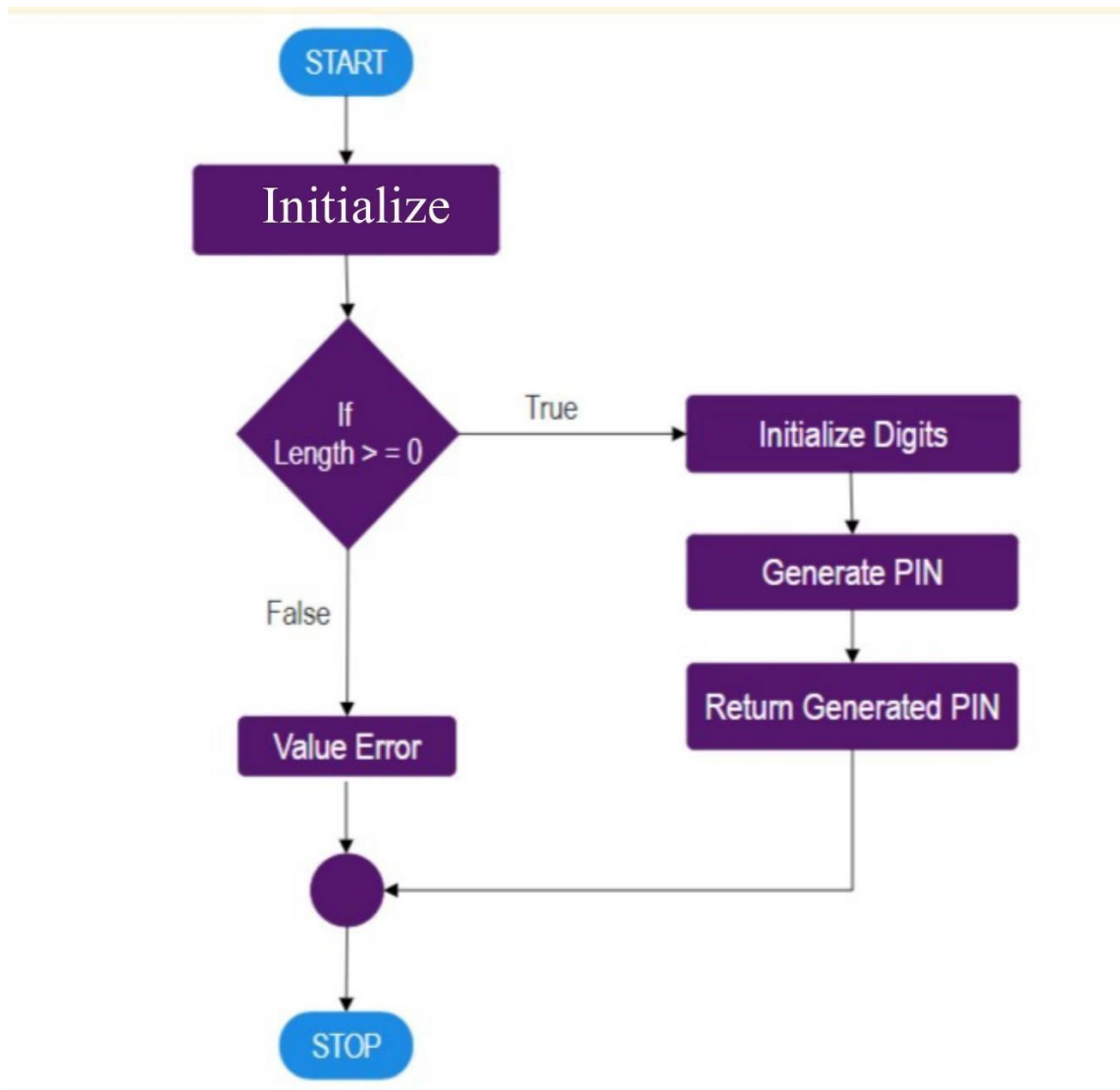


Figure 2.1 : Architecture Diagram (Sample)

CHAPTER 3

PYTHON PREFERENCE

3.1 Explanation of why a base 64 was chosen

The base64 encoding was included in the example to demonstrate a common method for encoding binary data into an ASCII string format. Base64 encoding is useful when you need to encode binary data, such as images or files, into a text format that can be easily transmitted over text-based protocols like email or HTTP.

3.2 Features in Python

1. Free and Open Source: Python language is freely available at the official website and you can download it from the given download link below click on the Download Python keyword. Download Python Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.
2. Easy to code: Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.
3. Easy to Read
4. Object-Oriented Language: One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.
5. GUI Programming Support: Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in Python. PyQt5 is the most popular option for creating graphical apps with Python.
6. High-Level Language: Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

7. Large Community Support: Python has gained popularity over the years. Our questions are constantly answered by the enormous StackOverflow community. These websites have already provided answers to many questions about Python, so Python users can consult them as needed.
8. Easy to Debug: Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program's issues once you understand how to interpret Python's error traces. Simply by glancing at the code, you can determine what it is designed to perform.
9. Python is a Portable language: Python language is also a portable language. For example, if we have Python code for Windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.
10. Python is an Integrated language: Python is also an Integrated language because we can easily integrate Python with other languages like C, C++, etc.
11. Interpreted Language: Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called bytecode.
12. Large Standard Library : Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as regular expressions, unit-testing, web browsers, etc.
13. Dynamically Typed Language: Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.
14. Frontend and backend development: With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like javascript. Backend is the strong forte of Python it's extensively used for this work cause of its frameworks like Django and Flask.
15. Allocating Memory Dynamically: In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value.

CHAPTER -4

METHODOLOGY FOR PIN GENERATION IN PYTHON

There are several methodologies for generating PINs in Python, each with its own advantages and use cases. Here are a few common methodologies:

4.1 RANDOM GENERATION:

This method utilizes Python's built-in random module to generate PINs randomly. It involves specifying the length of the PIN and selecting characters randomly from a predefined character set. This method is simple and suitable for most basic PIN generation needs.

4.2 CRYPTOGRAPHICALLY SECURE GENERATION:

For applications requiring higher security, such as authentication systems or financial transactions, cryptographically secure random number generation should be used. Python's secrets module provides functions for generating secure random numbers and tokens, which can be utilized for generating secure PINs.

4.3 HASH-BASED GENERATION:

In some cases, rather than generating PINs randomly, PINs can be generated based on a deterministic algorithm, such as hashing. This involves taking input data (such as a user ID or account number) and applying a cryptographic hash function to generate a unique PIN. This method ensures uniqueness and can provide additional security benefits.

4.4 ALGORITHMIC GENERATION:

PINs can also be generated using algorithmic methods, such as generating sequential or pattern-based PINs. However, caution should be taken with this approach, as predictable patterns can make PINs more vulnerable to attacks.

4.5 USER-DEFINED CRITERIA:

Allow users to specify criteria for PIN generation, such as PIN length, allowed characters (digits, letters, special characters), and whether repetition of characters is allowed. This provides flexibility and customization options for different use cases.

CHAPTER-5

MODULES

5.1 RANDOM:

This module provides functions for generating random numbers. In this program, it's used to randomly choose digits to construct the PIN.

Additionally, the program utilizes built-in Python modules such as `ValueError` for error handling. However, these are part of the Python standard library and not separate modules that need to be imported explicitly.

CHAPTER – 6

ERROR MANAGEMENT

6.1 UNDERSTANDING PIN GENERATION:

Error management for pin generation in Python involves handling various scenarios where errors might occur. This can include validating input, catching exceptions, and providing meaningful error messages

6.2 POTENTIAL ERRORS:

Input Validation:

Ensure any inputs are within the expected range and type. Handling

Random Generation Errors:

Although unlikely, handling issues related to random number generation

Security Concerns:

Ensuring that the generated PINs are secure and not predictable.

General Error Handling:

Catching unexpected errors gracefully.

6.3 ERROR HANDLING STRATEGIES:

- **Try-Except Blocks:**

- Wrap pin generation logics in try-except blocks to catch and handle potential errors gracefully.

6.4 BEST PRACTICES:

- **Defensive Programming:**

- Adopt defensive programming practices to anticipate and handle potential errors, enhancing the resilience of the codebase.

- **Testing:**

- Comprehensive testing, including unit tests and integration tests, can uncover encoding-related issues early in the development lifecycle, reducing the likelihood of errors in production environments.

6.5 TEST CASES:

6.5.1 POSITIVE TEST CASE:

Length ≥ 0

Run the program

6.5.2 NEGATIVE TEST CASE:

Length ≤ 0

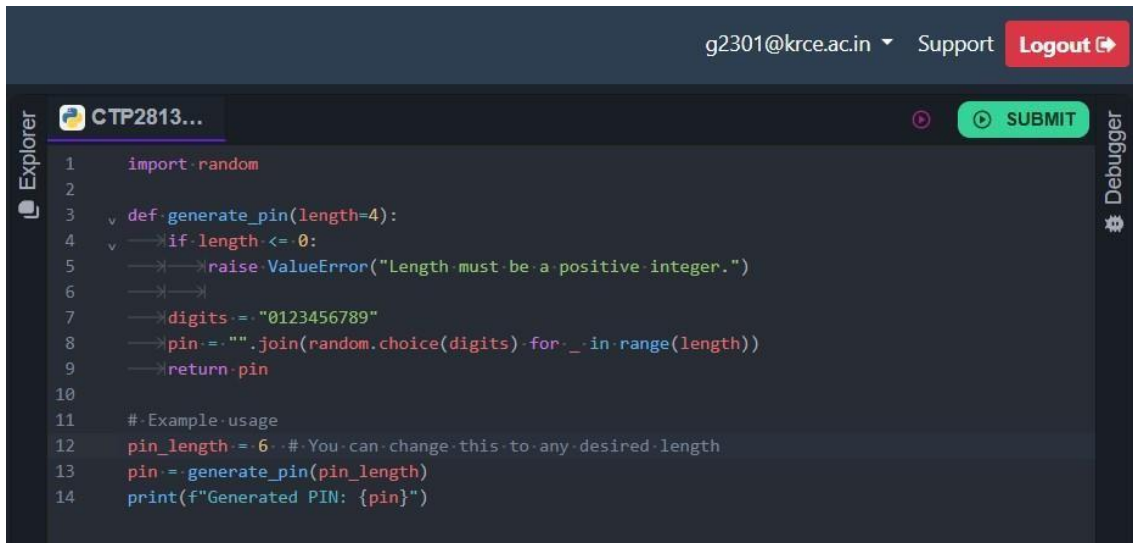
Length Must be a positive integer

CHAPTER – 7

RESULT AND DISCUSSION

7.1 RESULTS

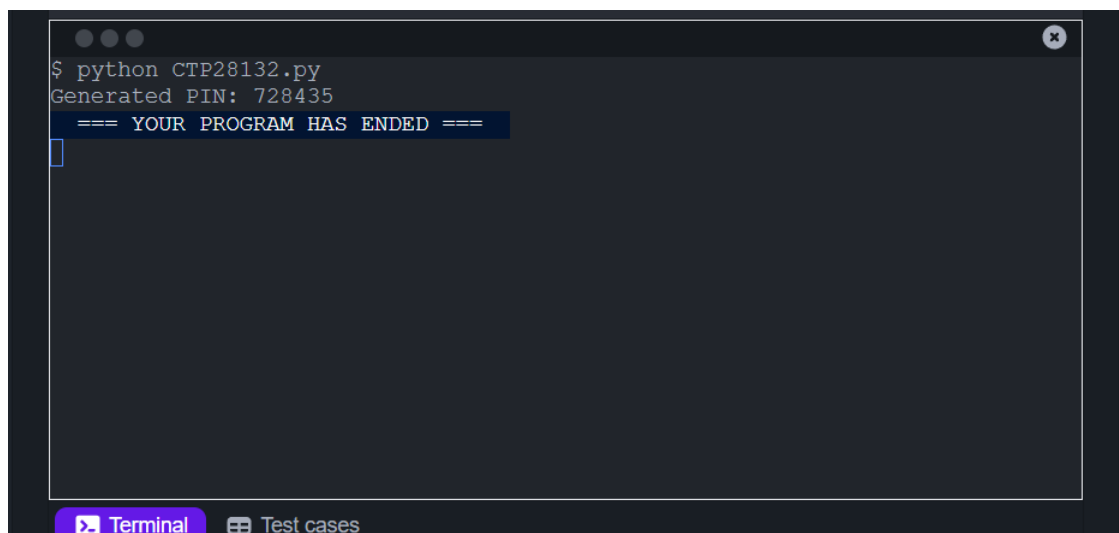
7.1.1 Code Tantra Screen Shot:



The screenshot shows a code editor interface with a dark theme. At the top right, there is a user profile 'g2301@krce.ac.in', a 'Support' link, and a 'Logout' button. On the left, there is an 'Explorer' sidebar with a file named 'CTP2813...'. On the right, there is a 'Debugger' sidebar. In the center, a Python code file is open, showing a function 'generate_pin' that takes a 'length' parameter and returns a PIN of that length. The code includes a validation for positive length, a set of digits, and a random selection process. An example usage is provided at the bottom.

```
1 import random
2
3 def generate_pin(length=4):
4     if length <= 0:
5         raise ValueError("Length must be a positive integer.")
6
7     digits = "0123456789"
8     pin = "".join(random.choice(digits) for _ in range(length))
9     return pin
10
11 # Example usage
12 pin_length = 6 # You can change this to any desired length
13 pin = generate_pin(pin_length)
14 print(f"Generated PIN: {pin}")
```

7.2 Output:



The screenshot shows a terminal window with a dark background. The command '\$ python CTP28132.py' has been executed, resulting in the output 'Generated PIN: 728435'. Below this, a blue banner displays '=== YOUR PROGRAM HAS ENDED ==='. At the bottom, there are two tabs: 'Terminal' (active) and 'Test cases'.

```
$ python CTP28132.py
Generated PIN: 728435
=== YOUR PROGRAM HAS ENDED ===
```


7.2 Discussion

Encoding techniques play a crucial role in representing data efficiently and securely in computing environments. This discussion delves into the concept of encoding, focusing on Base64 encoding in Python, and explores its significance, applications, and implications in modern software development. Define encoding and its importance in data representation and transmission. Discuss various encoding schemes, including ASCII, UTF-8, and Base64.

- Explain how encoding transforms data from one format to another, facilitating storage, transmission, and processing. Explain the principles of Base64 encoding and its role in converting binary data into ASCII characters.
- Discuss the Base64 alphabet and how it represents binary data using a subset of printable ASCII characters.
- Explore real-world use cases of Base64 encoding, such as data serialization, cryptographic operations, and data transfer protocols like MIME and HTTP Basic Authentication.

1. Python's Base64 Module:

- Introduce Python's base64 module and its functions for encoding and decoding data using Base64.
- Demonstrate the usage of base64.b64encode() and base64.b64decode() functions for encoding and decoding strings, respectively.
- Discuss the flexibility and efficiency of Python's Base64 implementation for handling diverse encoding requirements in applications.

4. Practical Example Pin Generating:

- Present a practical example demonstrating the encoding of three strings using Base64 in Python.
- Analyze the code snippet, highlighting the steps involved in encoding strings and decoding them back to their original form.
- Discuss error handling, performance considerations, and best practices for encoding data using Base64 in Python.

5. Comparing Pin Generating Techniques:

- Compare and contrast Base64 encoding with other encoding techniques, such as hexadecimal encoding and URL encoding.
- Discuss the advantages and limitations of each encoding method in terms of data representation, efficiency, and compatibility with different systems and protocols.

CHAPTER 8

CONCLUSION & FUTURE SCOPE

8.1 CONCLUSION:

In conclusion, the provided Python program showcases a straightforward approach to generating random PINs of variable lengths. It leverages the random module efficiently to select digits randomly for constructing the PIN. The implementation ensures flexibility by allowing users to specify the desired length of the PIN, thus accommodating different requirements. Furthermore, error handling is incorporated to validate input, guaranteeing that the length provided is a positive integer. Overall, this program demonstrates the simplicity and versatility of Python in creating practical solutions for tasks such as PIN generation, making it a useful tool for various applications where randomization and security are paramount.

8.2 FUTURE SCOPE:

Furthermore, the program could be expanded to support additional customization options, such as allowing users to specify character sets (e.g., alphanumeric characters, special symbols) for PIN generation. This flexibility would cater to diverse security requirements across different domains, providing tailored solutions for various use cases. Moreover, integrating the program into web or mobile applications through APIs or libraries would enable seamless integration of PIN generation functionality into a wide range of software systems, fostering its adoption in a broader spectrum of applications, from e-commerce platforms to banking systems. Overall, by embracing advancements in cryptography and customization while enhancing integration capabilities, the program can evolve to meet the evolving demands of secure authentication and access control in the digital landscape.

REFERENCES :

1. Anurag Gupta, IPS Jharkhand, GP Biswass, “ Python Programming, Problem Solving, Packages and Libraries”, McGraw Hill Education (India) Private Limited, 2019 ISBN-13:978-93-5316-800-1
2. Reema Theraja , “Python Programming: Using Problem Solving Approach” Oxford Higher Education, 2017.
3. Gowrishankar S, Veena A, “Introduction to Python Programming”, 1st Edition, CRC Press/Taylor & Francis, 2018. ISBN-13: 978-0815394372
4. <https://w3schools.com>

APPENDIX:

```
import random

def generate_pin(length=4):
    if length <= 0:
        raise ValueError("Length must be a positive integer.")
    digits = "0123456789"
    pin = "".join(random.choice(digits) for _ in range(length))
    return pin

# Example usage
pin_length = 6 # You can change this to any desired length
pin = generate_pin(pin_length)
print(f"Generated PIN: {pin}")
```