What do you call a computer superhero?

A Screen Saver!

Source: https://www.scarymommy.com/computer-jokes

Thank you Nick!
Last Lecture

- Introduced **Cryptography and Encryption**
- Arithmetic operators
  - **Modulo operator %**
- String indexing (works for lists as well)

  positive indexing->     index: 0 1 2 3 4 5 6 7 8 9 10 11 12
  \[\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\]

  **Example:** message = "Hello, World!"

- We started solving the problem of encryption

  **Step 1 - Problem Statement:**
  - Write a Python program that encrypts messages using a **transposition algorithm** called **odd&even**
Today’s Menu

• Continue investigating the topic of Cryptography and Encryption
  1. By finishing the implementation of our OddEvenEncryption program
     • Implementing the encryption algorithm
  2. By adding decryption to our OddEvenEncryption program
     • Implementing the decryption algorithm

• Introduce functions
Review from last lecture:
Transposition algorithm odd&even:

1. Get \textit{plainMsg} from user
2. Create a \textit{cipherMsg} that is made of 2 strings
   - \textbf{String1} contains the characters located in odd positions in \textit{plainMsg}
   - \textbf{String2} contains the characters located in even positions in \textit{plainMsg}
3. Lastly, concatenate these two strings:
   \textit{cipherMsg} = \textit{String1} + \textit{String2}
Let’s get coding!

• Step 2 – Design
  • Transposition algorithm odd&even
    • encryption part

• Step 3 – Implementation

• Step 4 - Testing
Transposition algorithm **odd&even**

Transposition algorithm **odd&even**:  
1. Find the middle of `cipherMsg` 
2. Store the first half of `cipherMsg` into `String3`  
   • This first half contains the characters originally located in **odd positions** of `plainMsg`  
3. Store the last half of `cipherMsg` into `String4`  
   • This second half contains the characters originally located in **even positions** of `plainMsg`  
4. Lastly, merge `String3` with `String4` to recreate `plainMsg`
Your turn

- **Problem Statement:**
  - Write a program that **encrypts** and **decrypts** messages using the **transposition algorithm** odd&even

- **Requirement:**
  - Your program must go on encrypting and decrypting messages entered by the user until the user only presses the ENTER key.
Review – Terminology related to functions

1. We call a function by name
   - Example of a function call:
     ```python
     userName = input("Please, enter your name: ")
     ```

2. The name of the function is `input`

3. The expression in parentheses is called the argument of the function
   ```python
   userName = input("Please, enter your name: ")
   ```

4. We say that a function takes an argument(s)

5. The result of the function is what the function produces
   - In the above example, the result of the function is what the user has typed, i.e., her/his name, which is assigned to `userName`
     ```python
     'Anne'
     ```

6. We say that a function returns a result and the result is called the returned value
Next Lecture

- Continue investigating **functions**
- We shall look at another way of repeating Python statements in our programs, i.e., another iterative statement:
  - *while* loop