Day 1: Division and Square-root

Write a function called ***divide_or_square*** that takes one argument (a number), and returns the square root of the number if it is divisible by 5, returns its remainder if it is not divisible by 5. For example, if you pass **10** as an argument, then your function should return **3.16** as the square root**.**

Extra Challenge: Longest Value

Write a function called ***longest_value*** that takes a dictionary as an argument and returns the **first** longest value of the dictionary. For example, the following dictionary should return – **apple** as the longest value.

fruits = {'fruit': 'apple', 'color': 'green'}

Day 2: Strings to Integers

Write a function called ***convert_add*** that takes a list of strings as an argument and converts it to integers and sums the list. For example **['1', '3', '5']** should be converted to **[1, 3, 5]** and **summed to 9.**

Extra Challenge: Duplicate Names

Write a function called ***check_duplicates*** that takes a list of strings as an argument. The function should check if the list has any duplicates. If there are duplicates, the function should return the duplicates. If there are no duplicates, the function should return **"No duplicates"**. For example, the list of **fruits** below should return **apple** as a duplicate and list of **names** should return "no duplicates".

```
fruits = ['apple', 'orange', 'banana', 'apple']
names = ['Yoda', 'Moses', 'Joshua', 'Mark']
```

Day 3: Register Check

Write a function called ***register_check*** that checks how many students are in school. The function takes a dictionary as a parameter. If the student is in school, the dictionary says **'yes'**. If the student is not in school, the dictionary says **'no'**. Your function should return the number of students in school. Use the dictionary below. Your function should return 3.

```
register = {'Michael':'yes','John': 'no',
            'Peter':'yes', 'Mary': 'yes'}
```

Extra Challenge: Lowercase Names

names = [**"kerry"**, **"dickson"**, **"John"**, **"Mary"**,
"carol", **"Rose"**, **"adam"**]

You are given a list of names above. This list is made up of names of lowercase and uppercase letters. Your task is to write a code that will return a tuple of all the names in the list that have only lowercase letters. Your tuple should have names sorted alphabetically in descending order. Using the list above, your code should return:

('kerry', 'dickson', 'carol', 'adam')

Day 4: Only Floats

Write a function called ***only_floats***, which takes two parameters ***a*** and ***b***, and returns **2** if both arguments are floats, returns **1** if only one argument is a float, and returns **0** if neither argument is a float. If you pass **(12.1, 23)** as an argument, your function should return a **1**.

Extra Challenge: Index of the Longest Word

Write a function called ***word_index*** that takes one argument, a list of strings and returns the index of the longest word in the list. If there is no longest word (if all the strings are of the same length), the function should return zero (0). For example, the list below should return 2.

words1 = ["Hate", "remorse", "vengeance"]

And this list below, shoul return zero (0)

words2 = ["Love", "Hate"]

Day 5: My Discount

Create a function called ***my_discount***. The function takes no arguments but asks the user to input the **price** and the **discount** (percentage) of the product. Once the user inputs the price and discount, it calculates the **price after the discount**. The function should return the price after the discount. For example, if the user enters **150** as price and **15%** as the discount, your function should return **127.5**.

Extra Challenge: Tuple of Student Sex

You work for a school and your boss wants to know how many female and male students are enrolled in the school. The school has saved the students in a list. Your task is to write a code that will count how many males and females are in the list. Here is a list below:

students = ['Male', 'Female', 'female', 'male', 'male', 'male',
'female', 'male', 'Female', 'Male', 'Female', 'Male', 'female']

Your code should return a list of tuples. The list above should return:

[('Male',7), ('female',6)]

Day 6: User Name Generator

Write a function called ***user_name*** that generates a username from the user's email. The code should ask the user to input an email and the code should return everything before the **@** sign as their user name. For example, if someone enters **ben@gmail.com**, the code should return ben as their user name.

Extra Challenge: Zero Both ends

Write a function called ***zeroed*** code that takes a list of numbers as an argument. The code should zero (0) the first and the last number in the list. For example, if the input is **[2, 5, 7, 8, 9]**, your code should return **[0, 5, 7, 8, 0]**.

Day 7: A String Range

Write a function called ***string_range*** that takes a single number and returns a string of its range. The string characters should be **separated by dots**(.) For example, if you pass **6** as an argument, your function should return **'0.1.2.3.4.5'**.

Extra Challenge: Dictionary of names

You are given a list of names, and you are asked to write a code that returns all the names that start with 'S'. Your code should return a dictionary of all the names that start with *S* and how many times they appear in the dictionary. Here is a list below:

```
names = ["Joseph","Nathan", "Sasha", "Kelly",
        "Muhammad", "Jabulani", "Sera", "Patel", "Sera"]
```

Your code should return: **{"Sasha": 1, "Sera": 2}**

Day 8: Odd and Even

Write a function called ***odd_even*** that has one parameter and takes a list of numbers as an argument. The function returns the difference between the largest **even** number in the list and the smallest **odd** number in the list. For example, if you pass **[1,2,4,6]** as an argument the function should return **6 -1= 5**.

Extra Challenge: List of Prime Numbers

Write a function called ***prime_numbers***. This function asks a user to enter a number (integer) as an argument and returns a list of all the prime numbers within its range. For example, if user enters 10, your code should return **[2, 3, 5, 7]** as prime numbers.

Day 9: Biggest Odd Number

Create a function called ***biggest_odd*** that takes a string of numbers and returns the biggest odd number in the list. For example, if you pass **'23569'** as an argument, your function should return **9**. Use **list comprehension**.

Extra Challenge: Zeros to the End

Write a function called ***zeros_last***. This function takes a list as argument. If a list has zeros (0), it will move them to the front of the list and maintain the order of the other numbers in the list. If there are no Zeros in the list, the function should return the original list sorted in ascending order. For example, if you pass **[1, 4, 6, 0, 7,0,9]** as an argument, your code should return **[1, 4, 6, 7, 9, 0, 0]**. If you pass [**2, 1, 4, 7, 6]** as your argument, your code should return **[1, 2, 4, 6, 7].**

Day 10: Hide my Password

Write a function called ***hide_password*** that takes no parameters. The function takes an input (a password) from a user and returns a hidden password. For example, if the user enters 'hello' as a password the function should return '****' as a password and tell the user that the password is **4 characters** long.

Extra Challenge: A Thousand Separator

Your new company has a list of figures saved in a list. The issue is that these numbers have **no separator**. The numbers are saved in the following format:
[1000000, 2356989, 2354672, 9878098].

You have been asked to write a code that will convert each of the numbers in the list into a **string**. Your code should then add a comma on each number as **a thousand separator** for readability. When you run your code on the above list, your output should be:

['1,000,000', '2,356,989', '2,354,672', '9,878,098']

Write a function called ***convert_numbers*** that will take one argument, a list of numbers above.

Day 11: Are They Equal?

Write a function called ***equal_strings***. The function takes two strings as arguments and compares them. If the strings are equal (if they have the same characters and have equal length), it should return **True**, if they are not, it should return **False**. For example, 'love' and 'evol' should return **True**.

Extra Challenge: Swap Values

Write a function called ***swap_values***. This function takes a list of numbers and swaps the first element with the last element. For example, if you pass **[2, 4,67, 7]** as a parameter, your function should return
[7, 4, 67, 2].

Day 12: Count the Dots

Write a function called ***count_dots***. This function takes a string separated by dots as a parameter and counts how many dots are in the string. For example, '**h.e.l.p.**' should return **4** dots, and '**he.lp.**' should return **2** dots.

Extra Challenge: Your Age in Minutes

Write a function called ***age_in_minutes*** that tells a user **how old they are in minutes**. Your code should ask the user to enter their **year of birth,** and it should return their age in minutes (by subtracting their year of birth to the current year). Here are things to look out for:

a. The user can only input a **4-digit** year of birth. For example, 1930 is a valid year. However, entering any number longer or less than 4 digits long should render input invalid. Notify the user to input a four digits number.
b. If a user enters a year **before** 1900, your code should tell the user that input is invalid. If the user enters the year **after the** current year, the code should tell the user, to input a valid year.

The code should **run until the user inputs a valid year**. Your function should return the user's age in minutes. For example, if someone enters 1930, as their year of birth your function should return:

You are 48,355,200 minutes old.

Day 13: Pay Your Tax

Write a function called ***your_vat***. The function takes no parameter. The function asks the user to input the price of an item and VAT (vat should be a percentage). The function should return the price of the item plus VAT. If the price is 220 and, VAT is 15% your code should return a vat inclusive price of 253. Make sure that your code can handle **ValueError**. Ensure the code runs until valid numbers are entered. (hint: Your code should include a while loop).

Extra Challenge: Pyramid of Snakes

Write a function called ***Python_snakes*** that takes a number as an argument and creates the following shape, using the number's range: (hint: Use the loops and emoji library. If you pass 7 as argument, your code should print the following:

Day 14: Flatten the List

Write a function called ***flat_list*** that takes one argument, a nested list. The function converts the nested list into a one-dimension list. For example **[[2,4,5,6]]** should return **[2,4,5,6]**.

Extra Challenge: Teacher's Salary

A school has asked you to write a program that will calculate teachers' salaries. The program should ask the user to enter the teacher's **name**, the **number of periods** taught in a month, and the **rate** per period. The monthly salary is calculated by multiplying the **number of periods** by the **monthly rate**. The current monthly rate per period is $20. If a teacher has more than 100 periods in a month, everything above 100 is overtime. Overtime is $25 per period. For example, if a teacher has taught **105** periods, their monthly gross salary should be **2,125.** Write a function called ***your_salary*** that calculates a teacher's gross salary. The function should return the **teacher's name**, **periods taught**, and **gross salary.** Here is how you should format your output:

Teacher: John Kelly,
Periods: 105
Gross salary:2,125

Day 15: Same in Reverse

Write a function called ***same_in_reverse*** that takes a string and checks if the string reads the same in reverse. If it is the same, the code should return **True** if not, it should return **False**. For example, **'dad'** should return **True**, because it reads the same in reverse.

Extra Challenge: What's My Age?

Write a function called ***your_age***. This function asks a student to enter their name and then it returns their age. For example, if a user enters **Peter** as their name, your function should return, **'Hi, Peter, you are 27 years old**. This function reads data from the database (dictionary below). If the name is not in the dictionary, your code should tell the user that their name is **not** in the dictionary, and ask them for their age. Your code should then add the **name** and **age** to the dictionary of **names_age** below. Once added your code should return to the user **'Hi, (added name), you are (added age) years old'**. Remember to convert the input from user to lowercase letters

```
names_age = {"jane": 23, "kerry": 45, "tim": 34, "peter": 27}
```

Day 16: Sum the List

Write a function called ***sum_list*** with one parameter that takes a **nested list** of integers as an argument and returns the sum of the integers. For example, if you pass **[[2, 4, 5, 6], [2, 3, 5, 6]]** as an argument your function should return a sum of **33**.

Extra Challenge: Unpack A Nest

Nested_list = [[12, 34, 56, 67], [34, 68, 78]]

Write a **code** that generates a list of the following numbers from the nested list above – 34, 67, 78. Your output should be another list:
[34, 67, 78]. The list output should not have duplicates**.**

Day 17: User Name Generator

Write a function called ***user_name***, that creates a username for the user. The function should ask a user to **input** their name. The function should then reverse the name and attach a randomly issued number between 0 – 9 at the end of the name. The function should return the **username**.

Extra Challenge: Sort by Length

names = ["Peter", "Jon", "Andrew"]

Write a function called ***sort_length*** that takes a list of strings as an argument, and sorts the strings in ascending order according to their length. For example, the list above should return:

['Jon', 'Peter', 'Andrew']

Do not use the built-in sort functions

Day 18: Any Number of Arguments

Write a function called ***any_number*** that can receive any number of arguments (integers and floats) and return the average of those integers. If you pass **12, 90, 12, 34** as arguments your function should return **37.0** as average. If you pass **12, 90** your function should return **51.0** as average.

Extra Challenge: Add and Reverse

Write a function called ***add_reverse***. This function takes **two** lists as argument and adds each corresponding number, and **reverses** the outcome. For example, if you pass **[10, 12, 34]** and **[12, 56, 78]**. Your code should return **[112, 22, 68].** If the two lists are not of equal lengths, the code should return a message that "the lists are not of equal lengths".

Day 19: Words and Elements

Write two functions. The first function is called ***count_words*** which takes a string of words and counts how many words are in the string.

The second function called ***count_elements*** takes a string of words and counts how many elements are in the string. Do not count the whitespaces. The first function will return the number of **words** in a string and the second one will return the number of **elements (**less whitespace). If you pass **'I love learning'**, the **count_words** function should return **3 words** and **count_elements** should return **13 elements**.

Day 20: Capitalize First Letter

Write a function called ***capitalize***. This function takes a string as an argument and **capitalizes** the first letter of each word. For example, **'i like learning'** becomes **'I Like Learning'**.

. Extra Challenge: Reversed List

```
str1 = 'leArning is hard, bUt if You appLy youRself ' \
       'you can achieVe success'
```

You are given a string of words. Some of the words in the string contain uppercase letters. Write a code that will return all the words that have an uppercase letter. Your code should return a list of the words. Each word in the list should be reversed. Here is how your output should look:

['gninrAel', 'tUb', 'uoY', 'yLppa', 'flesRuoy', 'eVeihca']

Day 21: List of Tuples

Write a function called ***make_tuples*** that takes two lists, **equal lists**, and combines them into a list of tuples. For example, if list **a** is **[1,2,3,4]** and list **b** is **[5,6,7,8]**, your function should return **[(1,5), (2,6), (3,7), (4,8)]**.

Extra Challenge: Even Number or Average

Write a function called ***even_or_average*** that ask a user to input five numbers. Once the user is done, the code should return the largest even number in the inputted numbers. If there is no even number in the list, the code should return the average of all the five numbers.

Day 22: Add Under_Score

Create **three** functions. The first called ***add_hash*** takes a string and adds a hash # between the words. The second function called ***add_underscore*** removes the hash (#) and replaces it with an underscore "_" The third function called ***remove_underscore***, removes the underscore and replaces it with nothing. if you pass **'Python'** as an argument for the three functions, and you call them at the same time like:

print(remove_underscore(add_underscore(add_hash('Python'))))

it should return **'Python'.**

Day 23: Simple Calculator

Create a simple calculator. The calculator should be able to perform basic math operations, **add**, **subtract**, **divide** and **multiply**. The calculator should take input from users. The calculator should be able to handle **ZeroDivisionError**, **NameError**, and **ValueError.**

Extra Challenge: Multiply Words

s = "love live and laugh"

Write a function called ***multiply_words*** that takes a string as an argument and multiplies the length of each word in the string by the length of other words in the string. For example, the string above should return **240** - **love (4) live (4) and (3) laugh (5).** However, your function should only multiply words will all lowercase letters. If a word has one upper case letter, it should be ignored. For example, the following string:
s = "Hate war love Peace" should return **12 – war (3) love (4).** **Hate** and **Peace** will be ignored because they have at least one uppercase letter.

Day 24: Average Calories

Create a function called **average_calories** that calculates the average calories intake of a user. The function should ask the user to input their calories intake for **any** number of days and once they hit '**done**' it should calculate and **return** the average intake.

Extra Challenge: Create a Nested List

Write a function called ***nested_lists*** that takes any number of lists and creates a 2-dimensional nested list of lists. For example, if you pass the following lists as arguments: **[1, 2, 3, 5], [1, 2, 3, 4], [1, 3, 4, 5]**.

Your code should return: **[[1, 2, 3, 5], [1, 2, 3, 4], [1, 3, 4, 5]]**

Day 25: All the Same

Create a function called ***all_the_same*** that takes one argument, a **string**, a **list,** or a **tuple** and checks if all the elements are the same. If the elements are the same, the function should return **True**. If not, it should return **False**. For example, **['Mary', 'Mary', 'Mary']** should return **True**.

Extra Challenge: Reverse a String

```
str1 = "the love is real"
```

Write a function called ***read_backwards*** that takes a string as an argument and reverses it. For example, the string above should return: **"real is love the"**

Day 26: Sort Words

Write a function called ***sort_words*** that takes a string of words as an argument, removes the whitespaces, and returns a list of letters sorted in alphabetical order. Letters will be separated by commas. All letters should appear once in the list. This means that you sort and remove duplicates. For example **'love life'** should return as **['e,f,i,l,o,v']**.

Extra Challenge: Length of Words

```
s = 'Hi my name is Richard'
```

Write a function called ***string_length*** that takes a string of words (words and spaces) as argument. This function should return the length of all the words in the string. Return the results in a form of a dictionary. The string above should return:

{'Hi': 2, 'my': 2, 'name': 4, 'is': 2, 'Richard': 7}

Day 27: Unique Numbers

Write a function called ***unique_numbers*** that takes a list of numbers as an argument. Your function is going to find all the unique numbers in the list. It will then sum up the unique numbers. You will calculate the difference between the **sum of all the numbers in the original list** and the **sum of unique numbers in the list**. If the difference is an **even** number, your function should return the **original list**. If the difference is an **odd** number, your function should return a **list with unique numbers only**. For example **[1, 2, 4, 5, 6, 7, 8, 8]** should return **[1, 2, 4, 5, 6, 7, 8, 8].**

Extra Challenge: Difference of two Lists

Write a function called ***difference*** that takes **two** lists as arguments. This function should return all the elements that are in list **a** but not in list **b** and all the elements in list **b** not in list **a. For example:**

list1 = [1, 2, 4, 5, 6]

list2 = [1, 2, 5, 7, 9]

should return:

[4, 6, 7, 9]

Use **list comprehension** in your function.

Day 28: Return Indexes

Write a function called ***index_position***. This function takes a string as a parameter and returns the positions or indexes of all lower letters in the string. For example, **'LovE'** should return **[1,2].**

Extra Challenge: Largest Number

Write a function called ***largest_number*** that takes a list of integers and re-arrange the individual digits to create the largest number possible. For example, if you pass the following as an argument: list1 = **[3, 67, 87, 9, 2]**. Your code should return the number in this exact format: **9,877,632** as the largest number.

Day 29: Middle Figure

Write a function called ***middle_figure*** that takes two parameters **a**, and **b**. The two parameters are strings. The function **joins** the two strings and finds the **middle element**. If the combined string has a middle element, the function should return the **element**, otherwise, return '**no middle figure'.** Use **'make love'** as an argument for **a** and **'not wars'** as an argument for **b.** Your function should return **'e'** as the middle element. Whitespaces should be removed.

Day 30: Most Repeated Name

Write a function called ***repeated_name*** that finds the most repeated name in the following list.

```
name = ["John", "Peter", "John", "Peter", "Jones", "Peter"]
```

Extra Challenge: Sort by Last Name

You work for a local school in your area. The school has a list of names of students saved in a list. The school has asked you to write a program that takes a list of names and sorts them alphabetically. The names should be sorted by last names. Here is a list of names:

['Beyonce Knowles, 'Alicia Keys', 'Katie Perry', 'Chris Brown',' Tom Cruise']

Your code should not just sort them alphabetically, but it should also switch the names (the last name must be the first). Here is how your code output should look:

['Brown Chris', 'Cruise Tom', 'Keys Alicia', 'Perry Katie', 'Knowles Beyonce']

Write a function called **sorted_names.**

Day 31: Longest Word

Write a function that has one parameter and takes a list of words as an argument. The function returns the longest word from the list. Name the function ***longest_word***. The function should return the longest word and the number of letters in that word. For example, if you pass **['Java, 'JavaScript', 'Python'],** your function should return
[10, JavaScript] as the longest word.

Extra Challenge: Create User

Write a function called ***create_user***. This function asks the user to enter their **name**, **age**, and **password**. The function saves this information in a dictionary. For example, if the use enters name as **Peter**, age - **18** and password - **love**. The function should save the information as: **{'name': 'Peter', 'age': 18, 'password': 'love'}**

Once the information is saved. The function should print to the user that - **"User saved. Please login"**

The function should then ask the user to **re-enter** their name and password. If the name and password match with what is saved in the dictionary, the function should return **"Logged in successfully".** If the name and password do not match with what is saved in the dictionary, the function should print **('Wrong password or user name. Please try again'.** The function should keep running until the user enters correct logging details.

Day 32: Password Validator

Write a function called ***password_validator***. The function asks the user to enter a password. A valid password should have at least **one upper letter**, **one lower letter**, and **one number**. It should not be less than **8 characters long**. When the user enters a password, the function should check if the password is **valid**. If the password is valid, the function should return the **valid password**. If the password is not valid, the function should tell the users the **errors** in the password and prompt the user to enter **another password**. The code should only stop once the user enters a valid password. (use while loop).

Extra Challenge: Valid Email

emails = ['ben@mail.com', 'john@mail.cm', 'kenny@gmail.com', '@list.com']

Write a function called ***email_validator*** that takes a list of emails and checks if the emails are valid. The function returns a **list** of only valid emails. A valid email address will have the following - **@ symbol** (if the @ sign is at the beginning of the name, the email is invalid. If there are more than one @signs, the email is invalid. All valid emails must have a **dot com at the end (.com)**, if not, the email is invalid. For example, the list of emails above should output the following as valid emails:

['ben@mail.com', 'kenny@gmail.com']

If no emails in the list are valid, the function should return 'all emails are invalid'

Day 33: List Intersection

Write a function called ***inter_section*** that takes **two** lists and finds the intersection (the elements that are present in both lists). The function should return a tuple of intersections. Use list comprehension in your solution. Use the lists below. Your function should return **(30, 65, 80)**.
list1 = [20, 30, 60, 65, 75, 80, 85]
list2 = [42, 30, 80, 65, 68, 88, 95]

Extra Challenge: Set or list

You want to implement a code that will search for a number in a range. You have a decision to make as to whether to store the number in a **set** or a **list**. Your decision will be based on time. You have to pick a data type that **executes faster**.

You have a range and you can either **store** it in a **set** or a **list** depending on which one executes faster when you are searching for **a number in the range**. See below:

```
a = range(10000000)
x = set(a)
y = list(a)
```

Let's say you are looking for a number 9999999 in the range above. Search for this number in the **list** and the **set**. Your challenge is to find which code executes faster. You will pick the one that executes quicker, lists, or sets. **Run the two searches and time them**.

Day 34: Just Digits

In this challenge, copy the **text below** and save it as a CSV file. Save it in the same folder as your Python file. Save it as **python.csv**.
Write a function called ***just_digits*** that reads the text from the CSV file and returns only digit elements from the file. Your function should return **1991, 2, 200, 3, 2008** as a list of strings.

"Python was released in 1991 for the first time. Python 2 was released in 2000 and introduced new features, such as list comprehensions and a cycle-detecting garbage collection system (in addition to reference counting). Python 3 was released in 2008 and was a major revision of the language that is not completely backward-compatible."

Source: Wikipedia

Day 35: Pangram

Write a function called ***check_pangram*** that takes a string and checks if it is a pangram. **A pangram is a sentence that contains all the letters of the alphabet**. If it is a pangram, the function should return **True**, otherwise, return **False**. The following sentence is a pangram so it should return **True**:

'the quick brown fox jumps over a lazy dog'

Extra Challenge: Find my Position

Write a function called ***find_index*** that takes two arguments; a list of integers, and an integer. The function should return the indexes of the integer in the list. If the integer is **not** in the list, the function should convert all the numbers in the list into the given integer.

For example, if the list is: **[1, 2, 4, 6, 7, 7]** and the integer is **7**, your code should return **[4, 5]** as the indexes of **7** in the list. If the list is [**1, 2, 4, 6, 7, 7]** and the integer is **8**, your code should return **[8, 8, 8, 8, 8, 8]** because **8** is not in the list.

Day 36: Count String

Write a function called ***count*** that takes one argument a **string,** and returns a **dictionary** of how many times each element appears in the string. For example, **'hello'** should return: **{'h':1,'e': 1,'l':2, 'o':1}.**

Day 37: Count the Vowels

Create a function called ***count_the_vowels***. The function takes one argument, a string, and returns the number of **vowels** in the string. For example, '**hello**' should return **2** vowels. If a vowel appears in a string more than once it should be **counted as one**. For example, **'saas'** should return 1 vowel. Your code should count lowercase and uppercase vowels.

Day 38: Guess a Number

Write a function called ***guess_a_number***. The function should ask a user to guess a randomly generated number. If the user guesses a higher number, the code should tell them that the guess is too high, if the user guesses low, the code should tell them that their guess is too low. The user should get a maximum of three guesses. When the user guesses right, the code should declare them a winner. After three wrong guesses, the code should declare them a loser.

Extra Challenge: Find Missing Numbers

list = [1, 2, 3, 5, 6, 7, 9, 11, 12, 23, 14, 15, 17,
18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 31]

Write a function called ***missing_numbers*** that takes a list of sequence of numbers and finds the missing numbers in the sequence. The list above should return:

[4, 8, 10, 13, 16, 29, 30]

Day 39: Password Generator

Create a function called ***generate_password*** that generates any length of password for the user. The password should have **a random mix of upper letters, lower letters, numbers, and punctuation symbols**. The function should ask the user how strong they want the password to be. The user should pick from - **weak, strong,** and **very strong**. If the user picks **weak**, the function should generate a 5**-character** long password. If the user picks **strong**, generate an **8-character** password and if they pick **very strong**, generate a **12-character** password.

Day 40: Pig Latin

Write a function called ***translate*** that takes the following words and translates them into pig Latin.

```
a = 'i love python'
```

Here are the rules:

1. If a word starts with a vowel (a,e, i, o, u) add ‘yay’ at the end. For example, ‘eat’ will become ‘eatyay’
2. If a word starts with anything other than a vowel, move the first letter to the end and add ‘ay’ to the end. For example, ‘day’ will become ‘ayday’.

Your code should return:
iyay ovelay ythonpay