Week 5 materials

topics covered

- lists
- list methods

some recap

- lists are used to store multiple data at once
- we create a list by placing elements inside `[]`, separated by commas

  `ages =  [19, 26, 23]`
- in a list we can store elements of different types
- in a list we can store duplicate elements

```
# list with elements of different data types
list1 = [1, "Hello", 3.4]

# list with duplicate elements
list1 = [1, "Hello", 3.4, "Hello", 1]

# empty list
list3 = []
```

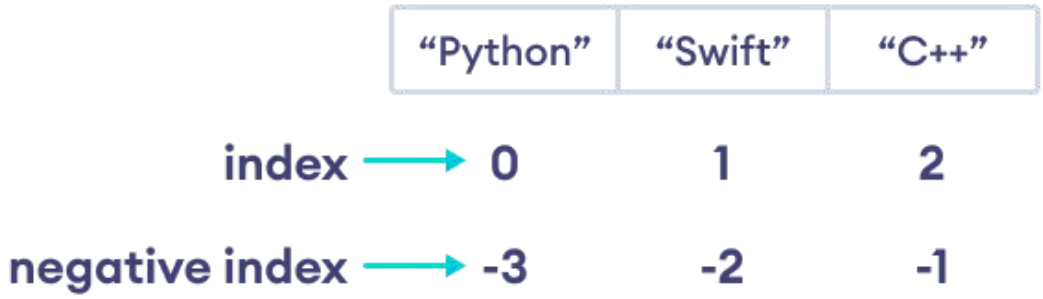
- lists are **ordered** and each item in a list is associated with a number. The number is known as a list **index**
- the index of the first element is **0**, second element is **1** and so on

```
languages = ["Python", "Swift", "C++"]

# access item at index 0
print(languages[0])   # Python

# access item at index 2
print(languages[2])   # C++
```

- Python allows negative indexing for its sequences. The index of **-1** refers to the last item, **-2** to the second last item and so on



- if the specified index does not exist in a list, Python throws the `IndexError` exception
- it is possible to access a portion of a list using the slicing operator `:`

```
# List slicing in Python
my_list = ['p','r','o','g','r','a','m','i','z']

# items from index 2 to index 4
# end position is NOT included!!!
print(my_list[2:5])

# items from index 5 to end
print(my_list[5:])

# items beginning to end
print(my_list[:])

# output:
# ['o', 'g', 'r']
```

```
# ['a', 'm', 'i', 'z']
# ['p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z']
```

you can read more on advanced list slicing here:
https://www.geeksforgeeks.org/python-list-slicing/

- lists are mutable (changeable). We can change items of a list by assigning new values using the `=` operator

```
languages = ['Python', 'Swift', 'C++']

# changing the third item to 'C'
languages[2] = 'C'

print(languages)  # ['Python', 'Swift', 'C']
```

- you can use `append()` method to add an item at the end of the list

```
numbers = [21, 34, 54, 12]
print("Before Append:", numbers)

# using append method
numbers.append(32)
print("After Append:", numbers)

# output:
# Before Append: [21, 34, 54, 12]
# After Append: [21, 34, 54, 12, 32]
```

- we can use the `del` statement to remove one or more items from a list:

```
languages = ['Python', 'Swift', 'C++', 'C', 'Java', 'Rust', 'R']

# deleting the second item
del languages[1]
print(languages) # ['Python', 'C++', 'C', 'Java', 'Rust', 'R']

# deleting the last item
del languages[-1]
print(languages) # ['Python', 'C++', 'C', 'Java', 'Rust']
```

```
# delete the first two items
del languages[0 : 2]  # ['C', 'Java', 'Rust']
print(languages)
```

- can also use the `remove()` method to delete the first matching element (which is passed as an argument) from the list:

```
languages = ['Python', 'Swift', 'C++', 'C', 'Java', 'Rust', 'R']

# remove 'Python' from the list
languages.remove('Python')

print(languages) # ['Swift', 'C++', 'C', 'Java', 'Rust', 'R']
```

- you can concatenate lists using `+` operator

```
first_list = [1, 2, 3]
second_list = [4, 5, 6]

# concatenating the two lists
concat_list = first_list + second_list

# print the concatenated list
print(concat_list)

# output: [1, 2, 3, 4, 5, 6]
```

- you can use the `*` operator to repeat a list a certain number of times

```
first_list = [1, 2, 3]
second_list = first_list * 3

print(second_list)

# output: [1, 2, 3, 1, 2, 3, 1, 2, 3]
```

- you can use the `in` keyword to check if an item exists in the list or not

```
languages = ['Python', 'Swift', 'C++']

print('C' in languages)    # False
print('Python' in languages)    # True
```

- use the `len()` function to find the size of a list

```
languages = ['Python', 'Swift', 'C++']
print("Total Elements:", len(languages))

# output:
# Total Elements: 3
```

some additional methods on lists:

- `index()` method returns the index of the specified element in the list

```
animals = ['cat', 'dog', 'rabbit', 'horse']

# get the index of 'dog'
index = animals.index('dog')
print(index)

# Output: 1
```

- `extend()` method adds all the elements of another list, tuple, string to the end of our list

```
# create a list
prime_numbers = [2, 3, 5]

# create another list
numbers = [1, 4]

# add all elements of prime_numbers to numbers
numbers.extend(prime_numbers)
print('List after extend():', numbers)

# Output: List after extend(): [1, 4, 2, 3, 5]
```

- `insert()` method inserts an element to the list at the specified index

```
# create a list of vowels
vowel = ['a', 'e', 'i', 'u']
# 'o' is inserted at index 3 (4th position)
vowel.insert(3, 'o')
print('List:', vowel)

# Output: List: ['a', 'e', 'i', 'o', 'u']
```

- `count()` method returns the number of times the specified element appears in the list

```
# create a list
numbers = [2, 3, 5, 2, 11, 2, 7]

# check the count of 2
count = numbers.count(2)
print('Count of 2:', count)

# Output: Count of 2: 3
```

- `pop()` method removes the item at the specified index. The method also returns the removed item

```
prime_numbers = [2, 3, 5, 7]

# remove the element at index 2
removed_element = prime_numbers.pop(2)

print('Removed Element:', removed_element)
print('Updated List:', prime_numbers)

# Output:
# Removed Element: 5
# Updated List: [2, 3, 7]
```

- `reverse()` method reverses the elements of the list

```
# create a list of prime numbers
prime_numbers = [2, 3, 5, 7]

# reverse the order of list elements
```

```
prime_numbers.reverse()
print('Reversed List:', prime_numbers)

# Output: Reversed List: [7, 5, 3, 2]
```

- `sort()` method sorts the items of a list in ascending order

```
prime_numbers = [11, 3, 7, 5, 2]
# sorting the list in ascending order
prime_numbers.sort()
print(prime_numbers)

# Output: [2, 3, 5, 7, 11]
```

- `copy()` method returns a shallow copy of the list

```
# mixed list
prime_numbers = [2, 3, 5]
# copying a list
numbers = prime_numbers.copy()
print('Copied List:', numbers)

# Output: Copied List: [2, 3, 5]
```

- `clear()` method removes all items from the list

```
prime_numbers = [2, 3, 5, 7, 9, 11]
# remove all elements
prime_numbers.clear()
# Updated prime_numbers List
print('List after clear():', prime_numbers)

# Output: List after clear(): []
```

exercises

Save each python program which you wrote any code for in this format: p5_exerciseNum.py e.g. p5_20.py

1. Create a list called `fruits` and store the names of three different fruits in it. Print the second fruit in the list.

2. Create a list called `numbers` and store 4 integers in it. Print the sum of the first and third numbers in the list.

3. Create a list called `mixed` with elements of different data types: an integer, a string, a boolean, and a float. Print the type of each element in the list.

4. Create a list called `grades` and store the following grades in it: 90, 85, 95, 80. Print the average of the grades. Use `len()` in this program. You can also use `sum()` function, for more info search online

5. Create a list called `colors` and store the names of three different colors in it. Print the last color in the list using negative indexing.

6. Create a list called `numbers` with integers from 1 to 10. Use slicing to print the numbers from index 2 to index 6 included.

7. Create a list called `letters` with 15 letters from the alphabet (any letters from 'A' to 'Z'). Use slicing to print the letters from index 7 to the end.

8. Create a list called `words` with the words "apple", "banana", "cherry", "date", "watermelon". Use slicing to print the words from index 1 to index 3 included.

9. Create a list called `names` with 5 names of your friends or relatives. Use slicing to print the names from the beginning to index 2 included.

10. Create a list called `animals` with the names of 4 different animals. Use slicing to print the animals from index 1 to the end.

11. Create a list called `groceries` with the items "Milk", "Eggs", "Bread", "Bananas". Change the last element to 'Tomatoes' using assignment.

12. You have a list of students in a class, called `students` (you can write any names there). Two new students have joined the class. Use the `append()` method to add the new students' names to the existing list. Print the updated list.

13. Given the following list, use the `del` statement to remove the last two elements:
    `items = ['Apple', 'Banana', 'Orange', 'Mango', 'Grapes']`

14. Remove the element 'Java' from the following list using the `remove()` method:
    `programming_languages = ['Python', 'JavaScript', 'Java', 'C++']`

15. Create a list called `numbers` with the elements 1, 2, 3, 4, 5. Use slicing to remove the second to fourth elements from the list(use `del`). Print the updated list.

16. Create two lists called `fruits` and `vegetables` with some fruit and vegetable names. Concatenate the two lists and store the result in a new list called `groceries`. Print the `groceries` list.

17. Create a list called `song` with the lyrics of your favorite song chorus (each line is a different element in a list). Use the `*` operator to repeat the chorus three times. Print the repeated chorus.

18. Create a program that stores several programming languages in a list. Prompt the user to enter their favorite programming language and check if it exists in the list. Print a message indicating whether it is in the list or not. Additionally, print number of languages in a list using `len()`

19. Imagine you have a shopping list stored as a list of items. Ask the user to enter an item. Check if the item is in the shopping list and print a message indicating whether the item is already on the list or not.

20. Create a program for a student management system. Create a list of students' IDs. Ask the user to enter a student ID. Check if the ID exists in the database of students and print a message indicating whether the student is registered or not.

21. Design a program for a recipe app that allows users to search for recipes by ingredient. Ask the user to enter an ingredient and check if it exists in the recipe database. Print a message indicating whether recipes containing that ingredient are available or not.

22. Write a program that simulates a bookshelf. Create a list of book titles. Prompt the user to enter a book title and use the `index()` function to find the index of the book in the list. If the book is found, print its index and a message indicating that it is in your bookshelf. If not, print a message saying that the book is not in your collection.

23. Imagine you have a list of cities you want to visit. Ask the user to enter the name of a city they want to visit. Use the `index()` function to find the index of the city in your list. If the city is found, print its index and a message indicating that you also want to visit that city. If not, print a message saying that you do not want to visit that city yet.

24. Write a program that keeps track of your favorite movies. Create a list with some of your favorite movies already included. Prompt the user to enter several new movies they have recently watched. Use the `extend()` function to add the user's movies to the existing list. Print the updated list of favorite movies.

25. Write a program that manages a playlist of songs. Create a list with some songs already included. Ask the user to enter a few new songs they have discovered. Use the `extend()` function to add the user's songs to the existing list. Print the updated playlist.

26. Create a program that manages a to-do list. Allow the user to add a certain amount of tasks to the list using the `insert()` method. Prompt the user to enter the task and the index at which the task should be inserted. Print the updated to-do list.

27. Create a program that stores a list of student names in a list. Ask the user to enter a name and use the `count()` method to count the number of times the name appears in the list. Print the count. If name is not in the list, print appropriate message

28. You have a list of countries. Use the `pop()` method to remove the last country from the list and store it in a variable. Print the updated list and the removed country.

29. Imagine you have a list of ingredients for a recipe. Initialize a list with the ingredients in the order they are listed in the recipe. Use the `reverse()` function to display the ingredients in reverse order, starting from the last ingredient and ending with the first ingredient.

30. You have a list of books on your bookshelf. Use the `sort()` method to sort the books in alphabetical order. Print the sorted list.

EXTRA TASK:
Create a program to manage a guest list and invitations for a party. Initialize an empty list to store the names of invited guests. Ask the name of the user. Ask the user to enter 5 names of guests they want to invite.

You can append each name to the guest list `append()` or you can extend the list `extend()`. After collecting all the names, ask the user to enter date and time of the party. Then ask for location. Ask if guests need to wear some costumes, if yes, then ask the user the theme of the party.

Finally, display the complete guest list.

And ask user to enter a number of a guest they want to create an invitation for. Create a complete invitation for that guest on the list, use all information that you have. You can extend this program in any way you want, however all of the data above is mandatory.

This is just an example output, you can do it in your own manner!

```
...

Guest List:
1. Alice
2. Bob
3. Claire
4. David
5. Emma

Enter a number of the guest you want to create an invitation for: 3

Invitation for the guest:
--------------------------
Dear Claire,
You are invited to the party on Saturday, July 15th at 7:00 PM at 123 Main Street
Theme: Superheroes
```