Loop

Definition: Repeats a block of code based on a condition.

Example:

```
# Looping until a condition is False
count = 5
while count > 0:
    print(count)
    count -= 1  # Decrement count
```

Conditional Statements (if, elif, else)

Definition: Executes different blocks of code based on certain conditions.

Example:

```
age = 20
if age >= 18:
    print("Adult")
else:
    print("Not an adult")
```

Boolean Values (True, False)

Definition: Represents truthiness in logic.

Example:

```
is_valid = True
if is_valid:
    print("The condition is True")
```

break Statement

Definition: Exits the nearest enclosing loop immediately.

Example:

```
# Exiting the loop when i equals 3
for i in range(5):
    if i == 3:
        break
    print(i)
```

input() Function

Definition: A function that allows the user to input data from the keyboard as a string.
Example:
```
name = input("Enter your name: ")
```

continue Statement

Definition: Skips the rest of the code inside the loop for the current iteration.
Example:
```
# Skipping even numbers
for i in range(6):
    if i % 2 == 0:
        continue
    print(i)  # This will print only odd numbers
```

range() Function

Definition: Generates a sequence of numbers.
Example:
```
 for i in range(0, 5):
    print(i)  # Prints numbers from 0 to 4
```

Use of .lower() in Conditions

Definition: Converts strings to lowercase to make case-insensitive comparisons.
Example:
```
answer = "Yes"
if answer.lower() == "yes":
    print("Answer matched")
```

Use of .upper()

Definition: Converts all the characters in a string to uppercase.
Example
```
text = "Hello World"
print(text.upper())  # Output: "HELLO WORLD"
```

strip()

Definition: Removes leading and trailing whitespace (spaces, tabs, newlines) from a string.

Example

```
text = "  Hello World  \n"
print(text.strip())  # Output: "Hello World"
```

startswith()

Definition: Checks if a string starts with a specified prefix or substring.

Example

```
color = "Blue"
print(color.startswith("B"))  # Output: True
print(color.startswith("C"))  # Output: False
```

open() Function with 'r'

- **Definition: Opens a file in read-only mode, allowing you to read its content.**

Example

```
with open('Colors.txt', 'r') as file:
    content = file.read()
    print(content)
```

open() Function with 'w'

Definition: Opens a file for writing only. If the file exists, it will be truncated (cleared). If the file does not exist, a new file will be created.

```
with open('Colors.txt', 'w', encoding='utf-8') as file:

    file.write("Azure\nLemon Chiffon\nHoneydew")
```

List

- **Definition: A collection of ordered items (elements) that can be of different data types (e.g., strings, numbers). Lists are mutable, meaning they can be modified after creation (add, remove, change elements).**
- **Example**

```
movie_list = [
    ["Monty Python and the Holy Grail", 1975],
    ["On the Waterfront", 1954],
    ["Cat on a Hot Tin Roof", 1958]
]
```

Len() Function

- **Definition: A built-in Python function that returns the number of elements in a list, string, dictionary, or other collection.**
- **Example**

```
total_movies = len(movie_list)
print(f"Total movies: {total_movies}")
```

UTF-8 Encoding in File Operations

Definition:
Specifies the character encoding to be used when opening a file. UTF-8 is a universal character encoding standard that supports all characters from all scripts used globally.

```
# Reading a file with UTF-8 encoding

with open('Colors.txt', 'r', encoding='utf-8') as file:

    content = file.read()

    print(content)

with open('InternationalColors.txt', 'w', encoding='utf-8') as file:

    file.write("Café\nNiçoise\nJalapeño")
```

return

- **Definition: Exits a function and sends a value back to where the function was called.**

Example

```
def add(a, b):
    return a + b

result = add(3, 4)
print(result)  # Output: 7
```

Exponentiation (**)

Definition: An operation that raises one number to the power of another.

Example:

```
# Squaring a number
result = 4 ** 2
print(result)  # Outputs 16
```

Function Definition (def)

Definition: The `def` keyword is used to define a function in Python. A function is a reusable block of code that performs a specific task.

Example:

```
# Defining a function that greets a user
def greet(name):
    print(f"Hello, {name}!")

# Calling the function
greet("Stone")
```

sorted()

Definition: Sorts the items of an iterable (such as a list) in a specific order (default is ascending) and returns a new sorted list.

Example:

```
numbers = [3, 1, 4, 1, 5, 9, 2]
sorted_numbers = sorted(numbers)  # Default is ascending order
print(sorted_numbers)  # Output: [1, 1, 2, 3, 4, 5, 9]
```

reverse=True

Definition: An optional argument used with sorting functions like `sorted()` to reverse the natural order of the elements, sorting them from high to low.

Example:

```
numbers = [3, 1, 4, 1, 5, 9, 2]
sorted_numbers_descending = sorted(numbers, reverse=True)
print(sorted_numbers_descending)  # Output: [9, 5, 4, 3, 2, 1, 1]
```

count_consonants()

Definition: A user-defined function that counts the number of consonant characters in a string.

Example:

```
def count_consonants(text):
    consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"
    return sum(1 for char in text if char in consonants)

state = "California"
print(count_consonants(state))  # Output: 7
```

try and except FileNotFoundError:

Definition: `try` is used to wrap a block of code that might throw an error, while `except FileNotFoundError:` catches the `FileNotFoundError` thrown when attempting to open a file that does not exist.

Example:

```
try:
    with open('non_existent_file.txt', 'r') as file:
        content = file.read()
except FileNotFoundError:
    print("The file was not found.")  # Output: The file was not found.
```

isalpha()

Definition: Checks if all characters in a string are alphabetic and returns `True` if so.

Example:

```
text = "Hello123"
result = text.isalpha()  # Checks if all characters are alphabetic
print(result)  # Output: False
```

choice Variable

Definition:

In the context of your script, `choice` is a variable used to control a while loop based on user input. It typically stores a string that the user enters to decide whether to continue or terminate the loop (and often the program)

```
choice = "y"
while choice.lower() == "y":
    print("Hello, let's do something!")
    choice = input("Continue? (y/n): ")
print("Goodbye!")
```

- **Syntax Error** – occurs when the code violates the grammatical rules of the programming language, leading to failure in compilation or interpretation
- **Runtime Error** – happens during the execution of a program, disrupting normal operations and often causing the program to terminate unexpectedly
- **Logic Error** – arises when the program runs without crashing but produces incorrect results due to mistakes in the program's logic or algorithm