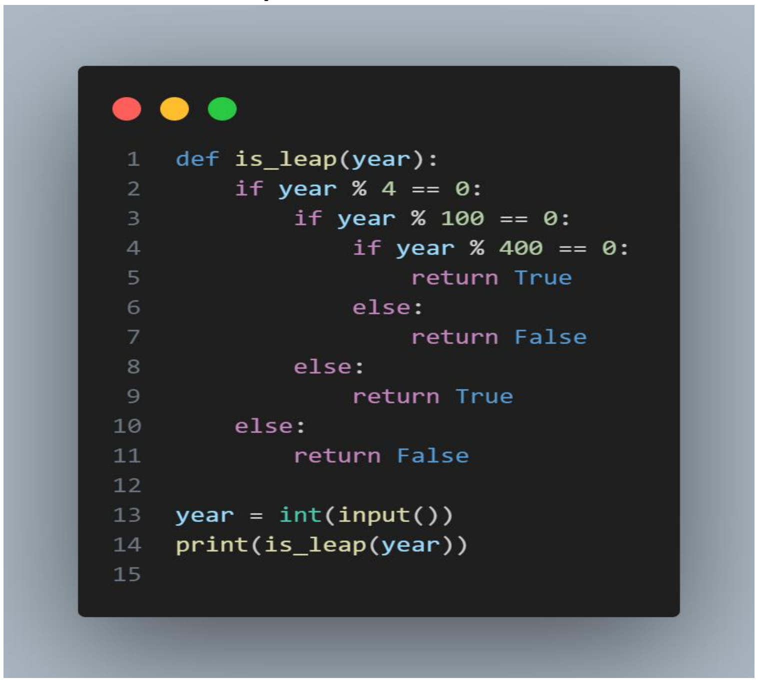
Functions:

✓ Functions are blocks of reusable code that perform a specific task.

✓ They help in organizing code, making it more readable, and promoting code re-usability.

✓ Functions typically take input parameters, perform operations, and may return output.



```python
def is_leap(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return True
            else:
                return False
        else:
            return True
    else:
        return False

year = int(input())
print(is_leap(year))

```

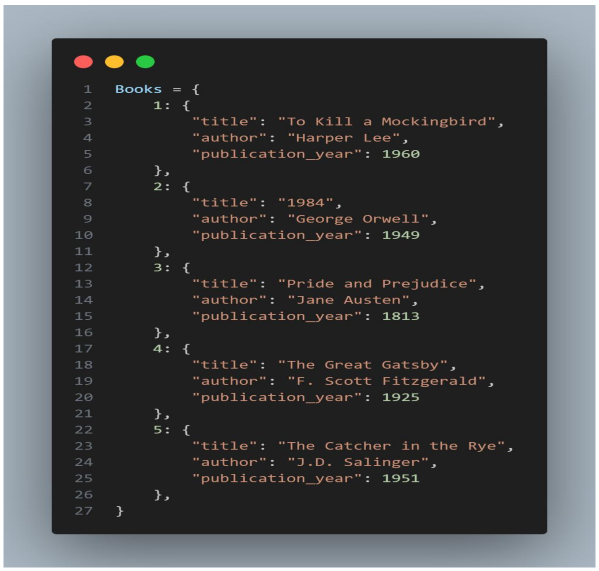
Data Structures:

✓ Lists: Ordered collections of items that can be of any data type. Lists are mutable, meaning their elements can be changed after creation.

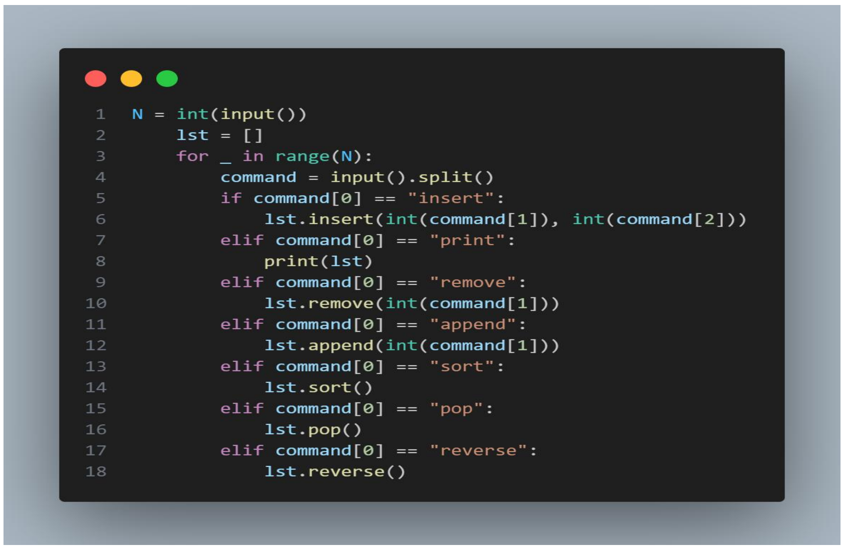
✓ Dictionary: Key-value pairs where each key is unique and associated with a value. Dictionaries are mutable.

✓ Tuple: Similar to lists but immutable, meaning their elements cannot be changed after creation.

✓ Sets: Unordered collections of unique items. Sets do not allow duplicate elements.



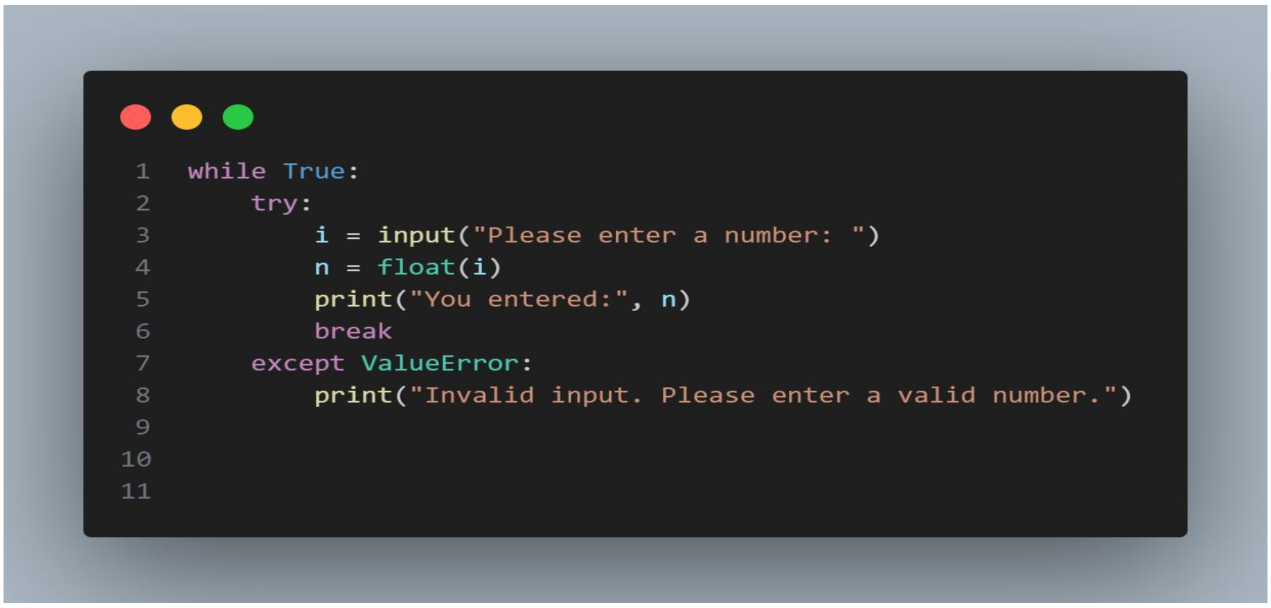
```python
Books = {
    1: {
        "title": "To Kill a Mockingbird",
        "author": "Harper Lee",
        "publication_year": 1960
    },
    2: {
        "title": "1984",
        "author": "George Orwell",
        "publication_year": 1949
    },
    3: {
        "title": "Pride and Prejudice",
        "author": "Jane Austen",
        "publication_year": 1813
    },
    4: {
        "title": "The Great Gatsby",
        "author": "F. Scott Fitzgerald",
        "publication_year": 1925
    },
    5: {
        "title": "The Catcher in the Rye",
        "author": "J.D. Salinger",
        "publication_year": 1951
    },
}
```



```python
N = int(input())
    lst = []
    for _ in range(N):
        command = input().split()
        if command[0] == "insert":
            lst.insert(int(command[1]), int(command[2]))
        elif command[0] == "print":
            print(lst)
        elif command[0] == "remove":
            lst.remove(int(command[1]))
        elif command[0] == "append":
            lst.append(int(command[1]))
        elif command[0] == "sort":
            lst.sort()
        elif command[0] == "pop":
            lst.pop()
        elif command[0] == "reverse":
            lst.reverse()
```

Error Handling:

- ✓ Error handling allows you to gracefully deal with unexpected errors that may occur during program execution.
- ✓ Python provides try, except, else, and finally blocks for error handling.
- ✓ try: Code block where you anticipate errors.
- ✓ except: Code block to handle specific types of errors.
- ✓ else: Code block to execute if no exceptions occur.
- ✓ finally: Code block that executes regardless of whether an exception occurred.

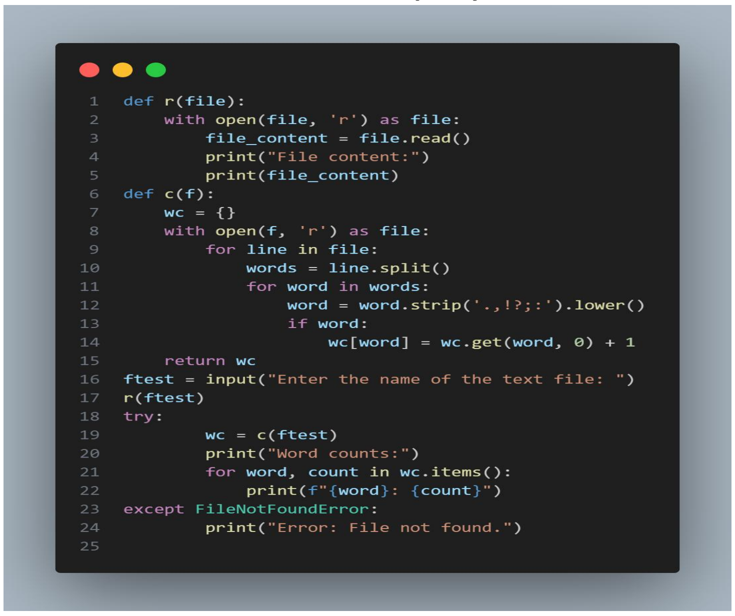


```python
while True:
    try:
        i = input("Please enter a number: ")
        n = float(i)
        print("You entered:", n)
        break
    except ValueError:
        print("Invalid input. Please enter a valid number.")


```

Files Input/Output:

- ✓ Python provides built-in functions to work with files, such as open(), read(), write(), close(), etc.
- ✓ Use open() to open a file and specify the mode ('r' for read, 'w' for write, 'a' for append, 'r+' for read/write).
- ✓ Always close files after use to free up system resources.

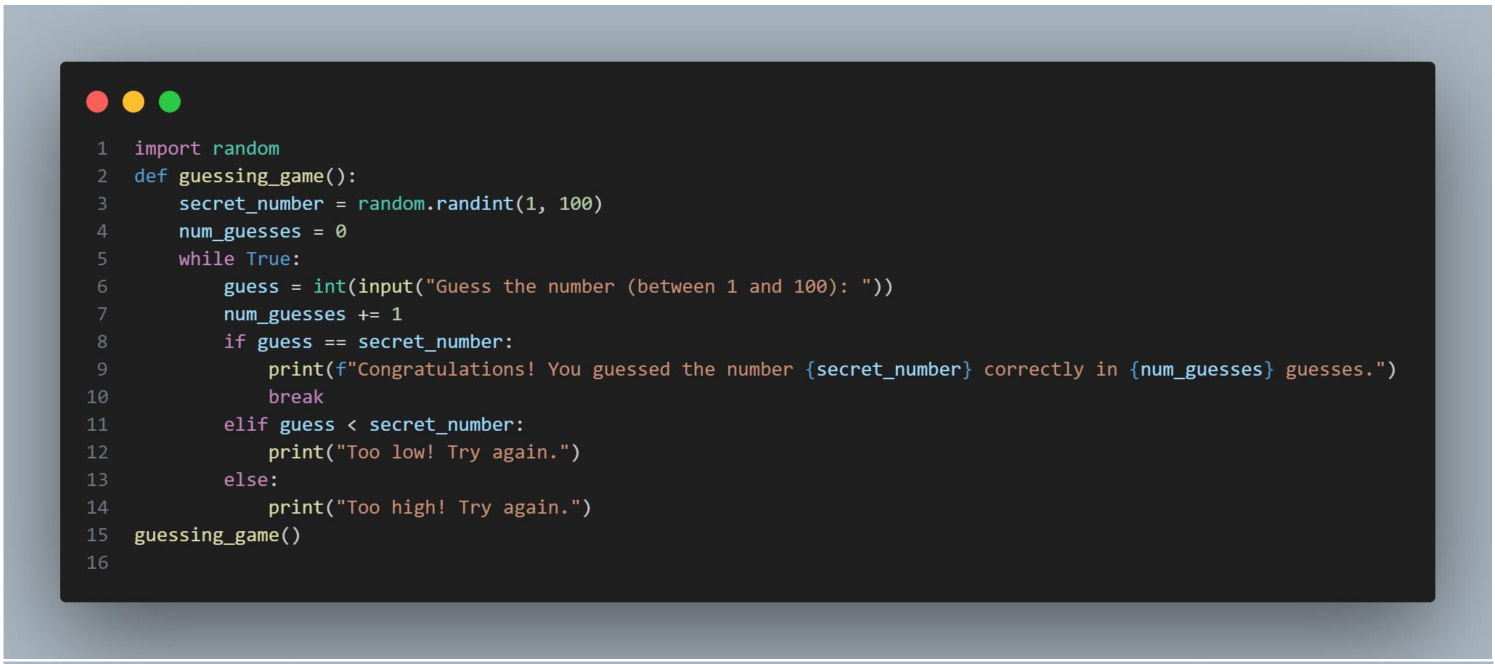


```python
def r(file):
    with open(file, 'r') as file:
        file_content = file.read()
        print("File content:")
        print(file_content)
def c(f):
    wc = {}
    with open(f, 'r') as file:
        for line in file:
            words = line.split()
            for word in words:
                word = word.strip('.,!?;:').lower()
                if word:
                    wc[word] = wc.get(word, 0) + 1
    return wc
ftest = input("Enter the name of the text file: ")
r(ftest)
try:
        wc = c(ftest)
        print("Word counts:")
        for word, count in wc.items():
            print(f"{word}: {count}")
except FileNotFoundError:
        print("Error: File not found.")

```

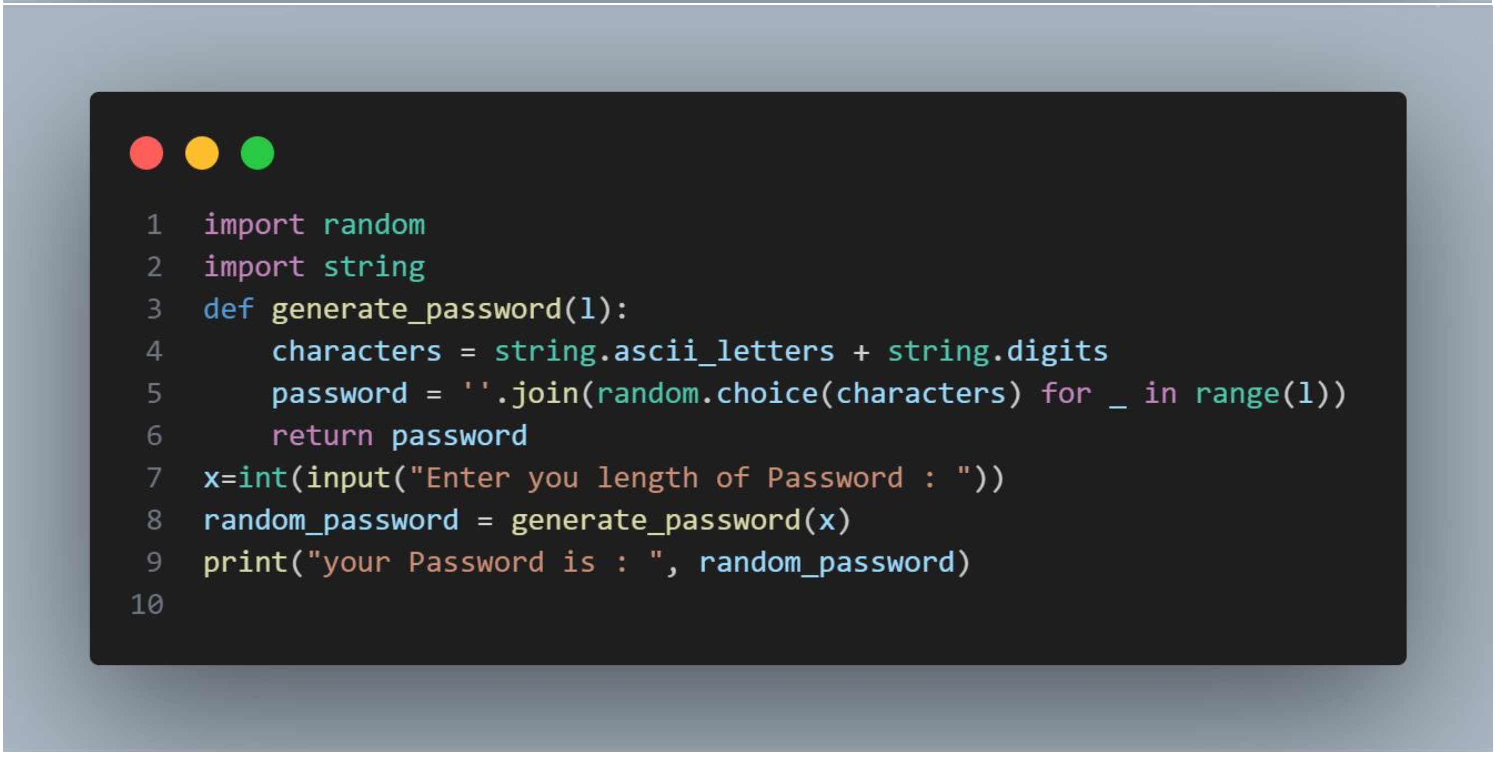
Random Numbers:

- ✓ Python's random module provides functions for generating random numbers and selecting random items.
- ✓ Commonly used functions include random(), randint(), choice(), shuffle(), etc.



```python
import random
def guessing_game():
    secret_number = random.randint(1, 100)
    num_guesses = 0
    while True:
        guess = int(input("Guess the number (between 1 and 100): "))
        num_guesses += 1
        if guess == secret_number:
            print(f"Congratulations! You guessed the number {secret_number} correctly in {num_guesses} guesses.")
            break
        elif guess < secret_number:
            print("Too low! Try again.")
        else:
            print("Too high! Try again.")
guessing_game()

```



```python
import random
import string
def generate_password(l):
    characters = string.ascii_letters + string.digits
    password = ''.join(random.choice(characters) for _ in range(l))
    return password
x=int(input("Enter you length of Password : "))
random_password = generate_password(x)
print("your Password is : ", random_password)

```