Data types:

1. `**str**` (short for string) - represents text, such as "hello", "world", "python", etc.

2. ‘**list**’ - represents an ordered collection of values. Lists are mutable, which means you can change their contents after they are created. You can create a list by enclosing a sequence of values in square brackets, separated by commas. For example, `['a', 'b', 'c']' is a list of three strings.

Functions:

1. ‘**input()**` - This function is used to get input from the user. It takes one argument, which

is a string that is used as a prompt to the user. The function returns a string.

2. `**print()**` - This function is used to display output to the user. It takes one or more arguments, which can be strings, numbers, or variables. The values are separated by commas, and the function automatically adds a space between each value.

3. `**len()**` - returns the length of a string or a list. For example, `len("hello")` returns `5 and `len([1, 2, 3])` returns `3`.

4. `**range()**` - returns a sequence of numbers. You can use it to create a loop that runs a specific number of times. For example, `range (5)` returns `[0, 1, 2, 3, 4]`.

Operators:

1 ‘**+**’ - concatenates two strings or two lists. For example, "hello" + "world"` returns "hello” + “world"` and `[1, 2] + [3, 4]` returns `[1, 2, 3, 4]`.

2. `**in**` - checks if an item is in a list or a string. For example, `"a" in "apple" returns `true` and `2 in [1, 2, 3]` returns `True`.

3. ‘**=**’ - assigns a value to a variable. For example, `x = 5' assigns the value *5* to the variable `x`.

4. `**==**` - checks if two values are equal. For example, `x == 5` returns `True` if `x`is equal to '5'.

Built-in functions:

1. ‘**random.choice()**` - returns a random item from a list. You can use this function to

choose a random word for the Hangman game. For example, `random.choice(["apple", "banana", "cherry"])` returns a random fruit.

2. `**str.join()**`-joins a sequence of strings into a single string. You can use this function to display the letters that the user has guessed so far. For example, `" ".join(["a", "p", "p", "1", "e"])` returns `"a p p 1 e"".

3. `**set()**` - creates a set, which is an unordered collection of unique elements. You can use this function to keep track of the letters that the user has guessed so far. For example, `set(["a", "b", "a", "c"])`returns `set(["a", "b", "c"])`.

If statements

Imports: Python allows you to import external modules into your program using the

- `**import**` statement. Modules are simply files containing Python definitions and statements. When you import a module, you gain access to all the functions, classes, and variables defined in that module. In the example Hangman game I provided earlier, we used the ‘**random**’ module to choose a random word from a list of words.
- `**if**` statements: `**if**` statements are used to test conditions and execute different code based on the outcome of the test. The basic syntax of an `**if**` statement is:

In the Hangman game, we used `if` statements to check if the user's guess was already in the set of guessed letters, and to check if the user's guess was in the word.

- ‘**while loops’** - while loops are used to repeat a block of code as long as a certain condition is True. The basic syntax of a `**while**` loop is:

In the Hangman game, we used a `while loop to keep getting guesses from the user until they either guessed the whole word or ran out of guesses.

An example will be in **hangmanExample.py.**