Strings:

1. `len()`: returns the length of the string.
2. `find()`: returns the index of the first occurrence of the substring in the string. Returns -1 if the substring is not found.
3. `count()`: returns the number of occurrences of the substring in the string.
4. `partition()`: splits the string into three parts: the part before the first occurrence of the substring, the substring, and the part after the substring. Returns a tuple of these three parts.
5. `upper()`: returns a new string with all uppercase characters.
6. `lower()`: returns a new string with all lowercase characters.
7. `swapcase()`: returns a new string with all uppercase characters converted to lowercase and all lowercase characters converted to uppercase.
8. `title()`: returns a new string with the first letter of each word capitalized.
9. `capitalize()`: Capitalizes the first character of a string.
10. `reversed()`: Returns a reverse iterator of a string. To get the reversed string, use `''.join(reversed(string))`.
11. `strip()`: Removes leading and trailing whitespaces from a string.
12. `lstrip()`: Removes leading whitespaces from a string.
13. `rstrip()`: Removes trailing whitespaces from a string.
14. `replace()`: Replaces a substring in a string with another string.
15. `isspace()`: Returns `True` if all characters in a string are whitespaces, `False` otherwise.
16. `isupper()`: Returns `True` if all characters in a string are uppercase, `False` otherwise.
17. `islower()`: Returns `True` if all characters in a string are lowercase, `False` otherwise.
18. `startswith()`: Returns `True` if a string starts with a specified prefix, `False` otherwise.
19. `endswith()`: Returns `True` if a string ends with a specified suffix, `False` otherwise.
20. `isalnum()`: Returns `True` if all characters in a string are alphanumeric (letters and digits), `False` otherwise.
21. `isdigit()`: Returns `True` if all characters in a string are digits, `False` otherwise.
22. `isalpha()`: Returns `True` if all characters in a string are letters, `False` otherwise.

Usage of String Functions:

```
string = "    Hello World!  "

print("Length of the string: ", len(string))

substring = "Hello"
print("First occurrence of substring '{}' at index: {}".format(substring, string.find(substring)))
print("Number of occurrences of substring '{}': {}".format(substring, string.count(substring)))

before, sep, after = string.partition("World")
print("Partition of 3 blocks",before,sep,after)

print("Uppercase: ", string.upper())
print("Lowercase: ", string.lower())
print("Swapcase: ", string.swapcase())
print("Title: ", string.title())
print("Capitalize: ", string.capitalize())

print("Reversed: ", ''.join(reversed(string)))

print("Strip: ", string.strip())
print("Left Strip: ", string.lstrip())
print("Right Strip: ", string.rstrip())

replacement_string = string.replace("Hello", "Hi")
print("Replaced 'Hello' with 'Hi': ", replacement_string)

print("Is all whitespace? ", string.isspace())
print("Is all uppercase? ", string.isupper())
print("Is all lowercase? ", string.islower())
print("Starts with 'Hello'? ", string.startswith("Hello"))
print("Ends with '!'? ", string.endswith("!"))
print("Is alphanumeric? ", string.isalnum())
print("Is digit? ", string.isdigit())
print("Is alphabet? ", string.isalpha())
```

Outputs:

```
Length of the string:  17
First occurrence of substring 'Hello' at index: 5
Number of occurrences of substring 'Hello': 1
Partition of string: '    Hello ' | 'World' | '!  '
Uppercase:  HELLO WORLD!
Lowercase:  hello world!
Swapcase:  HELLO wORLD!
Title:  Hello World!
Capitalize:  Hello world!
Reversed: !  dlroW olleH
Strip: Hello World!
Left Strip: Hello World!
Right Strip:     Hello World!
Replaced 'Hello' with 'Hi':     Hi World!
Is all whitespace?  False
Is all uppercase?  False
Is all lowercase?  False
Starts with 'Hello'?  False
Ends with '!'?  True
```

```
Is alphanumeric?  False
Is digit?  False
Is alphabet?  False
```

Lists:

A list is a collection of items in a ordered sequence, which can be of different data types (integer, string, float, etc.). Lists are defined using square brackets `[]` and items are separated by commas `,`.

- `min` is a built-in function in Python that returns the minimum value from a list or any iterable.
- `max` is a built-in function in Python that returns the maximum value from a list or any iterable.
- `sort` is a method in Python that can be used to sort a list in ascending or descending order. By default, the `sort` method sorts the list in ascending order.
- `reverse` is a method in Python that can be used to reverse the order of elements in a list.

  descending reverse: To sort a list in descending order, the `sort` method can be used with the argument `reverse=True`.
- `append` is a method in Python that can be used to add an element to the end of a list.
- `extend` is a method in Python that can be used to add elements from one list to another.

```
# Initializing a list
numbers = [3, 4, 1, 8, 5]

# Using the min function
min_value = min(numbers)
print("Min Value:", min_value)

# Using the max function
max_value = max(numbers)
print("Max Value:", max_value)

# Using the sort method
numbers.sort()
print("Sorted List:", numbers)

# Using the reverse method
numbers.reverse()
print("Reversed List:", numbers)

# Using the sort method with reverse argument
numbers.sort(reverse=True)
print("Sorted List in Descending Order:", numbers)

# Using the append method
numbers.append(9)
print("List after appending 9:", numbers)

# Using the extend method
```

```
new_numbers = [10, 11, 12]
numbers.extend(new_numbers)
print("List after extending with new numbers:", numbers)
```

Outputs:

```
Min Value: 1
Max Value: 8
Sorted List: [1, 3, 4, 5, 8]
Reversed List: [8, 5, 4, 3, 1]
Sorted List in Descending Order: [8, 5, 4, 3, 1]
List after appending 9: [8, 5, 4, 3, 1, 9]
List after extending with new numbers: [8, 5, 4, 3, 1, 9, 10, 11, 12]
```

List Comprehensions:

List comprehensions are a concise way to create new lists by transforming existing ones. They are written as a single line of code and can include an optional condition.

```
# Using List Comprehensions
numbers=[8, 5, 4, 3, 1, 9, 10, 11, 12]
squared_numbers = [x**2 for x in numbers]
print("Squared Numbers:", squared_numbers)
```

Output:

```
Squared Numbers: [64, 25, 16, 9, 1, 81, 100, 121, 144]
```