In Python, slicing is a way to extract a portion of a string (or any sequence like lists and tuples) using the slice syntax: string[start:stop:step]. Here's how it works:

1. Basic Syntax:

string[start:stop:step]

- start: The index where the slice starts (inclusive). If omitted, it defaults to 0.
- stop: The index where the slice ends (exclusive). If omitted, it defaults to the length of the string.
- step: The amount by which the index increases. If omitted, it defaults to 1.

2. Examples:

- Extract a substring:

```
text = "Hello, World!"
print(text[0:5])  # Output: "Hello"
```

This extracts characters from index 0 to 4.

- Omitting start and stop:

```
text = "Hello, World!"
print(text[:5])   # Output: "Hello" (equivalent to text[0:5])
print(text[7:])   # Output: "World!" (equivalent to text[7:len(text)])
print(text[:])    # Output: "Hello, World!" (equivalent to text[0:len(text)])
```

- Using step:

```
text = "Hello, World!"
print(text[::2])  # Output: "Hlo ol!" (every second character)
print(text[1::2]) # Output: "el,Wrd" (every second character, starting from index 1)
```

- Negative Indices:

```
text = "Hello, World!"
print(text[-6:-1])  # Output: "World"
```

Here, -1 refers to the last character, -2 to the second last, and so on.

- Reversing a String:

```
text = "Hello, World!"
print(text[::-1])  # Output: "!dlroW ,olleH"
```

This uses a step of -1 to reverse the string.

3. Important Rules:

- If start is greater than stop, the result will be an empty string unless a negative step is used.
- If the step is omitted or set to 1, the slice progresses normally from start to stop-1.
- Slicing doesn't modify the original string; it creates a new string.

4. Edge Cases:

- If start is greater than or equal to the length of the string, or if stop is less than 0, the result is an empty string.
- If step is 0, it will raise a ValueError because a step of zero is not valid.

These slicing techniques are very powerful and are used frequently in string manipulation and analysis.