SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

19AMB302-FULL STACK AI

M.POORNIMA DEVI,AP/AIML

STRING FORMATTING

1.Python String Formatting – How to format String?

•String formatting allows you to create dynamic strings by combining variables and values.

• In this article, we will discuss about 5 ways to format a string.

•You will learn different methods of string formatting with examples for better understanding.

Let’s look at them now!

2.How to Format Strings in Python

There are five different ways to perform string formatting in Python

1.Formatting with % Operator.

2.Formatting with format() string method.

3.Formatting with string literals, called f-strings.

4.Formatting with String Template Class

5.Formatting with center() string method.

3. How to Format String using % Operator

It is the oldest method of string formatting. Here we use the modulo [% operator](). The modulo **%** is also known as the “string-formatting operator”.

Python Format String Using the % Operator

In the expression “The mangy, scrawny stray dog %s gobbled down” % ‘hurriedly’, the %s placeholder within the string is replaced by the value ‘hurriedly’.

EXAMPLE:

```
print("The mangy, scrawny stray dog %s gobbled down"
%'hurriedly' +
          "the grain-free, organic dog food.")
```

OUTPUT:

```
The mangy, scrawny stray dog hurriedly gobbled downthe grain-free, organic dog food.
```

4.Injecting Multiple Strings using the modulo Operator

Here we are inserting multiple strings with the % operator.

EXAMPLE:

```
x = 'looked'
print("Misha %s and %s around"%('walked',x))
```

Output: Misha walked and looked around

2. How to Format String using format() Method

•Formatters work by putting in one or more replacement fields and placeholders defined by a pair of curly braces { } into a string and calling the **str.format()**.

•The value we wish to put into the placeholders and concatenate with the string passed as parameters into the format function.
•**Syntax:** ‘String here {} then also {}’.format(‘something1′,’something2’)
EXAMPLE:
print('We all are {}.'.format('equal'))

OUTPUT
We all are equal.

1.Index-based Insertion
In this code, curly braces {} with indices are used within the string ‘{2} {1} {0}’ to indicate the positions where the corresponding values will be placed.

2.Insert object by Assigning Keywords

In this code, curly braces {} with named placeholders ({a}, {b}, {c}) are used within the string ‘a: {a}, b: {b}, c: {c}’ to indicate the positions where the corresponding named arguments will be placed.

3.Reuse the inserted objects

In this code, curly braces {} with named placeholders ({p}) are used within the string ‘The first {p} was alright, but the {p} {p} was tough.’ to indicate the positions where the corresponding named argument p will be placed.

Float Precision with the.format() Method

Syntax: {[index]:[width][.precision][type]}

The type can be used with format codes:

‘d’ for integers

‘f’ for floating-point numbers

‘b’ for binary numbers

‘o’ for octal numbers

‘x’ for octal hexadecimal numbers

‘s’ for string

‘e’ for floating-point in an exponent format

3. Understanding Python f-string

1. String Formatting with F-Strings

In this code, the f-string **f"My name is {name}."** is used to interpolate the value of the name variable into the string.

EXAMPLE:

```
name = 'Ele'
print(f"My name is {name}.")
```

2.Arithmetic operations using F-strings

In this code, the f-string **f" He said his age is {2 * (a + b)}."** is used to interpolate the result of the expression 2 * (a + b) into the string.

EXAMPLE:

```
a = 5
b = 10
print(f"He said his age is {2 * (a + b)}.")
```

OutputHe said his age is 30.

3.Lambda Expressions using F-strings

In this code, an anonymous lambda function is defined using lambda x: x*2. This lambda function takes an argument x and returns its double.

EXAMPLE:

```
print(f"He said his age is {(lambda x: x*2)(3)}")
```

Output

```
He said his age is 6
```

4.Float precision in the f-String Method

In this code, f-string formatting is used to interpolate the value of the num variable into the string.

Syntax: {value:{width}.{precision}}

EXAMPLE:

```
num = 3.14159
print(f"The valueof pi is: {num:{1}.{5}}")
```

Output : The valueof pi is: 3.1416

4. Python String Template Class

1.Formatting String Python Using Template Class

This code imports the Template class from the string module. The Template class allows us to create a template string with placeholders that can be substituted with actual values. Here we are substituting the values n1 and n2 in place of n3 and n4 in the string n.

EXAMPLE:

```
from string import Template
n1 = 'Hello'
n2 = 'GeeksforGeeks'
n = Template('$n3 ! This is $n4.')
# and pass the parameters into the
# template string.
print(n.substitute(n3=n1, n4=n2))
```

OUTPUT:

```
Hello ! This is GeeksforGeeks.
```

5. How to Format String using center() Method

The center() method is a built-in method in Python’s str class that returns a new string that is centered within a string of a specified width.

EXAMPLE:

```
string = “GOOD MORNING!"
width = 30

centered_string = string.center(width)

print(centered_string)
```

OUTPUT:

```
GOOD MORNING!
```

COMMAND LINE PARAMETERS AND FLOW CONTROLS

• The arguments that are given after the name of the program in the command line shell of the operating system are known as **Command Line Arguments**. Python provides various ways of dealing with these types of arguments. The three most common are:

Using sys.argv

Using getopt module

Using argparse module

1.Using sys.argv

The sys module provides functions and variables used to manipulate different parts of the Python runtime environment.

It’s main purpose are:

•It is a list of command line arguments.

•len(sys.argv) provides the number of command line arguments.

•sys.argv[0] is the name of the current Python script.

2.Using getopt module

•Python **getopt module** is similar to the getopt() function of C.

• Unlike sys module getopt module extends the separation of the input string by parameter validation. It allows both short, and long options including a value assignment.

•**Syntax:**

getopt.getopt(args, options, [long_options])

Parameters:

args: List of arguments to be passed.

options: String of option letters that the script want to recognize. Options that require an argument should be followed by a colon (:).

long_options: List of string with the name of long options. Options that require arguments should be followed by an equal sign (=).

Return Type: Returns value consisting of two elements: the first is a list of (option, value) pairs. The second is the list of program arguments left after the option list was stripped.

3.Using argparse module

THANKYOU