Date	07 November 2022
Team ID	PNT2022TMID471454
Project Name	Project – Web Phishing Detection

Definition of sprint in Python

- The sprintf is a function to display the output of the given input using a python programming language.

- The sprintf is a print function to shows the output of the format strings in the python language.

- It is a coding element to assigns the "f get" method and displays string format output.

- The sprint is a print element to contain string – buffer data of the application and display in the string format using python technology.

- The sprint is a function similar to print, v print for display buffer output hustle-free.

- The python programming language is using the sprint function to declare argument and constructor elements.

- It is an output function to displays all data types like string and array elements.

Syntax:

The sprint python works with different data types, lengths of the data, and width.

It has used the percentage sign (%) before the type of the data.

The basic syntax of a sprint python shows below.

```
% [FLAG WIDTH. (DOT) PRECISION] TYPE
```

• The sprint python is using the "print" keyword to display output.

```
print("% [flag width . (dot) precision] type" % (value or
object))
```

The sprint python is used precision and type depends on the data type of the variable.

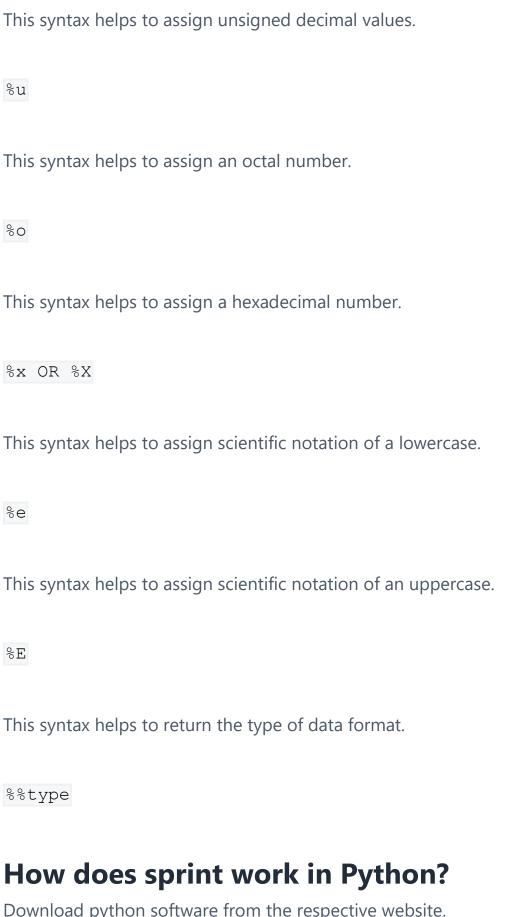This syntax helps to assign a signed decimal number. The length value of a decimal is 2.

```
%2d
```

This syntax helps to assign a binary number. The length value of binary is 4.

```
%4b
```

This syntax helps to assign a floating number. The length value of a decimal is 2.1.

```
%2.1f or %2.1F
```

This syntax helps to assign ASCII values.

```
%c
```

This syntax helps to assign unsigned decimal values.

`%u`

This syntax helps to assign an octal number.

`%o`

This syntax helps to assign a hexadecimal number.

`%x OR %X`

This syntax helps to assign scientific notation of a lowercase.

`%e`

This syntax helps to assign scientific notation of an uppercase.

`%E`

This syntax helps to return the type of data format.

`%%type`

How does sprint work in Python?

Download python software from the respective website.

Create a page with the dot (.) py extension.

The file name is the "function.py" to write a python program.

Create a variable with initializing the required data type value.

```
Varble_name = 34
```

Use the print keyword for the string format of the sprint python.

```
print (write sprint format)
```

Use percentage sign to return value.

```
print (" text : %d ")
```

Display length of the variable or value before data type.

```
print (" text : %2d ")
```

Display the type of the variable value in the format helping of the sprint.

Use parameter of the percentage sign to end of the sprintf function end.

Add variable to interconnect return function and application data.

```
print (" text : %2d " % (Varble_name))
```

Use direct value in sprint function to display decimal value.

```
print ("decimal number : %2d " % (7))
```

Use float type with sprintf formatted value in the return function.

```
print("Float number : %5.2f" % ( 23.11))
```

Combine the working procedure of the sprintf python to better understanding.

```
x = 34

print ("decimal number : %2d " % (x))

print ("decimal number : %2d " % (7))

print ("Float number : %5.2f" % ( 23.11))
```

Examples

Let us discuss examples of sprintf Python.

Example #1: The basic sprintf python example and the output show below

```
e_var = 34

print ("decimal number: %2d " % (e_var))

print ("decimal number: %2d " % (7))

print ("Float number: %5.2f" % ( 23.11))

print ("Float number: %5.4f" % (e_var))

print ("Octal number: %5o" % (e_var))

print ("Octal number: %3o" % (42))
```

Output:

```
decimal number: 34
decimal number:   7
Float number:   23.11
Float number:   34.0000
Octal number:       42
Octal number:    52
```

Example #2: The basic sprintf python with different types example and the output shows below

```python
e_var = 341234673

print ("decimal number: %d " % (e_var))

print ("Float number: %f" % (e_var))

print ("Float number: %F" % (e_var))

print ("unsigned decimal number: %u" % (e_var))

print ("Octal number: %o" % (e_var))

print ("first string value: %s" % (e_var))

print ("second string value: %s" % ("string data"))

print ("first hexadecimal value: %x" % (e_var))

print ("second hexadecimal value: %X" % (e_var))

print ("ASCII value: %c" % ("A"))

print ("lowercase scientific notation: %e" % (e_var))

print ("uppercase scientific notation: %E" % (e_var))

print ("first value: %g" % (e_var))

print ("second value: %G" % (e_var))
```

Output:

```
decimal number: 341234673
Float number:   341234673.000000
Float number:   341234673.000000
unsigned decimal number: 341234673
Octal number:   2425551761
first string value:   341234673
second string value:   string data
first hexadecimal value:   1456d3f1
second hexadecimal value:   1456D3F1
ASCII value:   A
lowercase scientific notation:   3.412347e+08
uppercase scientific notation:   3.412347E+08
first value:   3.41235e+08
second value:   3.41235E+08
```

Example #3: The basic sprintf python with positive and negative value example and output shows below

```python
e_var = 341234673

f_var = -341234673

print ("decimal number: %d " % (e_var))

print ("decimal number: %d \n " % (f_var))

print ("unsigned decimal number: %u" % (e_var))

print ("unsigned decimal number: %u \n" % (f_var))

print ("Octal number: %o" % (e_var))

print ("Octal number: %o \n" % (f_var))

print ("first hexadecimal value: %x " % (e_var))

print ("first hexadecimal value: %x \n" % (f_var))

print ("second hexadecimal value: %X " % (e_var))

print ("second hexadecimal value: %X \n" % (f_var))

print ("lowercase scientific notation: %e" % (e_var))

print ("lowercase scientific notation: %e \n" % (f_var))
```

```python
print ("uppercase scientific notation: %E" % (e_var))

print ("uppercase scientific notation: %E \n" % (f_var))

print ("first value: %g" % (e_var))
print ("second value: %G" % (f_var))
```

Output:

```
decimal number: 341234673
decimal number: -341234673

unsigned decimal number: 341234673
unsigned decimal number: -341234673

Octal number:  2425551761
Octal number:  -2425551761

first hexadecimal value:  1456d3f1
first hexadecimal value:  -1456d3f1

second hexadecimal value:  1456D3F1
second hexadecimal value:  -1456D3F1

lowercase scientific notation:  3.412347e+08
lowercase scientific notation:  -3.412347e+08

uppercase scientific notation:  3.412347E+08
uppercase scientific notation:  -3.412347E+08

first value:  3.41235e+08
second value:  -3.41235E+08
```

Example #4: The sprintf python with different length example and the output shows below

```python
e_var = 341234673

f_var = -341234673

print ("decimal number: %2d " % (e_var))

print ("unsigned decimal number: %1u \n" % (f_var))

print ("Octal number: %2o \n" % (e_var))

print ("first hexadecimal value: %1x " % (e_var))

print ("second hexadecimal value: %5X \n" % (e_var))

print ("lowercase scientific notation: %2e" % (e_var))

print ("uppercase scientific notation: %1E \n" % (f_var))
```

```
print ("Float value: %2.1f" % (e_var))

print ("Float value: %1.2f \n" % (f_var))

print ("Octal value: %2o" % (e_var))
```

Output:

```
decimal number: 341234673
unsigned decimal number: -341234673

Octal number:  2425551761

first hexadecimal value:  1456d3f1
second hexadecimal value:  1456D3F1

lowercase scientific notation:  3.412347e+08
uppercase scientific notation:  -3.412347E+08

Float value:  341234673.0
Float value:  -341234673.00

Octal value:  2425551761
```

Conclusion

- It is easy to return data in any format per application requirement.

- It helps to create web applications attractive, understandable, and user-friendly.