PYTHON STRING FORMAT

String format() : The string format() method formats the given string into a output in Python.

Syntax: '{} {}'.format('one', 'two')

String format() Parameters : format() method takes any number of parameters. But, is divided into two types of parameters:

Positional parameters - list of parameters that can be accessed with index of parameter inside curly braces {index}

Syntax: “sometext {0},sometext{1}”.format(value1,value2)

E.g: “Hello {0}, your employee id is{1}”.format(“sai”,123456)

Keyword parameters - list of parameters of type key=value, that can be accessed with key of parameter inside curly braces {key}

Syntax: “sometext {key},sometext{key}”.format(value1,value2)

E.g: “Hello {name}, your employee id is{idno}”.format(name=“sai”,idno=123456)

Default parameters: List of parameters that can be accessed without any index or key value, but left empty in curly braces are considered as default parameters

Syntax: “sometext {},sometext{}”.format(value1,value2)

E.g: “Hello { }, your employee id is{ }”.format(“sai”, 123456)

Mixed parameters: List of parameters that can be accessed with both index and key value, Inside curly braces are considered as mixed parameters

Syntax: “sometext {index},sometext{key}”.format(value1,value2)

E.g: “Hello {0}, your employee id is{idno}”.format(“sai”,idno=123456)

Numbers formatting with format()

You can format numbers using the format specifier:

d: integer

c: Corresponding Unicode character

b: Binary format

o: Octal format

x:Hexadecimal format (lower case)

X: Hexadecimal format (upper case)

n:Same as 'd'. Except it uses current locale setting for number separator

e: Exponential notation. (lowercase e)

E:Exponential notation (uppercase E)

f:Displays fixed point number (Default: 6)

F:Same as 'f'. Except displays 'inf' as 'INF' and 'nan' as 'NAN'

g:General format. Rounds number to p significant digits. (Default precision: 6)

G:Same as 'g'. Except switches to 'E' if the number is large.

%:Percentage. Multiples by 100 and puts % at the end.

E.g:

```
# integer arguments
print("The number is:{:d}".format(123))

# float arguments
print("The float number is:{:f}".format(123.4567898))

# octal, binary and hexadecimal format
print("bin: {0:b}, oct: {0:o}, hex: {0:x}".format(12))
```

Output:the number is 123
the float number is 123.4567898
bin:1100, oct:14,hex: c

Number formatting with padding for int and floats

E.g:

```
 # integer numbers with minimum width
print("{:5d}".format(12))
```

Output: ---12

E.g:

```
# width doesn't work for numbers longer than padding
print("{:2d}".format(1234))
```

Output: 1234

E.g:

```
# padding for float numbers
print("{:8.3f}".format(12.2346))
```

Output: --12.235

E.g:

```
# integer numbers with minimum width filled with zeros
print("{:05d}".format(12))
```

Output: 00012

E.g:

```
# padding for float numbers filled with zeros
print("{:08.3f}".format(12.2346))
```

output: 0012.235

Truncating long strings:

E.g:

```
# truncating strings to 3 letters
print("{:.3}".format("caterpillar"))
```

Output: cat

E.g:

```
# truncating strings to 3 letters and padding

print("{:5.3}".format("caterpillar"))
```

Output: cat--

E.g:

```
 # truncating strings to 3 letters,  padding and center alignment

print("{:^5.3}".format("caterpillar"))
```

Output: _ cat_

Formatting class and dictionary members using format()

Python internally uses getattr() for class members in the form ".age". And, it uses __getitem__()lookup for dictionary members in the form "[index]".

Formatting class members using format()

E.g:

```
# define Person class
class Person:
   age = 23
   name = "Adam"

# format age
print("{p.name}'s age is: {p.age}".format(p=Person()))
```

Output:

Adam's age is 23

Formatting dictionary members using format()

E.g:

```
# define Person dictionary
person = {'age': 23, 'name': 'Adam'}

# format age
print("{p[name]}'s age is: {p[age]}".format(p=person))
```

Output:

Adam’s age is 23