Project Development Phase
Project Development-Delivery of Sprint-4

Date	**15** November 2022
Team ID	PNT2022TMID483156
Project Name	Project – Crude Oil Prediction

sprintf like functionality in Python

I would like to create a string buffer to do lots of processing, format and finally write the buffer in a text file using a C-style sprintf functionality in Python. Because of conditional statements, I can't write them directly to the file.

e.g pseudo code:

```
sprintf(buf,"A = %d\n , B= %s\n",A,B)
/* some processing */
sprint(buf,"C=%d\n",c)
....
...
fprintf(file,buf)
```

So in the output file we have this kind of o/p:

```
A= foo B= bar
C= ded
etc...
```

Edit, to clarify my question:

buf is a big buffer contains all these strings which have formatted using sprintf. Going by your examples, buf will only contain current values, not older ones. e.g first in buf I wrote A= something ,B= something later C= something was

appended in the same buf, but in your Python answers buf contains only last value, which is not I want - I want buf to have all the printfs I have done since the beginning, like in C.

Python has a % operator for this.

```
>>> a = 5
>>> b = "hello"
>>> buf = "A = %d\n , B = %s\n" % (a, b)
>>> print buf
A = 5
 , B = hello


>>> c = 10
>>> buf = "C = %d\n" % c
>>> print buf
C = 10
```

See this reference for all supported format specifiers.

You could as well use format:

```
>>> print "This is the {}th tome of {}".format(5, "knowledge")
This is the 5th tome of knowledge
```

44

If I understand your question correctly, format() is what you are looking for, along with its mini-language.

Silly example for python 2.7 and up:

```
>>> print "{} ...\r\n {}!".format("Hello", "world")
Hello ...
 world!
```

For earlier python versions: (tested with 2.6.2)

```
>>> print "{0} ...\r\n {1}!".format("Hello", "world")
Hello ...
 world!
```

I'm not completely certain that I understand your goal, but you can use a `StringIO` instance as a buffer:

```
>>> import StringIO
>>> buf = StringIO.StringIO()
>>> buf.write("A = %d, B = %s\n" % (3, "bar"))
>>> buf.write("C=%d\n" % 5)
>>> print(buf.getvalue())
A = 3, B = bar
C=5
```

Unlike `sprintf`, you just pass a string to `buf.write`, formatting it with the `%` operator or the `format` method of strings.

You could of course define a function to get the `sprintf` interface you're hoping for:

```
def sprintf(buf, fmt, *args):
    buf.write(fmt % args)
```

which would be used like this:

```
>>> buf = StringIO.StringIO()
>>> sprintf(buf, "A = %d, B = %s\n", 3, "foo")
>>> sprintf(buf, "C = %d\n", 5)
>>> print(buf.getvalue())
A = 3, B = foo
C = 5
```

You can use string formatting:

```
>>> a=42
>>> b="bar"
>>> "The number is %d and the word is %s" % (a,b)
'The number is 42 and the word is bar'
```

But this is removed in Python 3, you should use "str.format()":

```
>>> a=42
>>> b="bar"
>>> "The number is {0} and the word is {1}".format(a,b)
'The number is 42 and the word is bar'
```

If you want something like the python3 print function but to a string:

```
def sprint(*args, **kwargs):
    sio = io.StringIO()
    print(*args, **kwargs, file=sio)
    return sio.getvalue()
```

```
>>> x = sprint('abc', 10, ['one', 'two'], {'a': 1, 'b': 2}, {1, 2, 3})
>>> x
"abc 10 ['one', 'two'] {'a': 1, 'b': 2} {1, 2, 3}\n"
```

or without the '\n' at the end:

```
def sprint(*args, end='', **kwargs):
    sio = io.StringIO()
    print(*args, **kwargs, end=end, file=sio)
    return sio.getvalue()
```

```
>>> x = sprint('abc', 10, ['one', 'two'], {'a': 1, 'b': 2}, {1, 2, 3})
>>> x
"abc 10 ['one', 'two'] {'a': 1, 'b': 2} {1, 2, 3}"
```