Python Fundamentals day 20

Today's Agenda

- String formatting
- Format specification types
- Programs

String formatting

In python we have two ways to achieve string formatting

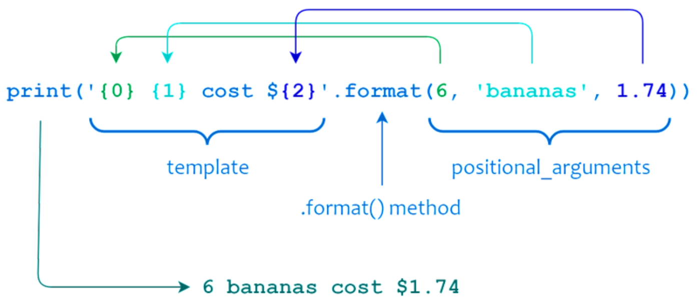
- ❖ format() method
- ❖ f string literal

format() method

The format() method formats the specified value(s) and insert them inside the string's placeholder. The placeholder is defined using curly brackets { }.

Syntax:

string.format(*args)



```
print('{0} {1} cost ${2}'.format(6, 'bananas', 1.74))
```

```
6 bananas cost $1.74
```

```
name=input("Enter your name\n")
place=input("Enter your place\n")

s="Hello {}, How is the weather in {}?".format(name,place)
print(s)
```

Output:

```
In [1]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')

Enter your name
rohit

Enter your place
blore
Hello rohit, How is the weather in blore?
```

Let us see another example

```
s="{} {} {}".format(10,20,30)
print(s)
```

Output:

```
In [2]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
10 20 30
```

Note: If nothing is specified in place holders, the default order is considered.

Now let us see an example where order is specified explicitly

```
s="{2} {0} {1}".format(10,20,30)
print(s)
```

Output:

```
In [3]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
30 10 20
```

Within the place holder { } not only can we pass positional values, but also additional values which is going to change the way of its representation in different ways.

string →"{position:format_specification}"

format specification types:

- Alignment
- Presentation
- Conversion

Format specification

Alignment:

- >right alignment

```
s="{0:*>10}".format(999) #right alignment with * as the fill
print(s)
```

Output:

```
In [5]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
*******999
```

- <left alignment

```
s="{0:*<10}".format(999) #left alignment with * as the fill
print(s)
```

Output:

```
In [4]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
999*******
```

- ^ center alignment

```
s="{0:*^11}".format(999) #center alignment with * as the fill
print(s)
```

Output

```
In [6]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
****999****
```

Presentation:

- f-fixed point notation

```
import math
s="{0:.4f}".format(math.pi) #print pi value with 4 decimal points
print(s)
```

Output:

```
In [7]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
3.1416
```

We can also have pi value beginning with 0's and mentioning number of digits as follows

```
import math
s="{0:010.4f}".format(math.pi)
print(s)
```

Output:

```
In [8]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
00003.1416
```

- e-exponent notation

```
ew=5976000000000000000000000 #earth weight
s="{0:e}".format(ew)
print(s)
```

Output:

```
In [9]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
5.976000e+23
```

As we know we can adjust the decimal numbers let us see how we can do it here

```
ew=5976000000000000000000000 #earth weight
s="{0:.3e}".format(ew)
print(s)
```

Output:

```
In [10]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
5.976e+23
```

Let us now try to pass different arguments inside place holders.

```
s="{} {} {}".format([10,20,30]) #passing list as input
print(s)
```

Output:

```
  File "C:\Users\rooman\Anaconda3\lib\site-
packages\spyder_kernels\customize
\spydercustomize.py", line 110, in execfile
    exec(compile(f.read(), filename, 'exec'),
namespace)

  File "C:/Users/rooman/OneDrive/Desktop/
python/test.py", line 1, in <module>
    s="{} {} {}".format([10,20,30]) #passing
list as input

IndexError: tuple index out of range
```

The above input will give an error because we have three place holders but we are trying to pass three inputs in a single list. Is there a way to tackle this? Definitely, let us see how

```
s="{0[0]} {0[1]} {0[2]}".format([10,20,30]) #passing list as input
print(s)
```

Output:

```
In [12]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
10 20 30
```

However there's also an easy way to do this called as **unpacking** as shown below

```
s="{0} {1} {2}".format(*[10,20,30]) #unpack by using *
print(s)
```

Output:

```
 In [13]: runfile('C:/Users/rooman/OneDrive/
 Desktop/python/test.py', wdir='C:/Users/rooman/
 OneDrive/Desktop/python')
 10 20 30
```

Programs

1. Let us take numbers as input from user and try to find the average of numbers with only 4 decimal points.

```
nums=input("Enter the number\n").split() #splits the number present in string
print(nums)
print(type(nums))
```

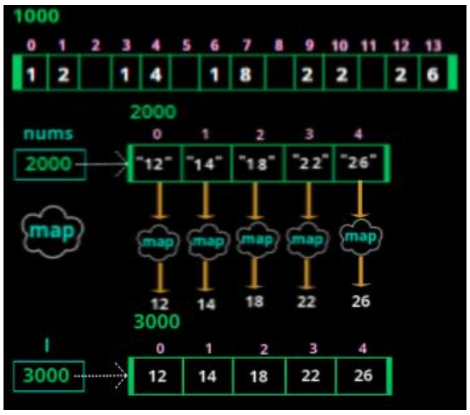
Output:

```
In [14]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')

Enter the number
12 14 18 22 26
['12', '14', '18', '22', '26']
<class 'list'>
```

We now have the list of numbers, but we cannot perform any math operations on list of strings. So now we have to map strings to integers as shown below

```
nums=input("Enter the number\n").split() #splits the number present in string
print(nums)
l=list(map(int,nums))
print(l)
```

Logic:



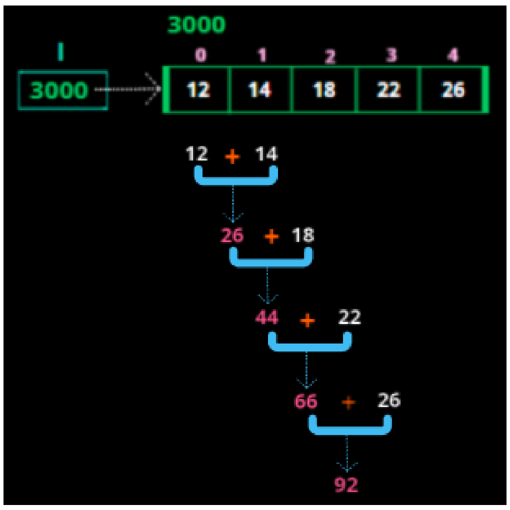
Output:

```
In [16]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')

Enter the number
12 14 18 22 26
['12', '14', '18', '22', '26']
[12, 14, 18, 22, 26]
```

Now we have list of integers, next we should reduce the list to the sum of all the integers by using reduce() function

```
from functools import reduce

nums=input("Enter the number\n").split() #splits the number present in string
l=list(map(int,nums))
res=reduce(lambda x,y:x+y,l)
print(res)
```

Logic:



Output:

```
In [17]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')

Enter the number
12 14 18 22 26
92
```

As expected we got the sum of all the integers, now we should take the average by dividing it with number of integers present as shown below

```
from functools import reduce

nums=input("Enter the number\n").split() #splits the number present in string
l=list(map(int,nums))
res=reduce(lambda x,y:x+y,l)
avg=res/len(l)
print("{0:.4f}".format(avg))
print("{0:^14.4f}".format(avg)) #center align to 14
```

Output:

```
In [18]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')

Enter the number
12 14 18 22 26
18.4000
   18.4000
```

Try to solve such programs which involve application of the functions you have studied so far and join the dots.