For the following format specifiers, PLEASE FILL OUT COMPLETE INFORMATION. It will not be sufficient to copy and paste information you find on the internet. For your own success in this lesson you should articulate your understanding of how each of these specifiers work.

|  |  |
| --- | --- |
| **{:<10}** | When do you use it?  When I want texts with a maximum of 10 spaces to be aligned to the left.  How does it work?  It aligns the texts to the left of the extra spaces. If I want the extra spaces to be filled by a specific character, I will put the character between “:” and “<”.  Provide examples:  print(“{:\*<10}”.format(“place”)) will print as place\*\*\*\*\* |
| **{:>10}** | When do you use it?  When I want texts with a maximum of 10 spaces to be aligned to the right.  How does it work?  It aligns the texts to the right of the filling characters.  Provide examples:  print(“{:&>10}”.format(“place”)) will print as &&&&&place |
| **{:^10}** | When do you use it?  When I want texts to be aligned in the center.  How does it work?  The the text will be printed in the center and the other character will be printed in the extra space on both sides.  Provide examples:  print(“{:\*^10}”.format(“text”)) will print as \*\*\*text\*\*\* |
| **{:00.0f}** | When do you use it?  When I want to align numerical data with decimal places.  How does it work?  The number to the left of “.” represents the minimum spaces I want when it prints. The number to the right of “.” represents decimal places.  Provide examples:  print(“{:6.2f}”.format(23.568)) will print as 23.57. |

Extra Notes

* Under the defining statement, the print format statement should be tabbed.
* The variable should be initialized below the print formatting statement.
* Function parameters allow the same function to be performed multiple times with different input data.
* def functionName(<parameters>):

#statement 1

#statement 2

#statement 3, etc...