

# FancyPrint Module

## Screen Functions

<code>clean()</code>	It cleans the terminal and returns the cursor to home.
<code>clear()</code>	It clears the terminal and returns the cursor to home.
<code>erase()</code>	It erases the terminal and leaves the cursor in the current position.
<code>dimensions()</code>	It returns the dimensions of the terminal, cols and rows.
<code>resize(rows=25, cols=80)</code>	It resizes the terminal size.

**Example:**

```
import fancyprint as fp
fp.clean()
r, c = fp.dimensions()
print(f"rows: {r}, Cols: {c}")
fp.resize(25, 120)
```

## Internal Functions

`bg_ansi_colors(bold=False, fg=-1, n_line=0)`

This function displays all background colors available with ansi code. The following options are for a better visualization.

- 1.- The bold option for the font (True / False)
- 2.- The fg option to visualize the background colors with a specific foreground color.
- 3.- The n\_line option to insert lines between the colors.

`fg_ansi_colors(bold=False, bg=-1, n_line=0)`

This function displays all foreground colors available with ansi code. The following options are for a better visualization.

- 1.- The bold option for the font (True / False)
- 2.- The bg option to visualize the background colors with a specific foreground color.
- 3.- The n\_line option to insert lines between the colors.

`ins_chr(n=1, unicode=" ")` → This function inserts n times the unicode provided, by default it is set to space.

`ins_newline(n=1)` → This function inserts n new lines.

`terminal_bell()` → This function makes the sound of the terminal bell.

`reset_font()` → This function resets the font attributes when we use the `set_font()` function.

`set_font(bold=False, bg=-1, fg=-1, italic=False, underline=False, strike=False, blinking=False, dim=False, hidden=False, inverse=False)` →

This function passes many attributes for the font. If passing all these arguments is a little annoying to you, you can use the Font Style Class for simplicity. The best way to use this function is to pass only the first 3 parameters like the example below.

Colors range goes from -1 to 256. To set the default color from the system use -1 or 256.

**Example:**

```
import fancyprint as fp
print(fp.set_font(1,11,21) + " Python is " + fp.set_font(0,1) + " Wonderful."+fp.reset_font())
print(f"{fp.set_font(bold=0, bg=22, fg=0)} Python {fp.set_font(1,90,7)} Language.{fp.reset_font()}")
```

**Note:** These functions are being used by the FancyFormat Class. Feel free to ignore them if not useful to you.

## Help Classes

**Move** → This class is used with the Cursor class and it contains 4 options.

Move.RIGHT      Move.LEFT      Move.UP      Move.DOWN

**Note:** These options can be replaced for the original values as displays below:

Move.RIGHT = "right" = "r"      Move.LEFT = "left" = "l"  
Move.UP = "up" = "u"      Move.DOWN = "down" = "d"

**Align** → This class is used with the FancyFormat class and FancyMessage class. It contains 4 options.

Align.RIGHT      Align.LEFT      Align.CENTER      Align.JUSTIFY

**Note:** These options can be replaced for the original values as displays below:

Align.RIGHT = "right" = "r"      Align.LEFT = "left" = "l"  
Align.CENTER = "center" = "c"      Align.JUSTIFY = "justify" = "j"

**Layout** → This class is used with FancyFormat class and Pen class. It contains 2 options.

Layout.HORIZONTAL = "horizontal"      Layout.VERTICAL = "vertical"

**Length\_bg** → This class is used with FancyMessage class and contains 2 options.

ALL\_ROW      ONLY\_WORD

**Unicode** → This class is to insert some unicode characters.

#-----		
# Lines	Triangle	-
#-----		
BOX_DRAWINGS_LIGHT_HORIZONTAL	BLACK_UP_POINTING_TRIANGLE	
BOX_DRAWINGS_LIGHT_VERTICAL_AND_RIGHT	WHITE_UP_POINTING_TRIANGLE	
BOX_DRAWINGS_LIGHT_VERTICAL_AND_LEFT	BLACK_RIGHT_POINT_TRIANGLE	
BOX_DRAWINGS_LIGHT_VERTICAL	WHITE_RIGHT_POINT_TRIANGLE	
BOX_DRAWINGS_LIGHT_DOWN_AND_HORIZONTAL	BLACK_DOWN_POINTING_TRIANGLE	
BOX_DRAWINGS_LIGHT_UP_AND_HORIZONTAL	WHITE_DOWN_POINTING_TRIANGLE	
BOX_DRAWINGS_LIGHT_VERTICAL_AND_HORIZONTAL	BLACK_LEFT_POINTING_TRIANGLE	
EM_DASH	WHITE_LEFT_POINTING_TRIANGLE	
#-----		
# Miscellaneous		-
#-----		
BLACK_DIAMOND	WHITE_DIAMOND	FACE
BLACK_CIRCLE	WHITE_CIRCLE	<b>Reference</b> → <a href="https://www.unicode.org/charts/nameslist/">https://www.unicode.org/charts/nameslist/</a>

**Line\_Style** → This class is used with FancyFormat class and Pen class. There are some options available.

CUSTOMIZED      SINGLE      DOUBLE\_SPACE\_COL\_COLOR  
SINGLE\_BOLD      DASH      NO\_SPACE\_COL\_COLOR  
SINGLE\_HEAVY      DOUBLE  
SQR\_BRACKETS      NONE

**Note:** DOUBLE\_SPACE\_COL\_COLOR and NO\_SPACE\_COL\_COLOR are not included in Pen class.

**Note:** These options can be replaced for the original value as displays below:

CUSTOMIZED	→ "customized"	SINGLE	→ "single"	NO_SPACE_COL_COLOR	→ "no_space_col_color"
SINGLE_BOLD	→ "single_bold"	DASH	→ "dash"	DOUBLE_SPACE_COL_COLOR	→ "double_space_col_color"
SINGLE_HEAVY	→ "single_heavy"	DOUBLE	→ "double"		
SQ_BRACKETS	→ "sq_brackets"	NONE	→ "none"		

Variables to visualize the effect on options `DOUBLE_SPACE_COL_COLOR` and `NO_SPACE_COL_COLOR` with `FancyFormat`.

<code>bg_horizontal_line = 21</code>	<code>bg_header = 90</code>	<code>bg_data = 231</code>
<code>bg_vertical_line = 21</code>	<code>fg_header = 231</code>	<code>fg_data = 0</code>
<code>bg_corner_chr = 21</code>	<code>bold_header = True</code>	<code>bold_data = True</code>
<code>bg_inner_corner_chr = 21</code>	<code>bg_corner_under_line_header = 21</code>	<code>middle_horizontal_line_on = True</code>
<code>bg_under_line_header = 21</code>	<code>bg_vertical_header_line_chr = 21</code>	<code>horizontal_line_under_header_on = True</code>

Example:

```
import fancyprint as fp
tbl1 = fp.FancyFormat()
tbl1.print_fancy_format(data=lst2, style=fp.Line_Style.DOUBLE_SPACE_COL_COLOR)
tbl1.print_fancy_format(data=lst3, style=fp.Line_Style.NO_SPACE_COL_COLOR)
```

## Cursor Class

This class contains 4 methods. The difference between `jump` and `move` is that `jump` executes the code while `move` returns the code.

<code>jumpTo(qty=0, direction=fp.Move.DOWN)</code>	→ This method jumps rows or columns for the cursor in the terminal.
<code>jumpxy(x=0, y=0)</code>	→ This method jumps the cursor to specific coordinates in the terminal.
<code>moveTo(qty=0, direction=fp.Move.DOWN)</code>	→ This method moves rows or columns for the cursor in the terminal.
<code>movexy(x=0, y=0)</code>	→ This method moves the cursor to specific coordinates in the terminal.

Example:

```
import fancyprint as fp
crs = fp.Cursor()
crs.jumpTo(4, "D")

crs.jumpTo(qty=20, direction=fp.Move.RIGHT) ← . → crs.jumpTo(qty=20, direction="right")
print("Hello There...!")

print(f'{crs.moveTo(qty=20, direction=fp.Move.RIGHT)}Hello There...!')
print(f'{crs.movexy(0,10)}Col 10, row 1')
```

## FontStyle Class

This class contains 4 methods and the attributes and their default values are displays below.

<code>bold</code>	= False	<code>bg</code>	= -1	<code>fg</code>	= -1	<code>italic</code>	= False
<code>dim</code>	= False	<code>underline</code>	= False	<code>blinking</code>	= False	<code>inverse</code>	= False
<code>hidden</code>	= False	<code>strike</code>	= False	<code>indent</code>	= False	<code>next_line</code>	= True

<code>indent</code>	→ this defines how far we want to start to print the message from the left.
<code>next_line</code>	→ this defines where we want to jump the line or not when printing the message.

`print_style(msg)` → This method will print the style with the defined attributes.

**Example:**

```
import fancyprint as fp
fs = fp.FontStyle()
fs.bg = 21
fs.fg = 231
fs.print_style(msg = " My Font Style ")

# reset_style() → This method will reset the style to the default values.
fs.reset_style()
fs.print_style(" My Font Style ")
```

`start_style()` and `stop_style()` → These methods are used if we will be continuing using the style in many rows.

**Example:**

```
import fancyprint as fp
fs = fp.FontStyle()
fs.bg = 21
fs.fg = 231
print(f"{fs.start_style()} Font Style Line 1 ")
print(f" Font Style Line 2 ")
print(f" Font Style Line 3 {fs.stop_style()}")
fs.reset_style()
print(f"{fs.start_style()} Default Style {fs.stop_style()}")
```

## FancyMessage Class

```
#-----
# Body Default Values
#-----
bg_body    = 4           strike_body    = False      msg_body    = "Body Msg"
fg_body    = 231         hidden_body   = False      right_indent = 2
bold_body  = False       inverse_body  = False      left_indent  = 2
dim_body   = False       blinking_body = False      top_lines   = 1
italic_body= False       underline_body= False      bottom_lines = 1

help_lines = False       length = Length_bg.ALL_ROW

# These two options work when length is Length_bg.ONLY_WORD. They don't do anything when length is Length_bg.All_ROW.
adj_bg_lines_to_right_indent = False
adj_bg_msg_to_space_available = False
```

**Note:** All the above variables are being used by both methods, `print_fancy_message` and `print_fancy_note`.

```
#-----
# Note Attributes
#-----
msg_note = " Note: "    align_note   = Align.JUSTIFY  blinking_note = False
bg_note  = 231          strike_note  = False         underline_note = False
fg_note  = 0            italic_note   = False         position_note  = 1
bold_note= False        inverse_note  = False         right_space_note = 2
dim_note = False        hidden_note   = False         left_space_note = 2
```

```
#-----
# Title Attributes -
#-----
msg_title = ""          align_title = Align.LEFT      blinking_title = False
bg_title   = 4          strike_title = False          underline_title = False
fg_title   = 231        italic_title  = False          title_indent    = 2
bold_title = False      inverse_title = False          lines_title_body = 1
dim_title  = False      hidden_title  = False

#-----
# Footnote Attributes -
#-----
msg_footnote = ""       align_footnote = Align.RIGHT   blinking_footnote = False
bg_footnote  = 4        strike_footnote = False        underline_footnote = False
fg_footnote  = 231      italic_footnote = False        footnote_indent    = 2
bold_footnote = False   inverse_footnote = False        lines_body_footnote = 1
dim_footnote = False    hidden_footnote = False
```

### Example:

```
import fancyprint as fp

msg = fp.FancyMessage()

paragraph = """
    Guido van Rossum, a Dutch programmer, created Python in the late 1980s
    as a hobby project. He started working on it in December 1989 at Cent-
    rum Wiskunde & Informatica (CWI) in the Netherlands.

    Python was first released on February 20, 1991. Python was named after
    the 1970s BBC comedy sketch series Monty Python's Flying Circus.
    """

msg.msg_title = "TITLE"

msg.msg_footnote = "FOOTNOTE"

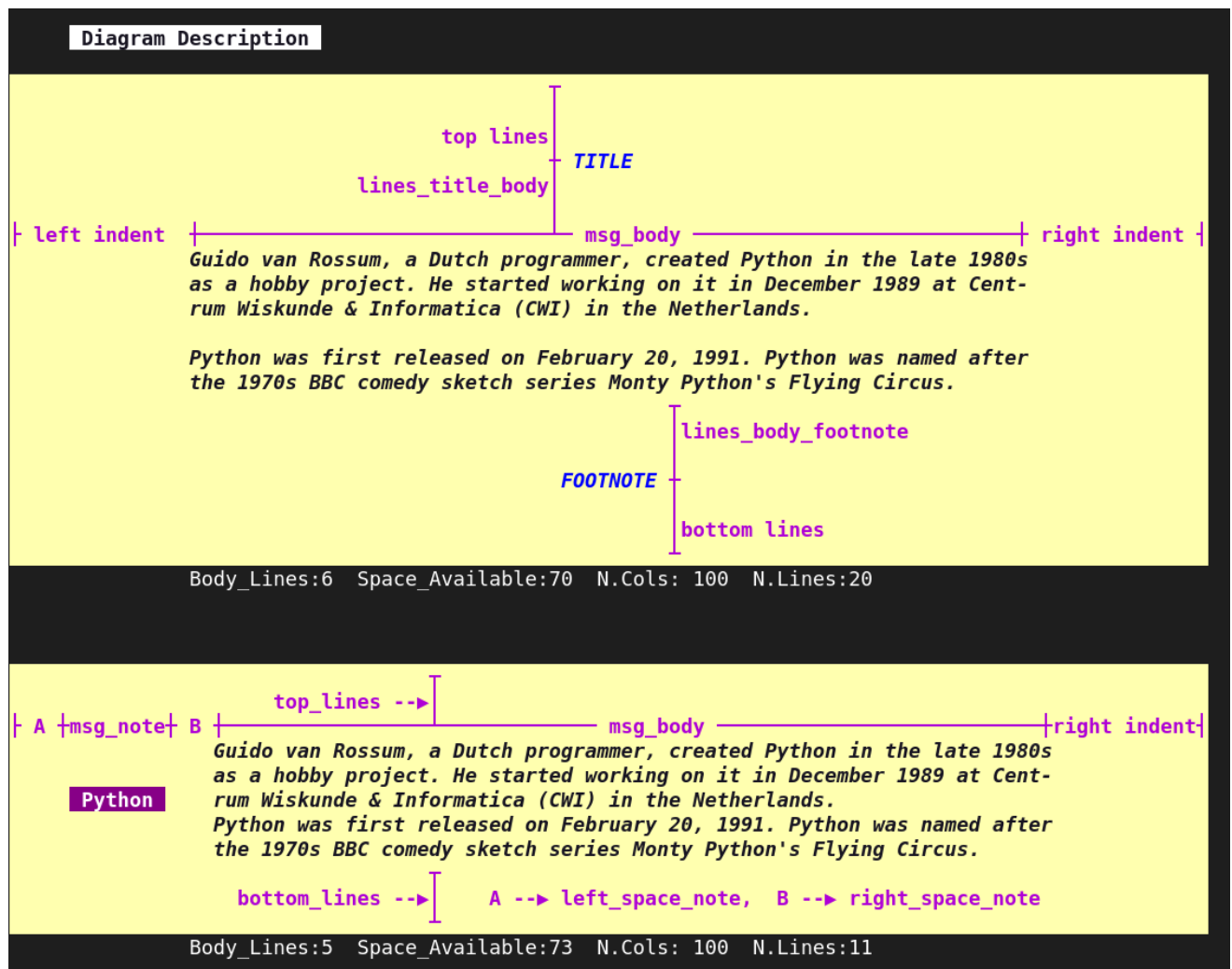
msg.print_fancy_message(paragraph)

fp.ins_newline(2)

msg.msg_note = "Python"

msg.position_note = 4

msg.print_fancy_note(paragraph)
```



## FancyFormat Class

This class contains two methods:

`print_fancy_format(data, style)` → Two arguments, the data to print and the line style.  
`reset_fancy_format()` → It resets all the attributes to their default values.

**Examples:** `import fancyprint as fp`  
`tbl = fp.FancyFormat()`

**Case 1:** Passing any type of variable.  
`tbl.print_fancy_format("Hello World...!")`

**Output:**

```
+-----+
| Hello World...! |
+-----+
```

**Case 2:** Passing an empty list.  
`tbl.print_fancy_format([])`

**Output:**

```
+-----+
| none |
+-----+
```

**Case 3:** Passing single item in a list.

```
my_list = ["Hello World...!"]
tbl.print_fancy_format(my_list)
```

**Output:**

Hello World...!

**Case 5:** Passing a list.

```
my_list = [1,2,3,4]
tbl.print_fancy_format(my_list)
```

**Output:**

1	2	3	4
---	---	---	---

**Case 6:** Passing a list in a single row.

```
my_list = [[1,2,3,4]]
tbl.print_fancy_format(my_list)
```

**Output:**

1	2	3	4
---	---	---	---

**Case 7:** Passing a list with a some combination of rows and cols.

```
my_list = [[5,"hello"],6,50,[45]]
tbl.print_fancy_format(my_list)
```

**Output:**

[5, 'hello']	6	50	[45]
--------------	---	----	------

**Case 8:** Passing a list with rows and one col.

```
my_list = [[10],[20],[30],[40]]
tbl.print_fancy_format(my_list)
```

**Output:**

R1C1	10
R1C2	20
R1C3	30
R1C4	40

**Case 9:**

Passing a list with a some combination of rows and cols.

```
my_list = [
    ["R1C1","R1C2","R1C3"],
    ["R2C1","R2C2","R2C3"],
    ["R3C1","R3C2","R3C3"]]
```

```
tbl.print_fancy_format(my_list1)
```

**Case 4:** Passing single item in a row to a list.

```
my_list = [["hello there!"]]
tbl.print_fancy_format(my_list)
```

**Output:**

hello there!
--------------

```
my_list = ["Terminology","hello there!", "I am Miguelito"]
tbl.print_fancy_format(my_list)
```

**Output:**

Terminology	hello there!	I am Migue
-------------	--------------	------------

```
my_list = [
    ["Terminology","hello there!", "I am Hello"]]
tbl.print_fancy_format(my_list)
```

**Output:**

Terminology	hello there!	I am Hello
-------------	--------------	------------

```
my_list1 = [10,[50],[250],["C"],["H"],10,20]
tbl.print_fancy_format(my_list)
```

**Output:**

10	[50]	[250]	['C']	['H']	10	20
----	------	-------	-------	-------	----	----

```
my_list = [
    ["R1C1","R1C2","R1C3"],
    ["R2C1","R2C2","R2C3"],
    ["R3C1","R3C2","R3C3"]]
```

```
tbl.horizontal_line_under_header_on= True
tbl.middle_horizontal_line_on      = True
tbl.print_fancy_format(my_list1)
```

```

Output: +-----+-----+-----+-----+
| R1C1 | R1C2 | R1C3 | R1C4 |
| R2C1 | R2C2 | R2C3 | R2C4 |
| R3C1 | R3C2 | R3C3 | R3C4 |
| R4C1 | R4C2 | R4C3 | R4C4 |
+-----+-----+-----+-----+

+-----+-----+-----+-----+
| R1C1 | R1C2 | R1C3 | R1C4 |
+-----+-----+-----+-----+
| R2C1 | R2C2 | R2C3 | R2C4 |
+-----+-----+-----+-----+
| R3C1 | R3C2 | R3C3 | R3C4 |
+-----+-----+-----+-----+
| R4C1 | R4C2 | R4C3 | R4C4 |
+-----+-----+-----+-----+

```

**Note:** Although the main idea is to use list type, `print_fancy_format(tbl)` accepts any type of variable. Refer to [Demo 3](#) figure.

### Attributes in FancyFormat Class:

```

#-----
# General Use Section
#-----
adj → adjust

```

```

adj_top_margin = 0      adj_bottom_margin = 0      adj_indent = 2      set_fill_chr = "----"      set_layout = Layout.HORIZONTAL
adj_top_space  = 0      adj_bottom_space  = 0      adj_space  = 2      update_list = False

```

`adj_top_margin` Lines to be added between the terminal (\$) and the title. It only accepts int values.

`adj_top_space` Lines to be added between title and top list. It only accepts int values.

`adj_bottom_margin` Lines to be added between the end of the list or footnote to the terminal (\$).

`adj_bottom_space` Lines to be added between the bottom of the list and the footnote. It only accepts int values.

`adj_indent` Space from the left terminal to the first character in the list to be printed. It only accepts int values.

`adj_space` Space from the left of the box to the first character in the list to be printed. It only accepts int values.

`set_fill_chr` When a list is not complete in the data, it will be filled out with some characters. `fill_chr` will be converted to string.

`update_list` Notice that every single element in the list being passed will be converted to string in a temporary internal list. If you want to save this conversion to your original list then set to True. It only works with the list type of variable.

`set_layout` This option only works with `set`, `frozenset` or `range` type of variables.

**Note:** `adj_top_space` won't work if the title is not set up. Also `adj_bottom_space` won't work if the footnote is not set up. Use `adj_top_margin` or `adj_bottom_margin` or `ins_newline(n)`, or `print("\n")` if you need more space.

```

#-----
# Title Section
#-----
msg_title = ""      align_title = "justify"      hidden_title = False
bold_title = False   italic_title = False      inverse_title = False
bg_title   = -1      strike_title = False      blinking_title = False
fg_title   = -1      dim_title   = False      underline_title = False

```

`msg_title` The title name for the list. It only accepts string values, by defaults is empty.

`bold_title` It only accepts two int values 0 and 1, by defaults is set to 0.

`bg_title` and `fg_title` accepts int values from -1 to 256. Default value from the system are -1 and 256.

`align_title` It accepts 4 values, left (l), justify (j), center (c), and right (r).



```
#-----
# Footnote Section -
#-----
msg_footnote = ""          align_footnote = "justify"      hidden_footnote = False
bold_footnote = False      italic_footnote = False      inverse_footnote = False
bg_footnote = -1           strike_footnote = False      blinking_footnote = False
fg_footnote = -1           dim_footnote = False          underline_footnote = False
```

`msg_footnote` The title name for the list. It only accepts string values, by default is empty.

`bold_footnote` It only accepts two int values 0 and 1, by defaults is set to 0.

`bg_footnote` and `fg_footnote` accepts int values from -1 to 256. Default values from the system are -1 and 256.

`align_footnote` It accepts 4 values, left (l), justify (j), center (c), and right (r).

```
#-----
# Data Section -
#-----
align_data = "justify"      hidden_data = False          inverse_data = False
bold_data = False           italic_data = False           blinking_data = False
bg_data = -1                strike_data = False           underline_data = False
fg_data = -1                dim_data = False             bg_all_cell_data = True
```

`bg_all_cell_data` The bg color will affect the entire cell or just the data.

`align_data` It accepts 4 values, left (l), justify (j), center (c), and right (r).

`bg_data` and `fg_data` accepts int values from -1 to 256. Default values from the system are -1 and 256.

```
#-----
# Horizontal Line Section -
#-----
top_horizontal_line_chr = "-"      bottom_horizontal_line_chr = "-"      middle_horizontal_line_chr = "-"
top_horizontal_line_on = True       bottom_horizontal_line_on = True       middle_horizontal_line_on = False
bold_horizontal_line = False        bg_horizontal_line = -1                fg_horizontal_line = -1
```

`middle_horizontal_line_on` These are the lines below the data. Check [Case 9:](#) for reference.

`bg_horizontal_line` and `fg_horizontal_line` accepts int values from -1 to 256. Default values from the system are -1 and 256.

For more reference check [Figure 1](#).

```
#-----
# Vertical Line Section -
#-----
bold_vertical_line = False          left_vertical_line_chr = "|"
bg_vertical_line = -1               middle_vertical_line_chr = "|"
fg_vertical_line = -1               right_vertical_line_chr = "|"
```

`middle_vertical_line_chr` → A string type. The char used to make the horizontal line. For more reference check [Figure 2](#).

`right_vertical_line_chr` → A string type. [Refer to Figure 1](#).

`left_vertical_line_chr` → A string type. [Refer to Figure 1](#).

`bg_vertical_line` and `fg_vertical_line` Accepts int values from -1 to 256. Default values from the system are -1 and 256.

```
#-----
# External Corner Section
#-----
top_left_corner_chr = "+"      bottom_right_corner_chr = "+"      bold_corner_chr = False
top_right_corner_chr = "+"     bottom_left_corner_chr  = "+"      bg_corner_chr   = -1
                                   fg_corner_chr    = -1
```

`top_left_corner_chr` A string type. For reference check Figure 1. By default set to “+”

`top_right_corner_chr` A string type. For reference check Figure 1. By default set to “+”

`bottom_right_corner_chr` A string type. For reference check Figure 1. By default set to “+”

`bottom_left_corner_chr` A string type. For reference check Figure 1. By default set to “+”

`bg_corner_chr` and `fg_corner_chr` Accepts int values from -1 to 256. Default values from the system are -1 and 256.

```
#-----
# Middle Corner Section
#-----
bold_inner_corner_chr = False    middle_top_corner_chr   = "+"    right_lateral_corner_chr = "+"
bg_inner_corner_chr   = -1       middle_inner_corner_chr = "+"    left_lateral_corner_chr  = "+"
fg_inner_corner_chr   = -1       middle_bottom_corner_chr = "+"
```

`bg_corner_chr` and `fg_corner_chr` Accepts int values from -1 to 256. Default values from the system are -1 and 256.

For reference check Figure 3 and 4.

```
#-----
# Header Section
#-----
align_header = "justify"    hidden_header = False    inverse_header = False
bold_header  = False        italic_header  = False    blinking_header = False
bg_header    = -1           strike_header  = False    underline_header = False
fg_header    = -1           dim_header    = False    bg_all_cell_header = True
```

`bg_all_cell_data` The bg color will affect the entire cell or just the header.

`align_header` It accepts 4 values, left (l), justify (j), center (c), and right (r).

`bg_header` and `fg_header` Accepts int values from -1 to 256. Default values from the system are -1 and 256.

### Attributes for the header lines

```
bold_vertical_header_line_chr = False    right_vertical_header_line_chr = "|"
bg_vertical_header_line_chr   = -1        left_vertical_header_line_chr  = "|"
fg_vertical_header_line_chr   = -1        middle_vertical_header_line_chr = "|"
```

For reference check Figure 3 and 4.

```
#-----
# Header Under Line Section -
#-----
```

### Attributes for the line below the header text

```
bold_under_line_header = False          horizontal_line_under_header_on = False
bg_under_line_header   = -1             horizontal_line_under_header_chr = "-"
fg_under_line_header   = -1
```

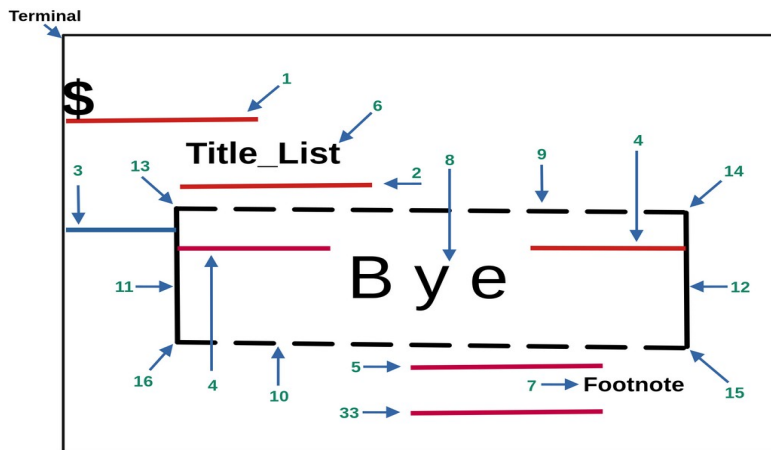
`horizontal_line_under_header_on` Horizontal lines between headers and the first data row.

`bg_under_line_header` and `fg_under_line_header` Accepts int values from -1 to 256. Default values from the system are -1 and 256.

### Attributes for the header corners (left, middles and right)

```
bold_corner_under_line_header = False    left_corner_line_under_header_chr  = "+"
bg_corner_under_line_header   = -1       right_corner_line_under_header_chr = "+"
fg_corner_under_line_header   = -1       middle_corner_line_under_header_chr = "+"
```

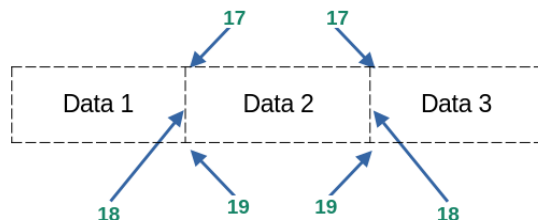
For more reference see figure 3.



**Figure 1**

1.- adj_top_margin	2.- top_space	3.- adj_indent
4.- adj_space	5.- bottom_space	6.- msg_title
7.- msg_footnote	8.- data	9.- top_horizontal_line_chr
10.- bottom_horizontal_line_chr	11.- left_vertical_line_chr	12.- right_vertical_line_chr
13.- top_left_corner_chr	14.- top_right_corner_chr	15.- bottom_right_corner_chr
16.- bottom_left_corner_chr	33.- adj_bottom_margin	

17.- middle_top_corner_chr
18.- middle_vertical_line_chr
19.- middle_bottom_corner_chr



**Figure 2**

20.- header
21.- horizontal_line_under_header_chr
22.- left_vertical_header_line_chr
23.- right_vertical_header_line_chr
24.- left_corner_line_under_header_chr
25.- right_corner_line_under_header_chr
26.- middle_horizontal_line_chr
27.- left_lateral_corner_chr
28.- right_lateral_corner_chr

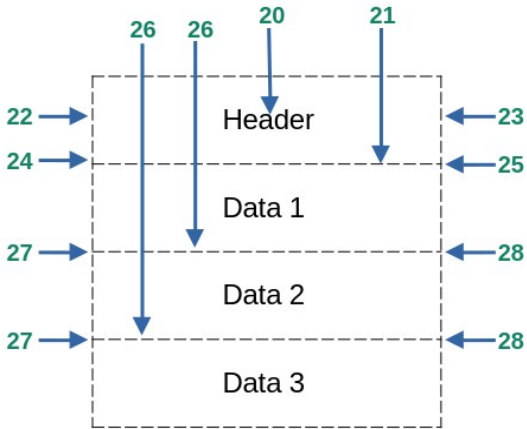


Figure 3

29.- middle_vertical_header_line_chr
30.- middle_corner_line_under_header_chr
31.- middle_inner_corner_chr
32.- set_fill_chr

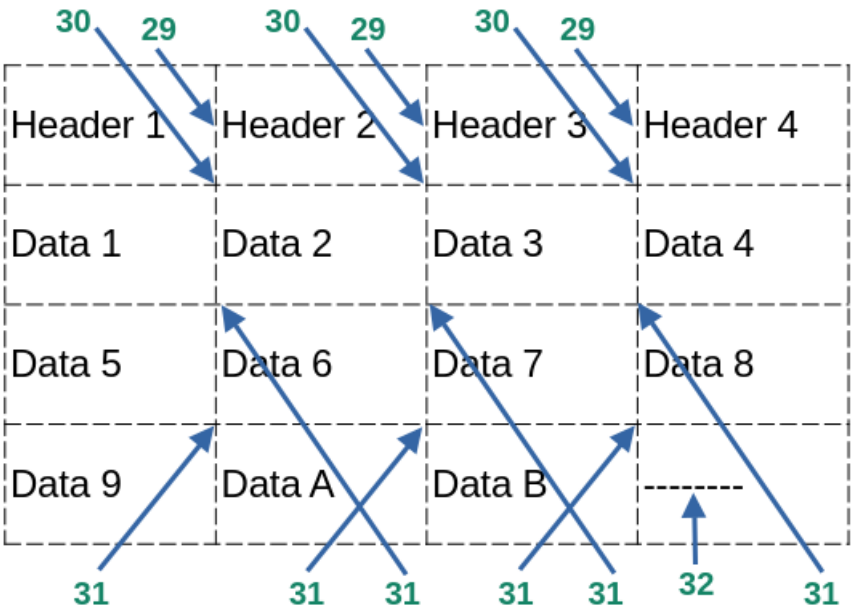
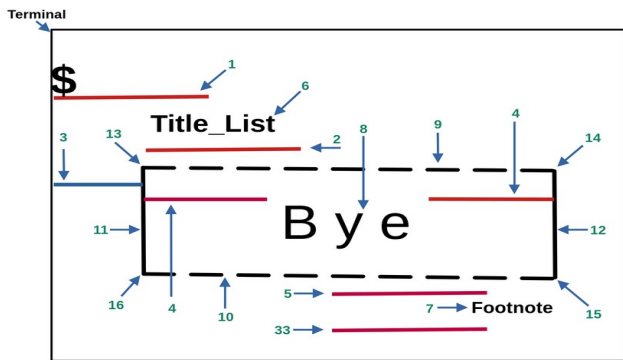


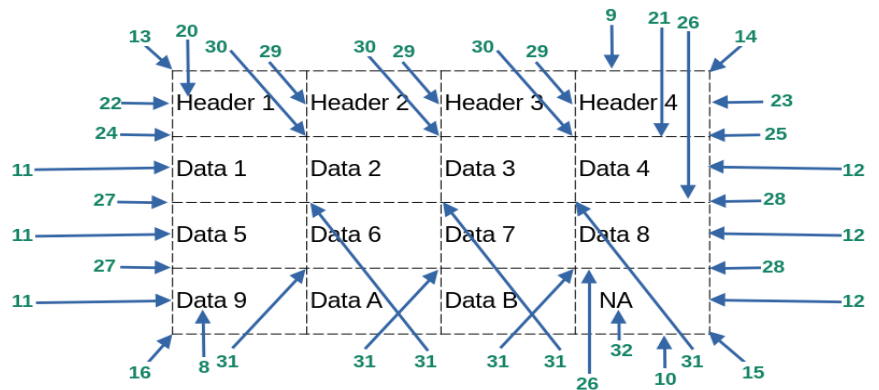
Figure 4

# Summarize

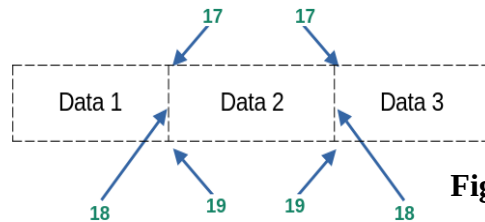


**Note:** 2 and 33 only work if the title and footnote exist.

**Figure 1**



**Figure 5**



**Figure 2**

1.- adj_top_margin	2.- top_space	3.- adj_indent
4.- adj_space	5.- bottom_space	6.- msg_title
7.- msg_footnote	8.- data	9.- top_horizontal_line_chr
10.- bottom_horizontal_line_chr	11.- left_vertical_line_chr	12.- right_vertical_line_chr
13.- top_left_corner_chr	14.- top_right_corner_chr	15.- bottom_right_corner_chr
16.- bottom_left_corner_chr	17.- middle_top_corner_chr	18.- middle_vertical_line_chr
19.- middle_bottom_corner_chr	20.- header	21.- horizontal_line_under_header_chr
22.- left_vertical_header_line_chr	23.- right_vertical_header_line_chr	24.- left_corner_line_under_header_chr
25.- right_corner_line_under_header_chr	26.- middle_horizontal_line_chr	27.- left_lateral_corner_chr
28.- right_lateral_corner_chr	29.- middle_vertical_header_line_chr	30.- middle_corner_line_under_header_chr
31.- middle_inner_corner_chr	32.- set_fill_chr	33. adj_bottom_margin

## Horizontal Line Default Values:

top\_horizontal\_line\_on = 1

middle\_horizontal\_line\_on = 0

bottom\_horizontal\_line\_on = 1

horizontal\_line\_under\_header\_on = 0

## bg\_all\_cell\_data/header Default Values:

bg\_all\_cell\_data = True

bg\_all\_cell\_header = True

## Some Other Default Values:

align\_title = "justify"

align\_footnote = "justify"

msg\_title = ""

msg\_footnote = ""

align\_data = "justify"

align\_header = "justify"

update\_list = False

set\_layout = Layout.HORIZONTAL

## Examples:

### Demo 1. Default Values

```
import fancyprint as fp

tlb = fp.FancyFormat()
lst = [
    ["Header 1", "Header 2", "Header 3", "Header 4"],
    ["R2C1", "R2C2", "R2C3", "R2C4"],
    ["R3C1", "R3C2", "R3C3", "R3C4"],
    ["R3C1", "R3C2"]
]

tlb.print_fancy_format(lst)
```

Header 1	Header 2	Header 3	Header 4
R2C1	R2C2	R2C3	R2C4
R3C1	R3C2	R3C3	R3C4
R4C1	R4C2	----	----

### Demo 2. A Little bit of Customization

```
import fancyprint as fp

tlb = fp.FancyFormat()
lst = [
    ["Header 1", "Header 2", "Header 3", "Header 4"],
    ["R2C1", "R2C2", "R2C3", "R2C4"],
    ["R3C1", "R3C2", "R3C3", "R3C4"],
    ["R4C1", "R4C2"]
]
```

```
tlb.msg_title = " Title "
tlb.align_title = fp.Align.CENTER
tlb.bold_title = True
tlb.fg_title = 21
tlb.bg_title = 231
```

```
tlb.bg_header = 90
tlb.fg_header = 231
tlb.horizontal_line_under_header_on = True
```

```
tlb.align_data = fp.Align.CENTER
tlb.fg_data = 14
```

```
tlb.msg_footnote = " Footnote "
tlb.align_footnote = fp.Align.RIGHT
tlb.bold_footnote = True
tlb.bg_footnote = 231
tlb.fg_footnote = 21
```

```
tlb.print_fancy_format(lst)
```

```
lst = [
    ["Header"], ["R2C1"], ["R3C1"], ["R4C1"]
]
```

```
tlb.print_fancy_format(lst, fp.Line_Style.SINGLE)
```

Title			
Header 1	Header 2	Header 3	Header 4
R2C1	R2C2	R2C3	R2C4
R3C1	R3C2	R3C3	R3C4
R4C1	R4C2	----	----
Footnote			

Title
Header
R2C1
R3C1
R4C1
Footnote

Demo 3 → Type of Variables

int

+-----+

| 2547 |

+-----+

Case 1

bool

+-----+

| True |

+-----+

Case 0

str

+-----+

| Hello There |

+-----+

Case 4

complex

+-----+

| 45.8+698.0j |

+-----+

Case 3

float

+-----+

| 25.987 |

+-----+

Case 2

complex

+-----+

| (45.9+25j) |

+-----+

Case 3

Range Data

+-----+

| 0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 |

+-----+

Case 5

Dictionary

+-----+

Keys

Values

NAME Miguel

LAST\_1 Aguilar

LAST\_2 Cuesta

+-----+

Case 6

Range Data

+-----+

Header

+-----+

0

2

4

6

8

10

12

14

+-----+

Case 5

Demo 4. Some More Customization

Header 1	Header 2	Header 3	Header 4
Data 1	Data 2	Data 3	Data 4
Data 5	Data 6	Data 7	Data 8
Data 9	Data A	Data B	----

## Demo 5. Two List Joined

Header 1	Header 2	Header 3	Header 4
Data 1	Data 2	Data 3	Data 4
Data 5	Data 6	Data 7	Data 8
Data 9	Data A	Data B	----

Header 1	Header 2	Header 3	Header 4
Data 1	Data 2	Data 3	Data 4
Data 5	Data 6	Data 7	Data 8
Data 9	Data A	Data B	----

## Demo 6. Some More Customization

Header 1	Header 2	Header 3	Header 4
Data 1	Data 2	Data 3	Data 4
Data 5	Data 6	Data 7	Data 8

Header 1	Header 2	Header 3	Header 4
Data 1	Data 2	Data 3	Data 4
Data 5	Data 6	Data 7	Data 8

Header 1	Header 2	Header 3	Header 4
Data 1	Data 2	Data 3	Data 4
Data 5	Data 6	Data 7	Data 8

```
import fancyprint as fp
tbl = fp.FancyFormat()
lst = [[ "Header 1"  "Header 2",  "Header 3"  ],
       [ "Data 1",   "Data 2",    "Data 3"   ],
       [ "Data 4",   "Data 5",    "Data 6"   ]]
```

# Colors

```
tbl.bg_horizontal_line = 21
```

```
tbl.bg_vertical_line = 21
```

```
tbl.bg_corner_chr = 21
```

```
tbl.bg_inner_corner_chr = 21
```

```
tbl.bg_under_line_header = 21
```

```
tbl.bg_corner_under_line_header = 21
```

```
tbl.bg_vertical_header_line_chr = 21
```

```
tbl.bg_header = 90
```

```
tbl.fg_header = 231
```

```
tbl.bold_header = True
```

```
tbl.bg_data = 231
```

```
tbl.fg_data = 0
```

```
tbl.bold_data = True
```

```
tbl1.adj_top_margin = 2
```

```
tbl1.print_fancy_format(data=lst, style=fp.Line_Style.NONE)
```

```
tbl1.print_fancy_format(data=lst, style=fp.Line_Style.DOUBLE_SPACE_COL_COLOR)
```

```
tbl1.print_fancy_format(data=lst, style=fp.Line_Style.NO_SPACE_COL_COLOR)
```

```
tbl1.horizontal_line_under_header_on = True
```

```
tbl1.middle_horizontal_line_on = True
```

```
tbl1.print_fancy_format(data=lst, style=fp.Line_Style.NONE)
```

```
tbl1.print_fancy_format(data=lst, style=fp.Line_Style.DOUBLE_SPACE_COL_COLOR)
```

```
tbl1.print_fancy_format(data=lst, style=fp.Line_Style.NO_SPACE_COL_COLOR)
```



Header 1	Header 2	Header 3
Data 1	Data 2	Data 3
Data 4	Data 5	Data 6

Header 1	Header 2	Header 3
Data 1	Data 2	Data 3
Data 4	Data 5	Data 6

Header 1	Header 2	Header 3
Data 1	Data 2	Data 3
Data 4	Data 5	Data 6

Header 1	Header 2	Header 3
Data 1	Data 2	Data 3
Data 4	Data 5	Data 6

Header 1	Header 2	Header 3
Data 1	Data 2	Data 3
Data 4	Data 5	Data 6

Header 1	Header 2	Header 3
Data 1	Data 2	Data 3
Data 4	Data 5	Data 6

## Pen Class

This class contains two methods:

```
draw_line(size=0, layout=Layout.HORIZONTAL, tail="\N{BLACK DIAMOND}", body="-", head="\N{BLACK DIAMOND}")
```

```
draw_rectangle(length=3, width=3, style=Line_Style.DASH)
```

### Rectangle Default Values

top_left_corner_chr	= "+"	top_horizontal_line_chr	= "-"	right_vertical_line_chr	= " "
top_right_corner_chr	= "+"	bottom_horizontal_line_chr	= "-"	left_vertical_line_chr	= " "
bottom_right_corner_chr	= "+"				
bottom_left_corner_chr	= "+"	refill_bg_color	= False		

### Line Default Values

bold_draw_line	= False
bg_draw_line	= -1
fg_draw_line	= -1

### General Default Values

adj_indent	= 0
------------	-----

### Example:

```
import fancyprint as fp
pen = fp.Pen()
pen.adj_indent = 8
pen.draw_line(size=20, layout=fp.Layout.HORIZONTAL, tail=fp.Unicode.BLACK_LEFT_POINTING_TRIANGLE,
              body=fp.Unicode.EM_DASH, head=fp.Unicode.BLACK_RIGHT_POINTING_TRIANGLE)

fp.ins_newline(2)
pen.adj_indent = 14
pen.draw_rectangle(length=8, width=4, style=fp.Line_Style.DOUBLE)
```



Report bugs at → [acma.mex@hotmail.com](mailto:acma.mex@hotmail.com)

FanyPrint module is not a big thing, but I hope you find useful occasionally. Python 3.12.1 or greater is required.

**Note:** fancyprint module has been tested on RedHat 9, Centos Stream 9, AlmaLinux 9, and Windows 10.

[https://github.com/acma82/Fancy\\_Print/](https://github.com/acma82/Fancy_Print/)

Saturday November 16, 2024.