Formatting

Fall 2023

Print Formatting

- Python's print statement can be adapted for personal use through adding a few variables to send to the print function
  - end - this variable is default set to a newline, but you can change it; that's why each print statement takes up its own line
  - sep - this variable is default set to a space; formats adding multiple arguments to a print statement

```
print("One", "Two", "Three")
print("One", "Two", "Three", sep="*")
```

```
One Two Three
One*Two*Three
```

```
print("Hello", end=" ")
print("It is me")

print("Hello")
print("It is me")
```

```
Hello It is me
Hello
It is me
```

Print Formatting Notes:

- When using “end” or “sep” to alter your print statements, they must *always* contain the name of the variable and an equal sign setting it to what you want
  - print(“Hello”, “ “)
    - This can’t tell whether you want to use “end” or “sep”, you have to tell it
  - print(“Hello”, end” “)
    - This will throw an error, you must have an assignment statement
- end and sep are *case sensitive*

Escape Characters

- Python also includes special characters you can use
  - These characters can be used in “end” and “sep” as well
- \ is the escape character
  - If you want to print quotations then you’ll need to escape quotations
  - Pay attention that you still have opening and closing quotations for your strings!

Table 2-8 Some of Python’s escape characters

Escape Character	Effect
\n	Causes output to be advanced to the next line.
\t	Causes output to skip over to the next horizontal tab position.
\'	Causes a single quote mark to be printed.
\"	Causes a double quote mark to be printed.
\\	Causes a backslash character to be printed.

Escape Characters

```
print("One", "Two", "Three", sep="\t")

print("\nI once read a book called \"To Kill a Mockingbird\"")

print("If I type two back slashes, it only shows one: \\")
print("If I type four back slashes, it only shows two: \\\\")
```

Escape Characters

```
print("One", "Two", "Three", sep="\t")

print("\nI once read a book called \"To Kill a Mockingbird\"")

print("If I type two back slashes, it only shows one: \\")
print("If I type four back slashes, it only shows two: \\\\")
```

```
One     Two     Three

I once read a book called "To Kill a Mockingbird"

If I type two back slashes, it only shows one: \
If I type four back slashes, it only shows two: \\
```

Format Function

- Python has a built in format() function
  - format() takes in two arguments: item to format and how it's being format
  - Item to format can be a variable or direct information
  - How it's being format must be contained in quotations
  - Data types still apply -- you can't format a string like an integer, etc
- You can print the contents of a format function or assign it to a variable

Format Code	Description
d	Integer
f	Floating Point
s	String
%	Percents
,	Adds Commas
<	Left Align
>	Right Align
^	Center Align

Format Examples

Example	Output
format(12.344444444, ".2f")	12.34
format("Hello", "^15s")	" Hello "
format(123456, ",d")	123,456
format(0.567345, "%")	56.734500%
format(0.567345, "10.0%")	" 57%"
format("Hello", ">10s")	" Hello"

Notice how format will round up

Quotations aren't included in the actual output … this is to show that it prints hello within a 15 space interval and centers it within that

Formatting Notes:

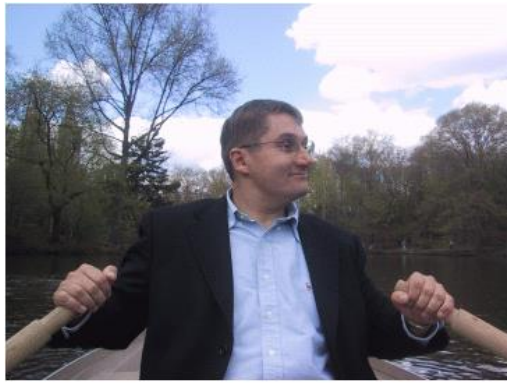
- Numbers (floats, decimals, percents) are automatically right aligned whereas strings are automatically left aligned
- All math should be done on numbers *before* formatting to ensure your calculations are as correct as possible

```
formattedInt = format(123456789, ",d")

print(formattedInt)
```

```
123,456,789
```

March of Progress



The March of Progress

1980: C

```
printf("%10.2f", x);
```

1988: C++

```
cout << setw(10) << setprecision(2) << showpoint << x;
```

1996: Java

```
java.text.NumberFormat formatter = java.text.NumberFormat.getNumberInstance();
formatter.setMinimumFractionDigits(2);
formatter.setMaximumFractionDigits(2);
String s = formatter.format(x);
for (int i = s.length(); i < 10; i++) System.out.print(' ');
System.out.print(s);
```

2004: Java

```
System.out.printf("%10.2f", x);
```

2008: Scala and Groovy

```
printf("%10.2f", x)
```

(Thanks to Will Iverson for the update. He writes: "Note the lack of semi-colon. Improvement!")

Another way to control decimals

- Similar to printf statement from C and Java
- Start with the format operator (%) end with the data type specifier (f for floats)

```
a = 3.14159
print("Pi = %.2f" % (a))
```

- Can also be used to form tables with multiple print statements using minimum field width

```
print("%10.2f %10.2f %10.2f" %(3.491, 2.9742, 4.1))
print("%10.2f %10.2f %10.2f" %(28.49, 942.7471, 400.61))
```