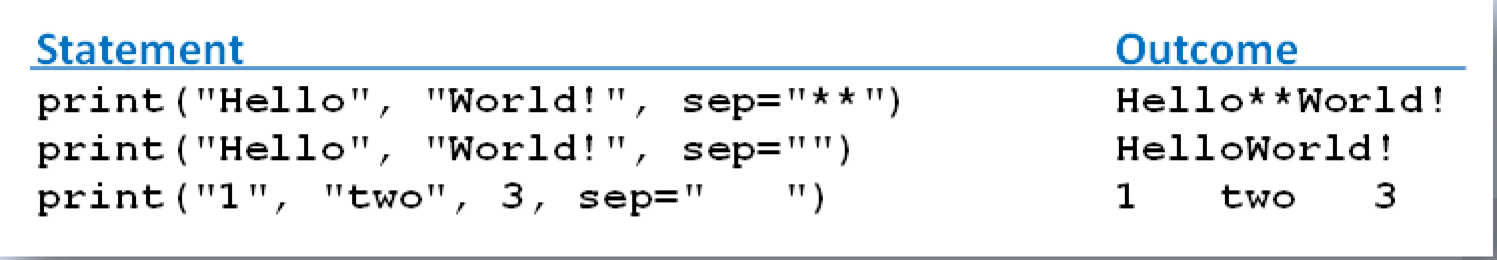
# Output Features
## Optional print Argument **`sep`**

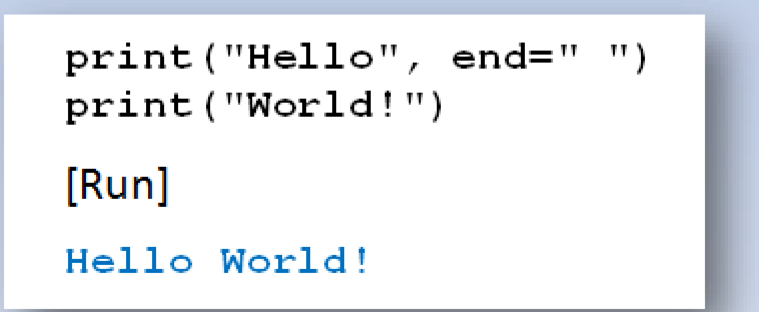
- Consider statement
  **`print(value0, value1, …, valueN)`**
- Print function uses string consisting of one space character as separator
- Optionally change the separator to any string we like with the **`sep`** argument



| Statement | Outcome |
|---|---|
| `print("Hello", "World!", sep="**")` | `Hello**World!` |
| `print("Hello", "World!", sep="")` | `HelloWorld!` |
| `print("1", "two", 3, sep="    ")` | `1    two    3` |

# Optional print Argument **`end`**

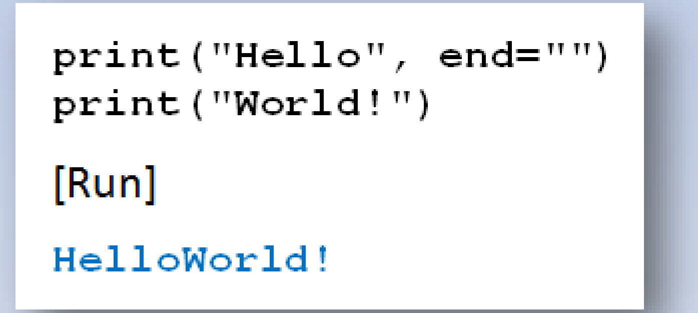
- Print statement ends by executing a newline operation.
- Optionally change the ending operation with the **`end`** argument



```
print("Hello", end=" ")
print("World!")
```

[Run]

```
Hello World!
```



```
print("Hello", end="")
print("World!")
```

[Run]

```
HelloWorld!
```

# Escape Sequences

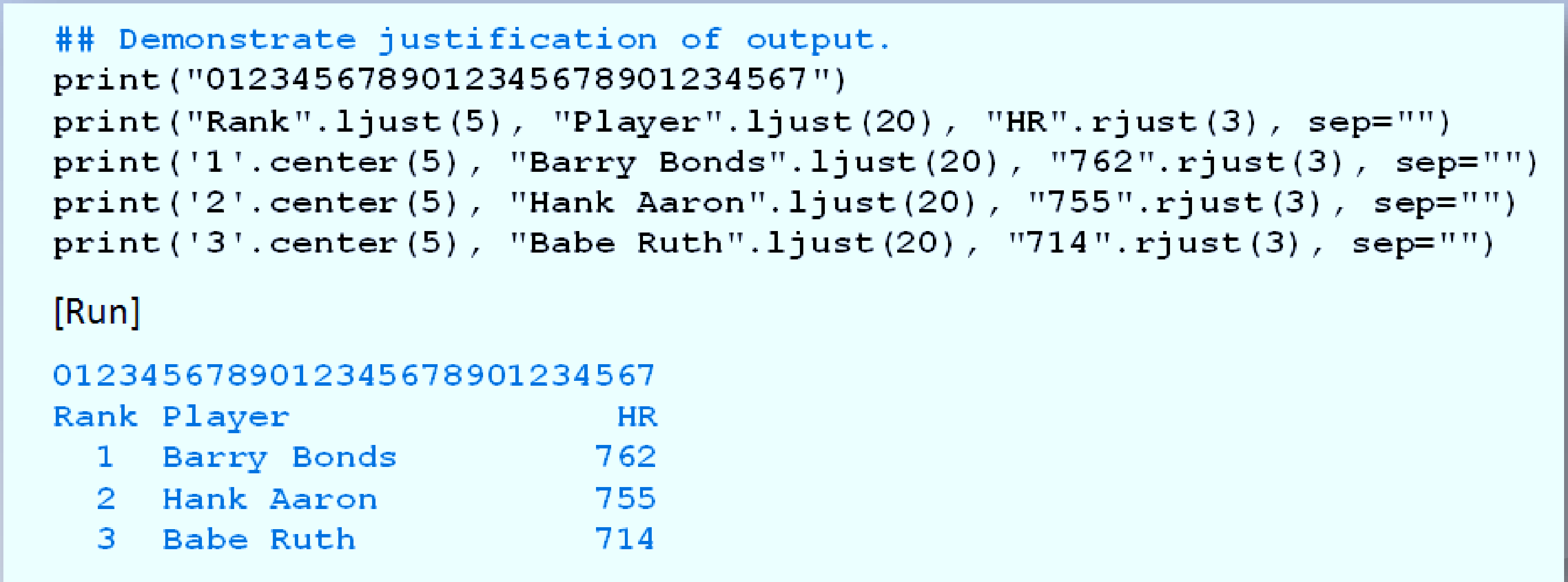
- Short sequences placed in strings
  - Instruct cursor or permit some special characters to be printed.
  - First character is always a backslash (\).
- \t induces a horizontal tab
- \n induces a newline operation

# Escape Sequences

- Backslash also used to treat quotation marks as ordinary characters.
- \”   causes print function to display double quotation mark
- \\   causes print function to display single backslash

# Justifying Output in a Field

- Example: Program demonstrates methods **`ljust(n)`**, **`rjust(n)`**, and **`center(n)`**



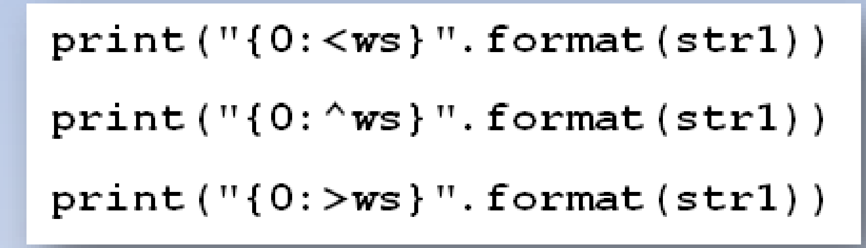
```
## Demonstrate justification of output.
print("0123456789012345678901234567")
print("Rank".ljust(5), "Player".ljust(20), "HR".rjust(3), sep="")
print('1'.center(5), "Barry Bonds".ljust(20), "762".rjust(3), sep="")
print('2'.center(5), "Hank Aaron".ljust(20), "755".rjust(3), sep="")
print('3'.center(5), "Babe Ruth".ljust(20), "714".rjust(3), sep="")
```

[Run]

```
0123456789012345678901234567
Rank Player               HR
  1  Barry Bonds         762
  2  Hank Aaron          755
  3  Babe Ruth           714
```

# Justify Output with `format`

- Given: *str1* is a string and *w* is a field width

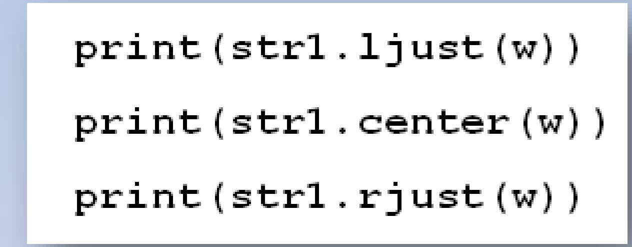


```
print("{0:<ws}".format(str1))

print("{0:^ws}".format(str1))

print("{0:>ws}".format(str1))
```

⟺



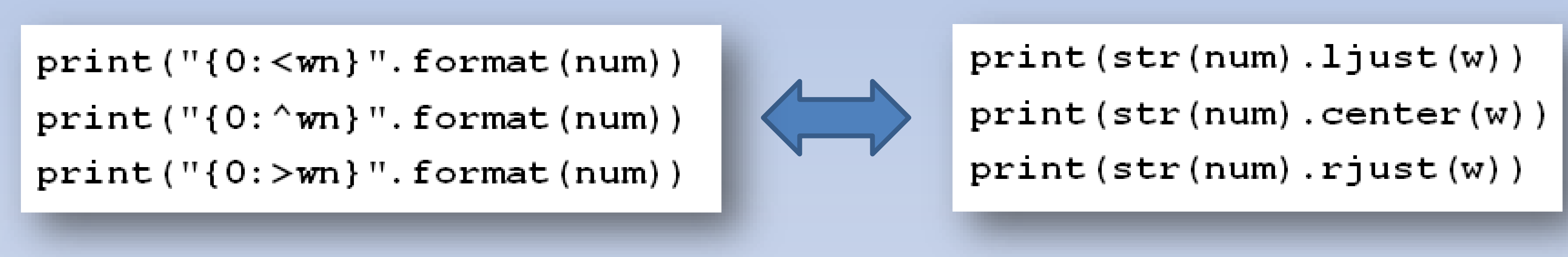
```
print(str1.ljust(w))

print(str1.center(w))

print(str1.rjust(w))
```

# Justify Output with `format`

- Given: *num* is a number and *w* is a field width



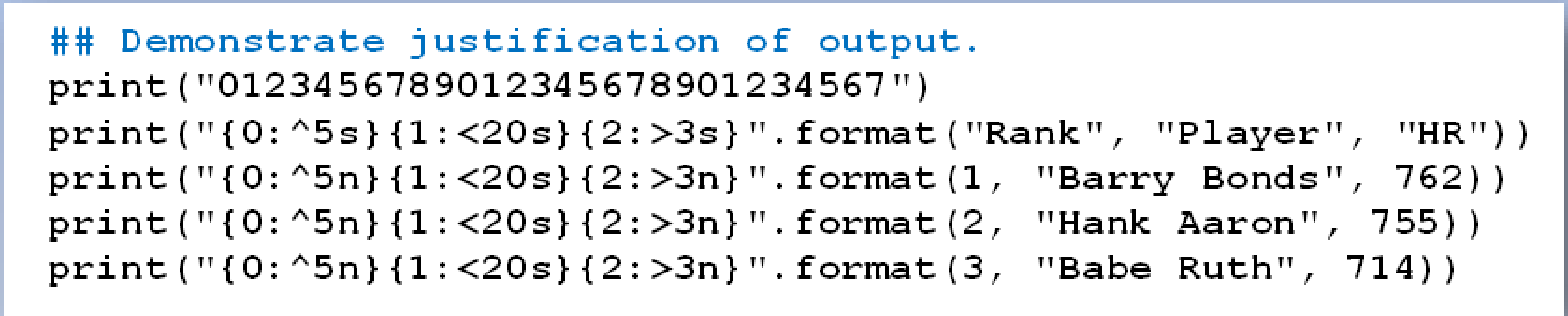
```
print("{0:<wn}".format(num))
print("{0:^wn}".format(num))
print("{0:>wn}".format(num))
```

⟺

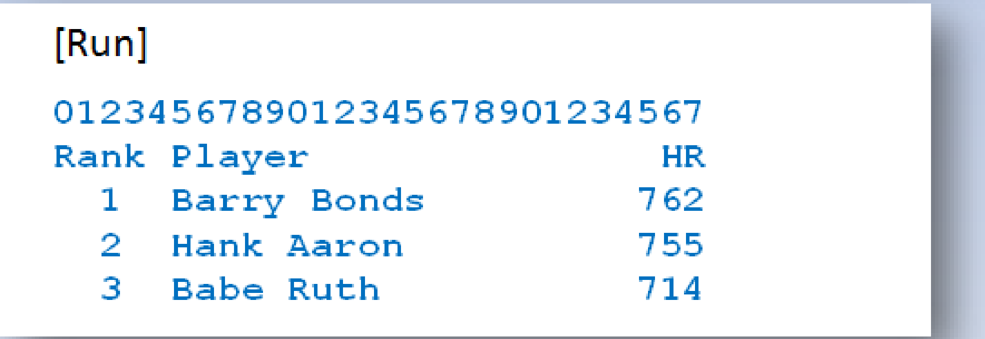
```
print(str(num).ljust(w))
print(str(num).center(w))
print(str(num).rjust(w))
```

# Justify Output with `format`

- Example: Program illustrates formatting



```
## Demonstrate justification of output.
print("0123456789012345678901234567")
print("{0:^5s}{1:<20s}{2:>3s}".format("Rank", "Player", "HR"))
print("{0:^5n}{1:<20s}{2:>3n}".format(1, "Barry Bonds", 762))
print("{0:^5n}{1:<20s}{2:>3n}".format(2, "Hank Aaron", 755))
print("{0:^5n}{1:<20s}{2:>3n}".format(3, "Babe Ruth", 714))
```

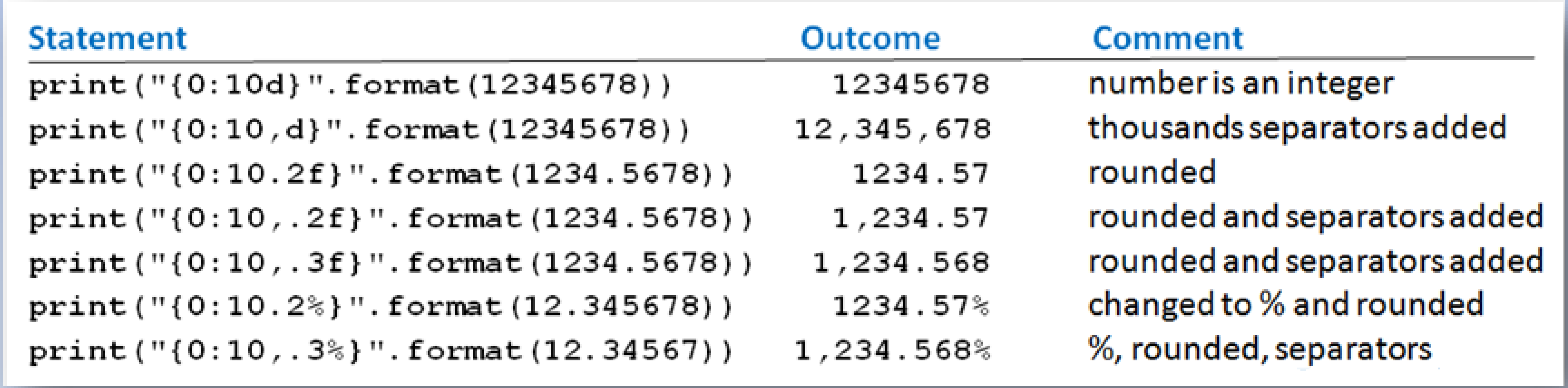


[Run]

```
0123456789012345678901234567
Rank Player               HR
  1  Barry Bonds         762
  2  Hank Aaron          755
  3  Babe Ruth           714
```

# Justify Output with `format`

- Demonstrate number formatting.



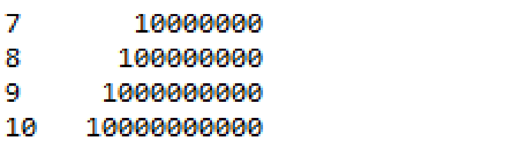
| Statement | Outcome | Comment |
|---|---|---|
| `print("{0:10d}".format(12345678))` | `12345678` | number is an integer |
| `print("{0:10,d}".format(12345678))` | `12,345,678` | thousands separators added |
| `print("{0:10.2f}".format(1234.5678))` | `1234.57` | rounded |
| `print("{0:10,.2f}".format(1234.5678))` | `1,234.57` | rounded and separators added |
| `print("{0:10,.3f}".format(1234.5678))` | `1,234.568` | rounded and separators added |
| `print("{0:10.2%}".format(12.345678))` | `1234.57%` | changed to % and rounded |
| `print("{0:10,.3%}".format(12.34567))` | `1,234.568%` | %, rounded, separators |

# Alternate formatting

- More like the C language format specifier
- Simpler?
- Format specifier starts with %
  - %d for integers
    - %2d %3d
  - %f for floating point numbers
    - %6.2f %8.4f
  - %s for strings
    - %20s %-10s

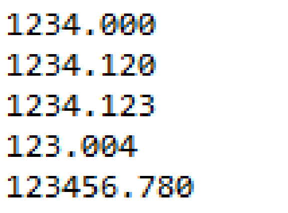
# Alternate formatting

- Examples
  - print(“%6s” % “four”)        # right justify
  - print(%-6s” % “four”)        # left justify
  - print(“%-3d %12d” % (exponent, 10 ** exponent))



```
7       10000000
8      100000000
9     1000000000
10   10000000000
```

  - print(“%6.3f” % amount)
    - amount is 1234, 1234.12, 1234.1234, 123.00375, 123456.78



```
1234.000
1234.120
1234.123
123.004
123456.780
```