W02_07_Strings_and_Formatted_Output

September 12, 2022

1 Strings and Formatted Output

1.0.1 Reminder

[1]:
```
s = "abc" # equivalent
s = 'abc' # but not s = 'abc"
```

[2]:
```
s.startswith("ab")
```

[2]:
```
True
```

1.0.2 Formatting strings

There are several ways of formatting strings in Python. The resulting strings can be printed to the screen or written to a file, or used for some other purpose – the formatting works the same regardless.

1.0.3 Formatting strings using %

The string interpolation operator % is probably still the most common in the wild.

The % operator is used *twice* in formatting.

[3]:
```
for i in range(5):
    if i % 2 == 0: # arithmetic % (mod)
        # %d is a format specifier
        # bare % is string interpolation
        s = "i = %d" % i
        print(s)
```

```
i = 0
i = 2
i = 4
```

We can use different % format strings to format values of different types:

[4]:
```
import math
"%d, %f, %.2f: %s" % (17, math.pi,
                      math.pi, "hello")
```

[4]:
```
'17, 3.141593, 3.14: hello'
```

1.0.4 Formatting strings using `.format()`

[7]:
```
for i in range(10):
    if i % 2 == 0:
        print("i = {}".format(i))
```

```
i = 0
i = 2
i = 4
i = 6
i = 8
```

1.0.5 Formatting strings using `f`-strings

I think modern Python authors prefer `f`-strings.

[6]:
```
for i in range(10):
    if i % 2 == 0:
        s = f"i = {i}" # special syntax f" ... {variable name} ... {or␣
↪expression}"
        print(s)
```

```
i = 0
i = 2
i = 4
i = 6
i = 8
```

1.0.6 Further reading

More examples and details, e.g.: https://realpython.com/python-f-strings/