CISS450: Artificial Intelligence
Lecture 15: String Formatting

Yihsiang Liow

Agenda

- Study the string formatting operator %

String Formatting

- Python supports C-style string formatting. (From your readings of the main text and the Python documentation, you should know by now that Python is closely related to C. In fact the Python interpreter is written in C.)
- Check Python's documentation (for 3.7).

% and %s

- Try the following:

```
s = "%s, World!" % "Hello"
t = "%s%s %s%s" % ("Hello", ",", "World", "!")
print(s, type(s))
print(t, type(t))
```

- Var `s`: `"Hello"` is substituted into the string `"%s, World!"` at substring `"%s"`.
- Var `t`: there are four strings to to be substituted into the string `"%s%s %s%s"`. These four strings are placed in a tuple.

% and %s

- Note that in all cases, % is the substitution operator and the object is inserted into the string where there is a `"%s"` substring. If there are more than one object to be inserted, then place them in a tuple:

```
    <string> % <tuple>
```

- Remember that the above gives a ***string***

% and %s

- If a variable is not a string, %s will try to “string” it up for you:

```
i = 123
s = "...%s..." % i
print(s)
print("%s" % [1,2,"a"])
print("%s" % None)
print("%s" % 1.23)
```

- (Actually objects can have a special function/method that converts them to strings. The % operator calls these functions for you.)

%d, %f and %e

- Besides %s there are many other formatting substrings. Try these:

```
print("%d" % 123.456)
print("%f" % 12)
print("%e" % 12)
```

Width and Precision

- You can also specify total ***width*** and ***precision***:

```
print("%12s" % "abcdef")
print("%12.3f" % 12.3456)
print("%12.3f" % 12)
print("%12.4f" % 12.3456)
```

- Can you tell what substring `"12.3"` do?

Padding

- You can pad with zeroes:

```
print("%012d" % 123.456789)
print("%012.3d" % 123.456789)
```

- Can you tell what causes the padding?

Alternatives

- Instead of string formatting operations, you can also use string concatenation. For instance, the following gives the same string:

```
x,y,z = 1,2,"buckle my shoe"
s0 = "%s, %s, %s" % (x,y,z)
s1 = str(x) + ", " + str(y) + ", " + z
print("[%s]" % s0)
print("[%s]" % s1)
```

Fixed-width Columns & String Float

- However, the formatting operations are very useful for formatting fixed column outputs and creating float as strings in a specific format. Here's such an example:

```
import math
for i in range(10):
    print("%5d  %10.3f" % (i,math.sqrt(i)))
```

Justification

- Try the following

```
print("%12s" % "abcdef")
print("%-12s" % "abcdef")
print("%12.4f" % 12.3456)
```

  - `print("%-12.3f" % 12.3456)`

Dictionary

- You can get the value of the key-value of a dictionary

```
print("[%(John)s]" % {"John":"Doe",
                      "Jane":"Smith"})
print("[%(John)12s]" % {"John":"Doe",
                        "Jane":"Smith"})
```

- Note that the key must be a string:

```
print("[%(1)12s]" % {1:"Doe"})
```

Python3 string format

- In python3, there's a new way to format strings.
- Try:

```
print("hello {} ... {}".format("world", "!"))
```

- There's more to it to the format function.
- See python documentation.