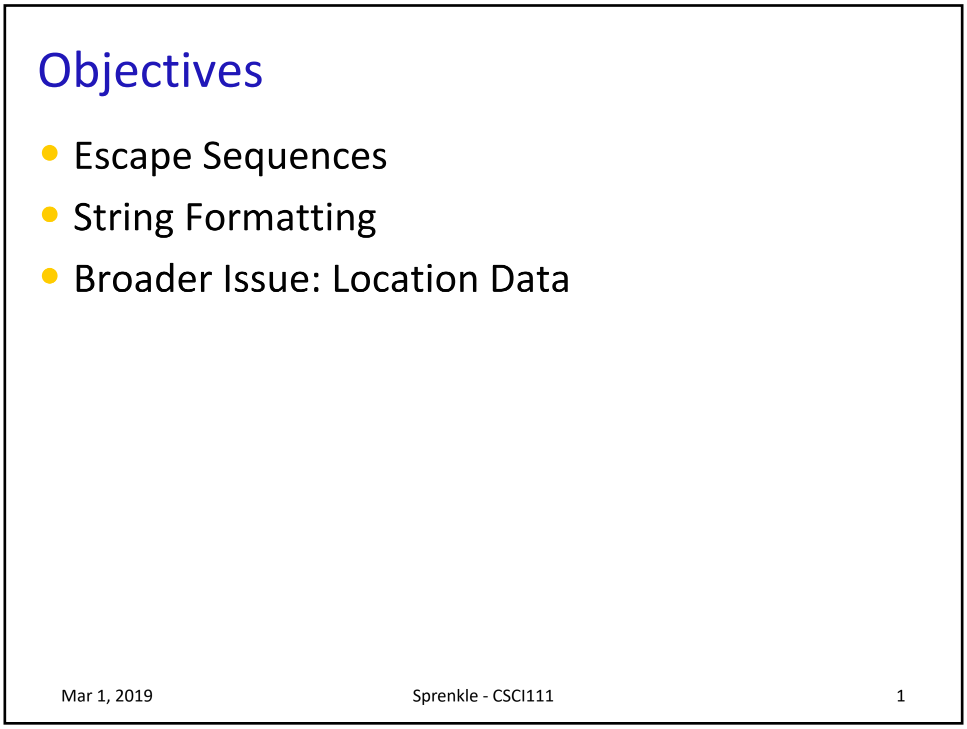
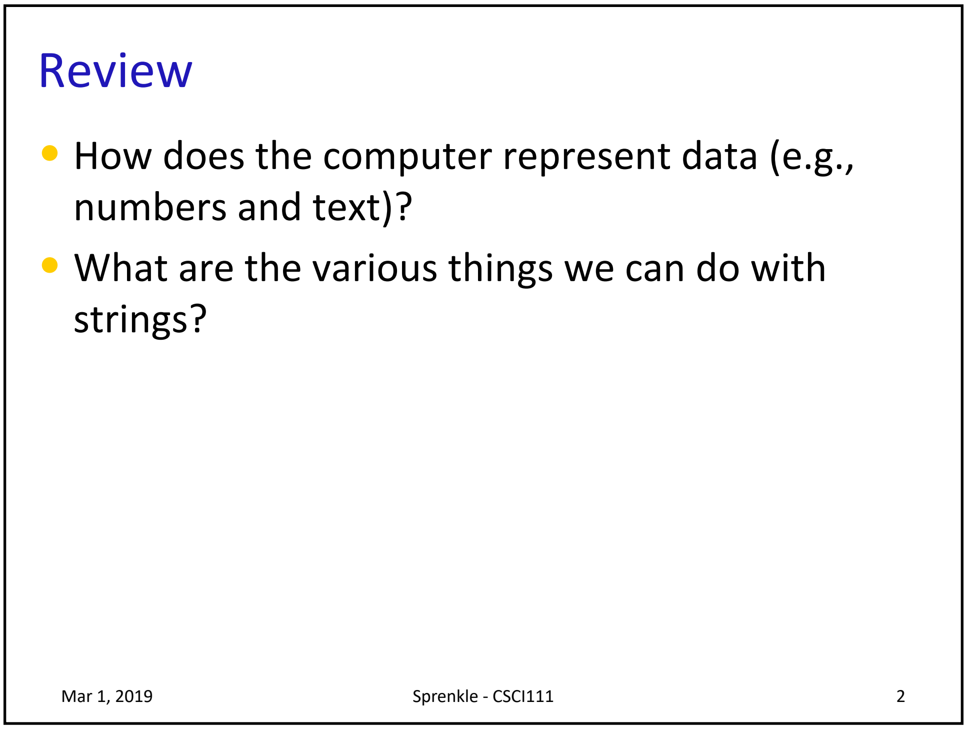


Objectives

- Escape Sequences
- String Formatting
- Broader Issue: Location Data



Review

- How does the computer represent data (e.g., numbers and text)?
- What are the various things we can do with strings?

Usability

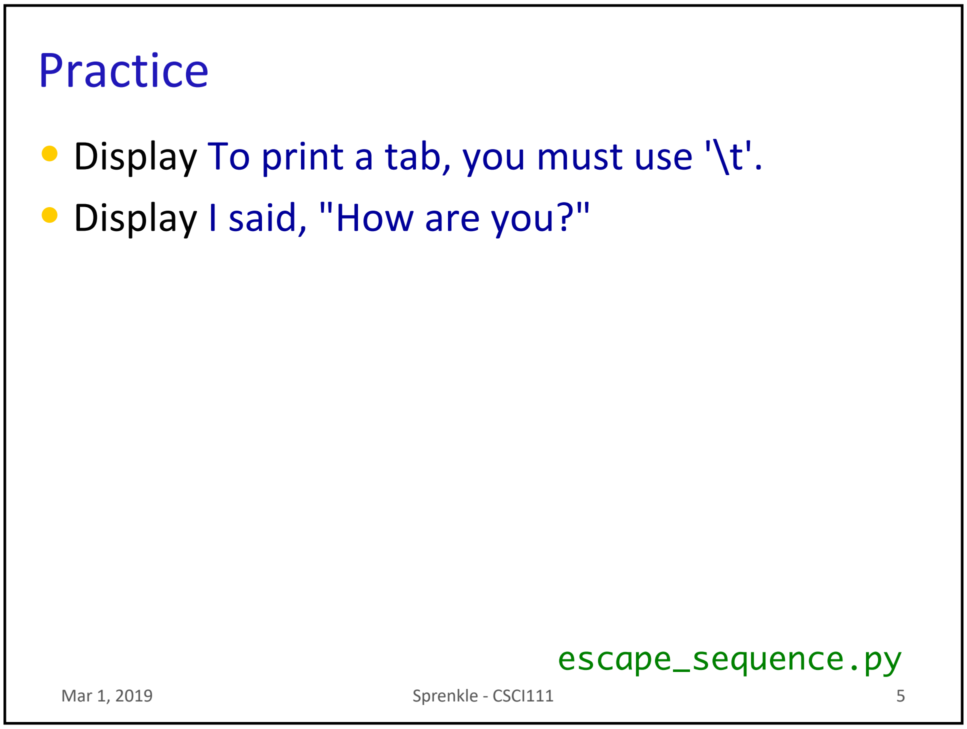
- Want users to *like* to use your software
  - More revenue
  - Develop even better software
- How Apple makes money:
  best user interfaces → user buys products

Escape Sequences

- Escape character: `\`
- Escape sequences
  - newline character (carriage return) → `\n`
  - tab → `\t`
  - quote → `\"` or `\'`
  - backslash → `\\`
- Example:
  - `print("To print a \\, you must use \"\\\\\"")`
    - What does this display?

Interactive demonstration

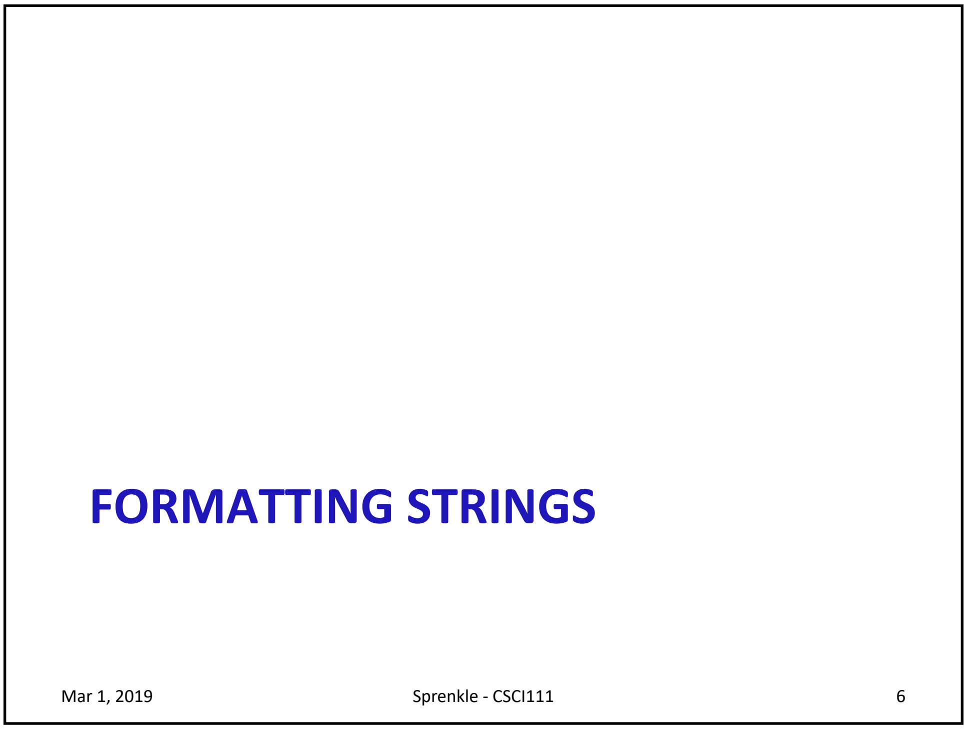
demo_str.py



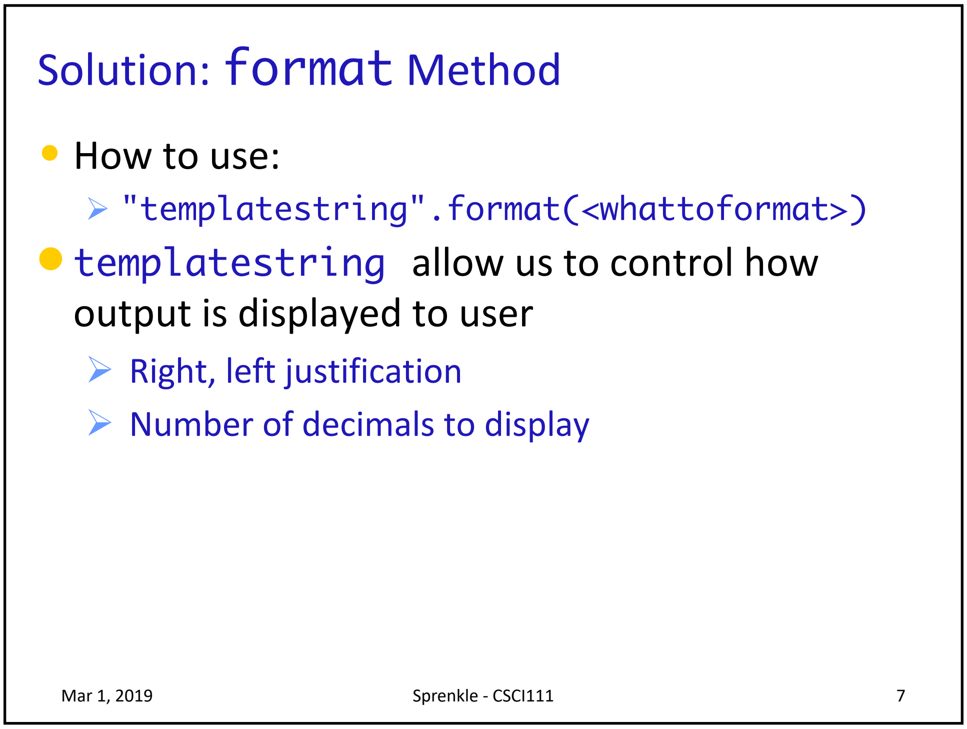
Practice

- Display To print a tab, you must use '\t'.
- Display I said, "How are you?"

escape_sequence.py

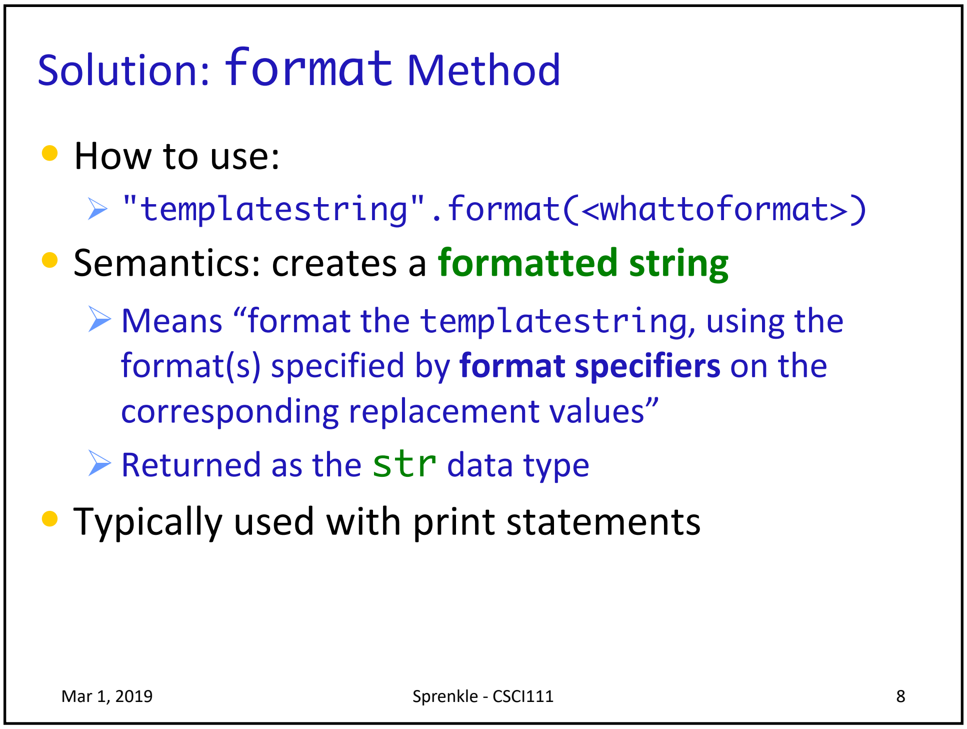


FORMATTING STRINGS



Solution: `format` Method

- How to use:
  - `"templatestring".format(<whattoformat>)`
- `templatestring` allow us to control how output is displayed to user
  - Right, left justification
  - Number of decimals to display



Solution: `format` Method

- How to use:
  - `"templatestring".format(<whattoformat>)`
- Semantics: creates a **formatted string**
  - Means “format the `templatestring`, using the format(s) specified by **format specifiers** on the corresponding replacement values”
  - Returned as the `str` data type
- Typically used with print statements

Formatting Strings

- `templatestring` is a template for the resulting string with format specifiers instead of the values
  - For each format specifier in templatestring, should have a `replacement value`
  - Throws `IndexError` if not enough replacements for specifiers in `templatestring`

```
"{:.2f}".format(3.14159)
```

Evaluates to `"3.14"`

One format specifier in template string

Corresponding replacement value

Format Specifiers

[ ] mean optional

- General format:
  `{[field_name]:conversion}`

index number of the argument, i.e., which field in the template string

- `conversion`
  - conversion code of the data type

Code	Type
s	string
d	integer
f	float
e	floating point with exponent

Default if code isn't given

(There are more…)

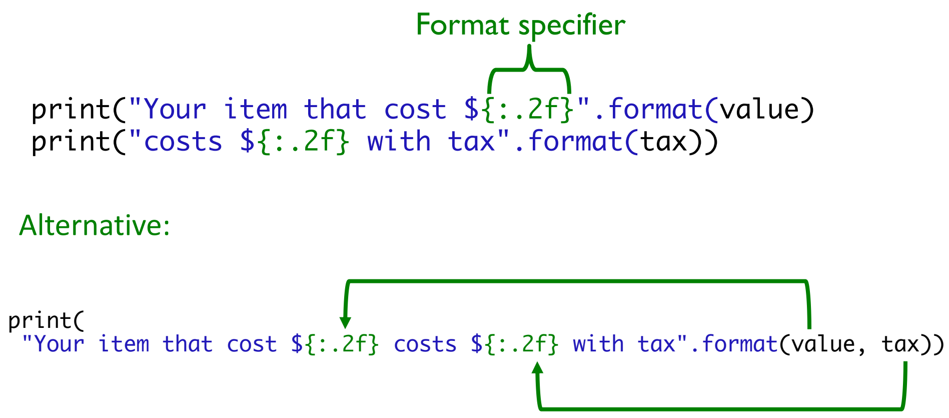
Format Specifiers

[ ] mean optional

Conversion options `:[flags][width][.precision][code]`

- flags:
  - 0: zero fills
  - +: adds a + sign before positive values
  - <: left-justification (default for strings)
  - >: right-justify (default for numbers)
  - ^: centered
- width:
  - *Minimum* number of character spaces reserved to display the entire value
  - Includes decimal point, digits before and after the decimal point and the sign
- precision:
  - Number of digits after the decimal point for **floating point** values

Example using Format Operator



```
print("Your item that cost ${:.2f}".format(value)
print("costs ${:.2f} with tax".format(tax))
```

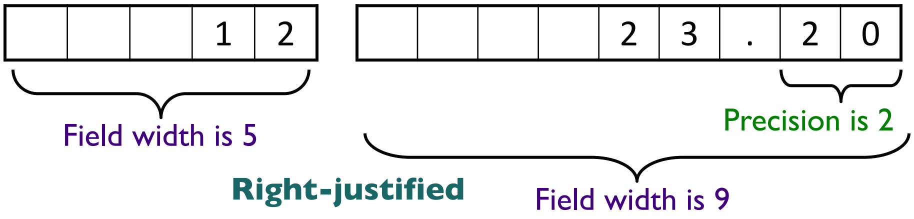
Alternative:

```
print(
 "Your item that cost ${:.2f} costs ${:.2f} with tax".format(value, tax))
```

`sales_tax2.py`

Example Format Specifiers

```
"{:5d}".format(12)   "{:9.2f}".format(23.1999)
   → "   12"            → "     23.20"
```

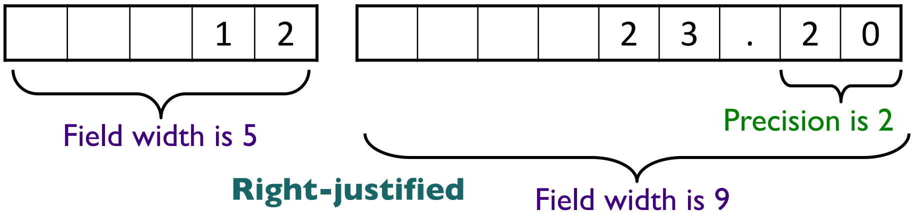


- What if precision is bigger than the decimal places?
- What if field width is smaller than the length of the value?

Any guesses? Try out in interpreter.

Example Format Specifiers

```
"{:5d}".format(12)   "{:9.2f}".format(23.1999)
   → "   12"            → "     23.20"
```



- What if precision is bigger than the decimal places?
  - Fills decimal with 0s
- What if field width is smaller than the length of the value?
  - String contains entire value

Formatting Practice

- `x = 10`
- `y = 3.5`
- `z = "apple"`
- `"{:6d}".format(x)`
- `"{:6.2f}".format(x)`
- `"{:6.2f}".format(y)`
- `"{:06.2f}".format(y)`
- `"{:^10s}".format(z)`
- `"{:5d} {:<7.3f}".format(x,y)`

Example: Printing Out Tables

- A table of temperature conversions

```
Temp F      Temp C      Temp K
------      ------      ------
-459.7      -273.1         0.0
   0.0       -17.8       255.2
  32.0         0.0       273.1
```

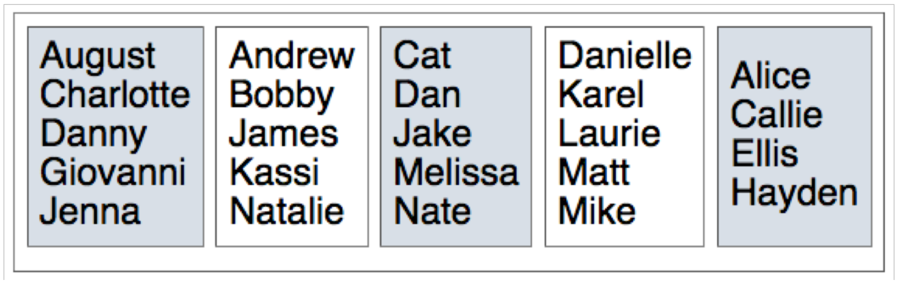
- If we want to print data in rows, what is the template for what a row looks like?
  - How do we make the column labels line up?

`temp_table.py`

String Formatting Note

- There are a lot more things you can do with string formatting
- Presenting just a subset of the most commonly used functionality

Broader Issue



August	Andrew	Cat	Danielle	Alice
Charlotte	Bobby	Dan	Karel	Callie
Danny	James	Jake	Laurie	Ellis
Giovanni	Kassi	Melissa	Matt	Hayden
Jenna	Natalie	Nate	Mike	

Apps and Your Location Data

- What are the concerns about the apps having your location data?
- React to these statements in the article:
  - "You would have to be pretty oblivious if you are not aware that this is going on."
  - "In the most recent version of Android, apps that are not in use can collect locations 'a few times an hour,' instead of continuously."
  - Apple shelved this plan: "Last year, [Apple] said an upcoming version of iOS would show a blue bar onscreen whenever an app not in use was gaining access to location data."
- Did you know about apps tracking your location?
- Does it matter that apps sell your location data?
  - What are the tradeoffs?

Looking Ahead

- Chelsea Barabas Talk
  - Today, 5 p.m. Northen
- Lab 7 prep
  - Think about how to implement the Caesar Cipher
- Lab 7