11.2. Templating

The [string](https://docs.python.org/3/library/string.html#module-string) module includes a versatile [Template](https://docs.python.org/3/library/string.html#string.Template) class with a simplified syntax suitable for editing by end-users. This allows users to customize their applications without having to alter the application.

The format uses placeholder names formed by $ with valid Python identifiers (alphanumeric characters and underscores). Surrounding the placeholder with braces allows it to be followed by more alphanumeric letters with no intervening spaces. Writing $$ creates a single escaped $:

>>>

**>>> from** **string** **import** Template

**>>>** t = Template('$**{village}**folk send $$10 to $cause.')

**>>>** t.substitute(village='Nottingham', cause='the ditch fund')

'Nottinghamfolk send $10 to the ditch fund.'

The [substitute()](https://docs.python.org/3/library/string.html#string.Template.substitute) method raises a [KeyError](https://docs.python.org/3/library/exceptions.html#KeyError) when a placeholder is not supplied in a dictionary or a keyword argument. For mail-merge style applications, user supplied data may be incomplete and the [safe\_substitute()](https://docs.python.org/3/library/string.html#string.Template.safe_substitute) method may be more appropriate — it will leave placeholders unchanged if data is missing:

>>>

**>>>** t = Template('Return the $item to $owner.')

**>>>** d = dict(item='unladen swallow')

**>>>** t.substitute(d)

Traceback (most recent call last):

*...*

KeyError: 'owner'

**>>>** t.safe\_substitute(d)

'Return the unladen swallow to $owner.'

Template subclasses can specify a custom delimiter. For example, a batch renaming utility for a photo browser may elect to use percent signs for placeholders such as the current date, image sequence number, or file format:

>>>

**>>> import** **time**, **os.path**

**>>>** photofiles = ['img\_1074.jpg', 'img\_1076.jpg', 'img\_1077.jpg']

**>>> class** **BatchRename**(Template):

**...**  delimiter = '%'

**...**

**>>>** fmt = input('Enter rename style (**%d**-date %n-seqnum **%f**-format): ')

Enter rename style (%d-date %n-seqnum %f-format): Ashley\_%n%f

**>>>** t = BatchRename(fmt)

**>>>** date = time.strftime('**%d**%b%y')

**>>> for** i, filename **in** enumerate(photofiles):

**...**  base, ext = os.path.splitext(filename)

**...**  newname = t.substitute(d=date, n=i, f=ext)

**...**  print('**{0}** --> **{1}**'.format(filename, newname))

img\_1074.jpg --> Ashley\_0.jpg

img\_1076.jpg --> Ashley\_1.jpg

img\_1077.jpg --> Ashley\_2.jpg

Another application for templating is separating program logic from the details of multiple output formats. This makes it possible to substitute custom templates for XML files, plain text reports, and HTML web reports.