Appendix E. Cheat Sheets

I find myself looking up certain things a little too often. Here are some tables that I hope you'll find useful.

Operator Precedence

This table is a remix of the official documentation on precedence in Python 3, with the *highest* precedence operators at the top.

Operator	Description and examples
`[ v , …] , { v1 , …} , { k 1 : v1 , …} , (…)`	List/set/dict/generator creation or comprehension, parenthesized expression
`seq [ n ] , seq [ n : m ] , func ( args …) , obj . attr`	Index, slice, function call, attribute reference
`**`	Exponentiation
`+ n , – n , ~ n`	Positive, negative, bitwise `not`
`* , / , // , %`	Multiplication, float division, int division, remainder
`+ , -`	Addition, subtraction
`<< , >>`	Bitwise left, right shifts
`&`	Bitwise `and`
`|`	Bitwise `or`
`in , not in , is , is not , < , <= , > , >= , != , ==`	Membership and equality tests
`not x`	Boolean (logical) `not`
`and`	Boolean `and`
`or`	Boolean `or`
`if … else`	Conditional expression
`lambda …`	`lambda` expression

String Methods

Python offers both string *methods* (can be used with any `str` object) and a `string` module with some useful definitions. Let's use these test variables:

```
>>> s = "OH, my paws and whiskers!"
>>> t = "I'm late!"
```

In the following examples, the Python shell prints the result of the method call, but the original variables `s` and `t` are not changed.

Change Case

```
>>> s.capitalize()
'Oh, my paws and whiskers!'
>>> s.lower()
'oh, my paws and whiskers!'
>>> s.swapcase()
'oh, MY PAWS AND WHISKERS!'
>>> s.title()
'Oh, My Paws And Whiskers!'
>>> s.upper()
'OH, MY PAWS AND WHISKERS!'
```

Search

```
>>> s.count('w')
2
>>> s.find('w')
9
>>> s.index('w')
9
>>> s.rfind('w')
16
>>> s.rindex('w')
16
```

```
>>> s.startswith('OH')
True
```

Modify

```
>>> ''.join(s)
'OH, my paws and whiskers!'
>>> ' '.join(s)
'O H ,   m y   p a w s   a n d   w h i s k e r s !'
>>> ' '.join((s, t))
"OH, my paws and whiskers! I'm late!"
>>> s.lstrip('HO')
', my paws and whiskers!'
>>> s.replace('H', 'MG')
'OMG, my paws and whiskers!'
>>> s.rsplit()
['OH,', 'my', 'paws', 'and', 'whiskers!']
>>> s.rsplit(' ', 1)
['OH, my paws and', 'whiskers!']
>>> s.split(' ', 1)
['OH,', 'my paws and whiskers!']
>>> s.split(' ')
['OH,', 'my', 'paws', 'and', 'whiskers!']
>>> s.splitlines()
['OH, my paws and whiskers!']
>>> s.strip()
'OH, my paws and whiskers!'
>>> s.strip('s!')
'OH, my paws and whisker'
```

Format

```
>>> s.center(30)
'  OH, my paws and whiskers!   '
>>> s.expandtabs()
'OH, my paws and whiskers!'
>>> s.ljust(30)
'OH, my paws and whiskers!     '
>>> s.rjust(30)
'     OH, my paws and whiskers!'
```

String Type

```
>>> s.isalnum()
False
>>> s.isalpha()
False
>>> s.isprintable()
True
>>> s.istitle()
False
>>> s.isupper()
False
>>> s.isdecimal()
False
>>> s.isnumeric()
False
```

String Module Attributes

These are class attributes that are used as constant definitions.

Attribute	Example
`ascii_letters`	`'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'`
`ascii_lowercase`	`'abcdefghijklmnopqrstuvwxyz'`
`ascii_uppercase`	`'ABCDEFGHIJKLMNOPQRSTUVWXYZ'`
`digits`	`'0123456789'`
`hexdigits`	`'0123456789abcdefABCDEF'`
`octdigits`	`'01234567'`
`punctuation`	`'!"#$%&\'()*+,-./:;<=>?@[\\]^_\{|}~'\``
`printable`	digits + ascii_letters + punctuation + whitespace
`whitespace`	`' \t\n\r\x0b\x0c'`

Coda

Chester wants to express his appreciation for your diligence. If you need him, he's taking a nap…

Figure E-1. Chester

...but Lucy is available to answer any questions.

Figure E-2. Lucy

He’s moved about a

foot
to
the
right
since
Figure 3-1.