Type Conversion to integer

* int ( ) turn numeric or bool to int (drops fractional part)

*returns*

int (True) 1

int (98.6) 98

int (‘99’) 99 because text string only contains digits (otherwise throws exception)

int (‘+12’) 12

int (’98.6’) throws exception

Type conversion to float

*returns*

* 4 + 2.0 6.0

Type conversion with bool

*returns*

* True + 2.0 3.0

Integer overflow

* Integer overflow doesn’t occur in Python 3 because an int can now be any size
  + In Python 2 int was limited to 32 bits

Type conversion to float

* float ( ) turn numeric or bool to float

*returns*

float (True) 1.0

float (98) 98.0

float (’98.6’) 98.6

- - -

Strings

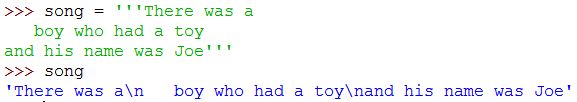
* Python 3 supports Unicode
  + So it can contain any written language in the world + many symbols
* A string in Python is a *sequence*
* Strings in Python are *immutable*

Creating strings

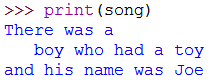
* Enclose characters in single or double quotes to create strings
* ‘Snap’ and “Crackle” are both string
  + Interactive interpreter returns string with single quotes in both cases
* But when we enclose single quotes within double, or vice-versa, the output is the same as our input

Multiline strings

* Created with triple quotes



* If we use print( ), it only returns the content:

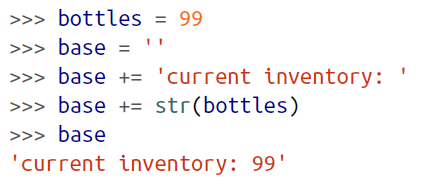




* + Note how print( ) also inserts spaces between the elements!

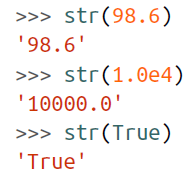
Initialise blank string

* blankStr = ‘ ‘
* Use:



Convert to string

* str ( )



* When print( ) is called, Python uses str( ) internally to operate on the non-string objects
  + str( ) also used for *string interpolation*

Escape sequences

* \n new line

\t tab

\’ to use ‘ inside print( )

\” to use “ inside print( )

\\ to use \ inside print( )

String concatenation

* Use + between strings

Repeat string

* Use \* x after string

