01-5 Python Formatted Printing

CSI 500

Spring 2018

Note: course material adopted loosely from :
Downey, Allen B. *Python for software design: how to think like a computer scientist*. Cambridge University Press, 2009.
http://greenteapress.com/wp/think-python/

Old print formatting in Python (2.x only…)

- Python 2.x supports the C formatted printing features

Formatting Expression	Usage
%d %4d	Integer Integer, 4 columns wide if possible
%s %20s	String (also default string representation for an object) Example:, right justified string in 20 columns
%c	Single character
%u	Unsigned integer
%o	Octal integer
%x, %X	Hexadecimal integer, lower case, upper case
%e, %E, %f, %F, %G %8.4f	Floating point formats Example: 8 digits, 4 after the decimal
%%	Literally a `%` character

```
>>> x = 123.45678
>>> print("%4.2f\n")%(x)
123.46

>>> print "hello world"
```

New print formatting in Python 3.x

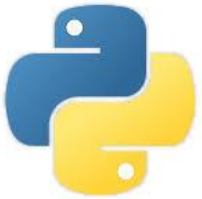
- print() expects a sequence of values
- use str() features
- use embedded codes within braces { }

```
>>> x = 123.456789
>>> print( '{0:16.8f}'.format(x))
   123.45678900
>>>

>>>  y = 17
>>> print( '{0:4.2f} {1:04d}'.format(x, y) )
123.45 0017

>>> z = "spam"
>>> print( '{}'.format(x) )
spam
>>> print( 'I like {0}, {1} and {other}'. \
   format('spam', 'eggs', other='bacon') )
I like spam, eggs, and bacon
>>> print( z.rjust(20) )
                spam
```

Python 3.x print()

print('spec'.format(list of items))

- spec
  - quoted string
  - may contain free text
  - contains bracketed format codes
    - {n:dT}
    - n is the item number or name in your list
    - d is the size code
    - T is the type code
- list of items
  - list of things you want to print
  - comma separated



```
# example with integers
#
#
>>> print( '{0:4d} {1:8d}'.format(10, 20))
  10       20

# example with floats
#
>>> print( '{0:4.2f} {1:8.4f}'.format(10.12345, 20.98765))
10.12  20.9876

# example with strings
#
>>> print( '{0:16s} {1:8s}'.format('monty', 'python'))
monty            python

# example with named strings
#
>>> print( '{last:16s} {first:8s}'.format(first='monty',
last='python'))
python           monty
```

Work around : printf()

- You can write your own work-around print function "printf()"

```
printf( '<spec>', item1, item2, …, itemN )
```

- spec
  - quoted string
  - may contain free text
  - contains C-style format codes
  - see https://alvinalexander.com/programming/printf-format-cheat-sheet for a nice summary
- sequence of items
  - list of things you want to print

```
>>> import sys
>>> def printf( format, *args ):
        sys.stdout.write( format % args )

# example with integers
#
>>> printf('%4d %8d', 10, 20)
  10       20

# example with floats
#
>>> printf('%4.2f %8.4f', 10.12345, 20.98765)
10.12  20.9876

# example with strings
#
>>> printf('%8s %16s', "monty", "python")
   monty          python
```

Selected C-Style formatting codes

Code	Type
%c	single character
%d	decimal integer in base 10
%e	exponential number
%f	floating point real number
%o	integer in base 8 (octal)
%s	string of characters
%u	unsigned integer in base 10
%x	integer in base 16 (hex)
%%	print a '%' character
\%	print a '%' character

Example	result
printf("%4d", 15)	15
printf("%-4d", 15)	15
printf("%8.2f", 123.45)	123.45
printf("%8.4e", 12345678)	12.3456E7
printf("%04d', 15)	0015
printf("%10s", "hello")	' hello'
printf("%-10s", "hello")	'hello '

https://alvinalexander.com/programming/printf-format-cheat-sheet

Summary

- Python provides various ways to print formatted output
  - Version 2.x and prior are *NOT* compatible with version 3.x and above
- The Python 3.x print function requires
  - a format specifier using positional or named parameters
  - a list of items to print
- You can also make a quick work-around if you prefer C-style formatting
  - define a "printf() function
  - note: your code won't be as portable