Exposure CS 2020 for CS1

Chapter 3 Section 3-4 Slides

Introduction to Python Coding: Output with print & Comments

PowerPoint Presentation
created by:
Mr. John L. M. Schram
and Mr. Leon Schram
Authors of Exposure
Computer Science

Section 3.3

Text Output With print

```
1 # TextOutput01.py
2 # This program demonstrates
3 # text output with <print>.
4
5
6 print("Hello World!")
7
8
```

```
 ----jGRASP exec: python TextOutput01.py
Hello World!


 ----jGRASP: operation complete.
```

Python Keywords & Program Statements

A Python *keyword* is a word that has a special meaning in a program or performs a special function.

A *program statement* usually contains one or more *keywords*.

Keywords in Python are case-sensitive.
For example:
print is a Python keyword, while **PRINT** is not.

```
1  # TextOutput02.py
2  # This program demonstrates how to
3  # display 4 lines of text using 4
4  # separate <print> commands.
5
6
7  print("Line1")
8  print("Line2")
9  print("Line3")
10 print("Line4")
11
```

```
  ----jGRASP exec: python TextOutput02.py
Line1
Line2
Line3
Line4

  ----jGRASP: operation complete.
```

```
 1 # TextOutput03.py
 2 # By default, the <print> command "ends" by going to
 3 # the next line -- as if it pressed the <enter> key.
 4 # However, you can change what happens at the <end>
 5 # of a <print> command.  This allows multiple outputs
 6 # to be on the same line.  In the particular example,
 7 # the <end> values are spaces, so the first 3 outputs
 8 # are all on the same line, separated by spaces.
 9 # Note that "Line4" is displayed on its own line.
10 # This is because "Line3" was displayed with a normal
11 # <print> command without an <end>.
12
13
14 print("Line1",end = " ")
15 print("Line2",end = " ")
16 print("Line3")
17 print("Line4")
18
```

```
----jGRASP exec: python TextOutput03.py
Line1 Line2 Line3
Line4


----jGRASP: operation complete.
```

```
1  # TextOutput04.py
2  # This program is very similar to the previous
3  # program.  The only difference is that instead
4  # of ending with a space, the first 2 <print>
5  # commands <end> with an "empty string".  While
6  # the first 3 outputs are still on the same line,
7  # this time there is nothing in-between them.
8
9
10 print("Line1",end = "")
11 print("Line2",end = "")
12 print("Line3")
13 print("Line4")
14
```

```
  ----jGRASP exec: python TextOutput04.py
Line1Line2Line3
Line4


  ----jGRASP: operation complete.
```

```
1  # TextOutput05.py
2  # This program demonstrates that you can put any
3  # character, or even several characters, inside
4  # the quotes of an <end> keyword, which can lead
5  # to some weird looking output.
6
7
8  print("Line1",end = "$")
9  print("Line2",end = "???")
10 print("Line3")
11 print("Line4")
```

```
  ----jGRASP exec: python TextOutput05.py
Line1$Line2???Line3
Line4


  ----jGRASP: operation complete.
```

```python
# TextOutput06.py
# This program shows how to skip one or more
# lines when displaying text.  Using <print>
# with empty parentheses will generate a
# crlf (carriage-return/line-feed).


print("Line1")
print()
print("Line3")
print()
print()
print()
print("Line7")

```

```
----jGRASP exec: python
Line1

Line3




Line7

----jGRASP: operation co
```

```
 1 # TextOutput07.py
 2 # This program has the exact same output
 3 # as TextOutput06.py. It shows another
 4 # way to skip one or more lines by using
 5 # the "Escape Sequence" <\n> which means
 6 # "New Line".
 7
 8
 9 print("Line1\n")
10 print("Line3\n\n\n")
11 print("Line7")
12
```

```
----jGRASP
Line1

Line3



Line7

----jGRASP:
```

print & end

The **print** command generates an output display of the characters contained between double quotes.

If the **end** keyword is not used, **print** will also generate a *carriage-return/line-feed* (crlf).

If the **end** keyword is used, **print** will not generate a crlf and instead display the specified text in the second set of quotes.

Examples:

print("Hello") **print("World")**	**print("Hello", end = " ")** **print("World")**
will display:	will display:
Hello World	Hello World

Creating Blank Lines

Creating a blank line of output can be done 2 different ways. One is to use **print()** with absolutely nothing in the parentheses. The other is to add the **\n** *Escape Sequence* to an existing **print** statement:

Examples:

print("Hello") **print()** **print("World")**	**print("Hello\n")** **print("World")**
will display:	will also display:

`Hello`
 	
`World`
	`Hello`
	
	`World`

Section 3.4

Program Comments

```
 1 # Comments01.py
 2 # This program displays several number words.
 3 # The focus now is on program comments.
 4 # Program comments aid in "program documentation"
 5 # and makes your program more readable.  Every line
 6 # that begins with a "hashtag" is considered a
 7 # "comment" by the Python interpreter.
 8 # The Python interpreter ignores all comments.
 9 # They are not executed.
10 # If a line begins with a hashtag, it is simply
11 # ignored by Python. That is precisely what is
12 # happening with the first 19 lines of this program.
13 # Note below that a comment can also be placed in
14 # the middle of the program.  They can even be placed
15 # right after a program statement on the same line.
16 # You will also see that the word "Thirteen" is not
17 # displayed because it has been "commented-out"
18 # possibly by someone suffering from "Triskaidekaphobia"
19 # (the fear of the number 13).
```

```
20
21
22 print()
23 print("One")
24 print("Two")
25 print("Three")
26 print("Four")
27 print("Five")
28 print("Six")        # half a dozen
29 print("Seven")
30 print("Eight")
31 print("Nine")
32 print("Ten")
33 print("Eleven")
34 print("Twelve")     # one dozen
35 #print("Thirteen")  # one baker's dozen
36
37
38
```

```
20
21
22 print()
23 print("One")
24 print("Two")
25 print("Three")
26 print("Four")
27 print("Five")
28 print("Six")        # half a dozen
29 print("Seven")
30 print("Eight")
31 print("Nine")
32 print("Ten")
33 print("Eleven")
34 print("Twelve")     # one dozen
35 #print("Thirteen")  # one baker's
36
37
38
```

```
----jGRASP

One
Two
Three
Four
Five
Six
Seven
Eight
Nine
Ten
Eleven
Twelve

----jGRASP:
```

```
1  # Comments02.py
2  # This program demonstrates that a section of
3  # code can essentially be "commented-out" by
4  # using triple-double-quotes.
5
6
7  print()
8  print("One")
9  print("Two")
10 """
11 print("Three")
12 print("Four")
13 print("Five")
14 print("Six")          # half a dozen
15 print("Seven")
16 print("Eight")
17 print("Nine")
18 print("Ten")
19 print("Eleven")
20 """
21 print("Twelve")      # one dozen
22 print("Thirteen")    # one baker's dozen
```

```
----jGRASP

One
Two
Twelve
Thirteen

----jGRASP:
```

String Literal Definition

A *string literal* is any text in-between a set of quotes.

The “text” can be a name, letter, group of words, or basically anything that you can type inside a set of quotes.

Examples:
"computer"
"John Smith"
"Hello all you happy people."
"Q"
"?"
"811 Fleming Trail"
"Richardson, Texas"
"75081"

Most **print** statements contain a *string literal* in their parentheses.

Python Multi-Line Comment Disclaimer

Technically, Python does not have *Multi-Line Comments*.

The “Triple-Double-Quote” is actually used to create a multi-line *string literal*.

This is why they are **green** and not **red** like the *Single-Line Comments*.

However, multi-line string literals which are not part of a **print** statement are simply ignored by the Python interpreter.

This essentially makes them multi-line comments.

```
1  ###################################
2  #                                 #
3  #          Comments03.py          #
4  #        Numbers from 1-13        #
5  #         By: John Schram         #
6  #             11/8/17             #
7  #                                 #
8  #  This program is similar to     #
9  #  the previous two and shows     #
10 #  that a comment can be used     #
11 #  to create a heading.           #
12 #                                 #
13 ###################################
14
15
16 print()
17 print("One")
18 print("Two")
19 print("Three")
20 print("Four")
21 print("Five")
22 print("Six")        # half a dozen
23 print("Seven")
24 print("Eight")
25 print("Nine")
26 print("Ten")
27 print("Eleven")
28 print("Twelve")     # one dozen
29 print("Thirteen")   # one baker's dozen
```

```
----jGRASP

One
Two
Three
Four
Five
Six
Seven
Eight
Nine
Ten
Eleven
Twelve
Thirteen

----jGRASP:
```

2 Different Types of Comments

```
# This is a single-line comment.

print("Good morning!")   # So is this.
```

```
"""
This is a
multi-line
comment.
"""
```