#1.py

```python
print("Hello World!")
```

#2.py

```python
# Read in and print whatever you type in

sometext = input("Enter any text you want ")

print("This is what you typed:")

print(sometext)
```

#3.py

```python
# The \n (just as in the C-language) between quotes
# causes a new line wherever it occurs

sometext = input("Enter any text you want \
")

print("This is what you typed:")

print(sometext)
```

#4.py

```python
# assign a character string to a variable, and then use that variable

waitforinput = "Enter any text you want 
"

sometext = input(waitforinput)
```
print ("This is what you typed:")

print (sometext)

# a function with no parameters (i.e., nothing between the parentheses)
def print_some_text():
    print("This is an example of a function ")
    print("that prints these lines ")
    print("whenever you invoke it by name.")
    print("Note that each print is on a new line ")

# A function can accept parameters as input so that you can use
# them flexibly

def say_hello(name, shoe_colour):
    print("Hello there ", name)
    print("I do like your stunning ", shoe_colour," shoes!")

# A function that computes y = f(x^2) (that is x squared) for integer x in
# the
# interval [-20,20]
def main():
    print ("A simple example of a for-loop")
```python
for x in range(-20, 21):  # x is an integer variable now used as a for-loop index
    y = x * x
    print(x, y)
```

# Example from textbook (chaotic function), but with two inputs (x and xprime) simultaneously

def main():
    print("We will demonstrate a chaotic function")
    x = eval(input("Enter any number between 0 and 1: "))
    xprime = x + 0.01
    print(""", x, "", xprime)
    for i in range(10):
        x = 3.9 * x * (1-x)  # this is the
        xprime = 3.9 * xprime * (1 - xprime)  # chaotic function
        print(i, "", x, "", xprime)
```

# You'll notice that, even though we left spaces (blank characters) in the # print statement, the numbers will not be printed in even vertical columns. # Why? Because some output numbers have fewer digits after the decimal point # than others. The extra zeroes are not printed.

# We will learn how to format strings later, to get prettier output.
```