```python
# String review from the reading

name = "Neal"
# Draw name's layout in memory.





# Now tell me what these lines print:
print(len(name))


print(name[0])


# What do we call the 0?

# How do we access the LAST character in the string?




# Python is unusual: it allows negative indexes, which count from
# the right end instead of the left.
print(name[-1])
# Same as:
print(name[len(name) - 1])


# Although we can access at an index, we can't modify at an index.
# name[0] = 'D' # not allowed



# Strings can be compared with == and !=.
if name == "Neal":
    print("Hi, Neal!")
else:
    print("I don't know who you are!")


# Individual characters can be compared, too.
other_name = "Neil"
if other_name[2] == "i":
    print("You spell your name wrong,", other_name, "!")


# We can use a loop to iterate over each individual
# character of a string.
```

```python
for i in range(len(name)): # what does this do?
    print(name[i])


# CHALLENGE: write a loop that prints the string backwards.
print("BACKWARDS")





# CHALLENGE: write a loop that prints every other letter of the string.
print()
print("EVERY OTHER")







# CHALLENGE: write a loop that prints out all the consonants in
# the string.
print("CONSONANTS ONLY")
```

```python
# String slices

# A "slice" is a portion of an existing string, designated with a start index
# and an end index. The slice creates a copy of the string starting at the given
# index, and copying all characters in the string up to but not including the
# end index.

st = "CECS 174 is so much fun"
department = st[0:4]
# First number is the start index; second is the end index (remember, not inclusive).
# Another way:
department = st[:4]
# With no start index, we start at 0.
print(department)

course_num = st[5:8]
print(course_num)

print(st)

feeling = st[12:] # with no end index, we go to the last character
print(feeling)

# Remember: a slice does NOT MODIFY the original string. It just copies some indexes
# into a new string.

# Simple functions on strings

# count_x: return the number of times the letter 'x' can be found in a string.
def count_x(st):





print(count_x("abxcdefxxg"))


# double_letter: returns True if the given string has the same letter
#   twice in a row.
def double_letter(st):






print(double_letter("terrell"))


# slice: returns a slice between the given indices, without using
#   the built-in Python slicing operator.
def slice_string(st, start, end):




# This "overloaded function" takes a stride amount as well.
def slice_string(st, start, end, stride):







# reverse_string: returns the reverse of the given string, without using
#   the built-in Python function.
def reverse_string(st):






```

```python
print(reverse_string("terrell"))


# is_palindrome: given a string that only contains lowercase letters,
#   determine if the string is a palindrome.

# hannah
# otto
# racecar

def is_palindrome(st):







print(is_palindrome("hannah"))
print(is_palindrome("A dog, a panic in a pagoda!"))
print(is_palindrome("lonelytylenol"))
```

```python
# "Parse" a phone number into three integer variables: the
# area_code, the prefix, and the suffix.

# The phone number can be in one of three formats:
# (AAA) BBB-CCCC
# AAA BBB CCCC
# AAABBBCCCC

phone_number = input("Enter a phone number")

if phone_number[0] == '(':
    area_code = phone_number[1:4]
    prefix = phone_number[6:9]
    suffix = phone_number[10:]
elif phone_number[3] == ' ':
    sp = phone_number.index(" ")
    # sp == ?
    area_code = phone_number[:sp]
    sp2 = phone_number.index(" ", sp + 1)
    # sp2 == ?
    prefix = phone_number[sp+1:sp2]
    suffix = phone_number[sp2+1:]
else:
    parsed = int(phone_number)
    suffix = parsed % 10000
    prefix = (parsed // 10000) % 1000
    area_code = parsed // 10000000

print("Area code {0}, prefix {1}, suffix {2}".format(area_code, prefix, suffix))

```