Introduction to Electrical and Computer Engineering Design ECEG 201

Session 13 Lecture Notes

Announcements

1. Reading:
   (a) The Python Tutorial
      i. Section 3.1.2, *Strings*
      ii. Section 4.6, *Defining Functions*
      iii. Section 4.7, *Documentation Strings*
2. Homework 7 is due by *noon* on 2020-03-06

   We will use class time on Wednesday and Friday to work on the homework.

Python byte string methods

```
>>> my_bytes = b'spam and eggs'
>>> my_bytes
b'spam and eggs'
>>> other_bytes = b' and spam'
>>> my_regular_string = my_bytes.decode()
>>> my_menu = my_bytes + other_bytes
>>> my_menu
b'spam and eggs and spam'
>>> len(my_menu)
22
>>> mixed = b'chars ' + chr(1).encode() + b' and data'
>>> mixed
b'chars \x01 and data'
>>> len(mixed)
16
```

Remember that a `bytes` object is immutable…it can not be modified in place (except for a few simple cases, like the `strip` method).

The `chr()` function returns a 1-character string. The character is the unicode character that corresponds to the integer parameter passed to `chr()`. Note that the **value** of the integer and the **value** of the character are actually exactly the same…this function actually changes the type of the object. When you print the value of that character you will see the printable character that has the given integer value. The `ord()` function does the reverse…it basically changes a single-character string into the corresponding integer.

```
>>> chr(66)
'B'
>>> ord(chr(66))
66
>>> ord('B')
66
```

If there is no printable character corresponding to that integer value then an "escaped" value is printed. For example, the ASCII "carriage return" has a decimal value of 13. This is used so often that Python has a special way of representing it: a backslash followed by the lower-case letter r. Even though it looks like there are two characters (i.e. two bytes) used to store this value, there is only one byte in that actual string.

```
>>> chr(13)
'\r'
```

If the integer parameter value that you pass to `chr()` is not a printable character and doesn't have a special representation it will be printed as an escaped hexadecimal value: backslash followed by a lower-case x, followed by the two hexadecimal digits for the value of the byte. Again, it looks like there are four bytes used to represent this single integer, but in fact the string object holds just one character. And just like any other string you can use the `encode()` method to convert it to a bytes object containing a single byte. The default for the `encode()` method is to encode the string as 7-bit ASCII characters…if you want to encode as 8-bit bytes you need to use the latin-1 encoding.

```
>>> chr(00)
'\x00'
>>> len(chr(00))
1
>>> chr(00).encode()
b'\x00'
>>> len(chr(00).encode())
1
>>> chr(127).encode()
b'\x7f'
>>> chr(128).encode()
b'\xc2\x80'
>>> chr(128).encode('latin-1')
b'\x80'
```

If you want to see all of the bytes in a byte string printed as their hexadecimal values, you can use this incantation:

```
>>> msg = b'parror\r\n'
>>> print(" ".join(hex(char) for char in msg))
0x70 0x61 0x72 0x72 0x6f 0x72 0xd 0xa
```

Python functions

```
def my_function_name(my_first_parameter, my_second_parameter):
    """This is my docstring for this function."""
    my_local_variable = something
    something_else = my_first_parameter / my_second_parameter
```

- Python function definitions begin with the keyword `def`, followed by the function name, followed by a list of parameters (inside parentheses), followed by a colon.
- The first line of a function is its **docstring**…a complete sentence, ending with a period, surrounded by three double-quotes.
- All variable assignments inside a function are to **local** variables.

Note that PEP 8 specifies that names of functions and variables should be `lower_case_with_underscores`.

Function return values

```
def my_func(my_param):
    """This function does magic."""
    my_return_value = my_param * 2.0
    return my_return_value
```

- If there is no `return` statement in the function, the default object returned is `None`.
- The `return` statement without an expression returns `None`.
- The return value can have any data type.
- A function may have multiple `return` statements, but the first one executed will terminate the function.