MicroPython Introduction (cont.)

MCHE 201 – Spring 2019

Dr. Joshua Vaughan

Rougeou 225

joshua.vaughan@louisiana.edu

@Doc_Vaughan

Comments Review

```
""" This is a block comment. It will
continue across multiple lines, until it
is closed with the proper characters """

# This is a single-line comment

x = 4 # single-line comments can come
after code too
```

Print Statement Review

```
# Print a string
print("Hello.")

# Print an integer
x = 10
print(x) # Prints with no formatting
print("x = {:d}.".format(x))
```

```
# Print an integer and always include +/- sign
print("Integer = {:+d}.".format(42))

# Print a float
print("Pi is {:f}.".format(3.141592))

# Print a float with 4 decimal places
print("Pi is {:.4f}.".format(3.141592))
```

If... then Review

```
# ----- if... else if... else syntax -----------------

if (condition1 == True):
    # Everything indented here is run if condition1 is true

# if condition1 isn't true, check condition2
elif (condition2 == True):
    # Everything indented here is run if condition1 is
    false, but condition2 is true

else: # if neither condition1 or condition2 are true, do
this
    # Everything indented here is run if both condition1 and
    condition2 are false
```

For Loops Review

```
# ----- for loop syntax --------------------------
for counter in sequence:
    # do something
    # Everything indented here is run during each
    # loop until the sequence is finished
```

Variable that's incremented

What to loop over... a few options for what

```
# ----- for loop example --------------------------
for counter in range(10):
    # do something
    # This would run 10 times
    # The values of counter would be 0, 1, 2, …, 9
```

While Loop Review

```
// ----- while loop syntax ------------------------
```

```
while (condition == True):
```

The condition is tested at the beginning of each iteration

```
    # If the condition is true, run the code here.

    # Once the code in the indented block is
    finished, check the condition and repeat.

    # If the condition is not true at the first check
    above, this will never be run.
```

Python Functions

Says "This is a function"

Function Name

Input Variable Names

```
def printRepeatPhrase(repeat, phrase):
  for counter in range(repeat):
    print(phrase)
```

Nothing is returned from this function

Must Space/tab Consistently – Remember that in Python, whitespace matters

To Use That Function

```
def printRepeatPhrase(repeat, phrase):
    for counter in range(repeat):
        print(phrase)

# Assign values
numRepeats = 3
textToPrint = "Hello"

# Call the function
printRepeatPhrase(numRepeats, textToPrint)
```

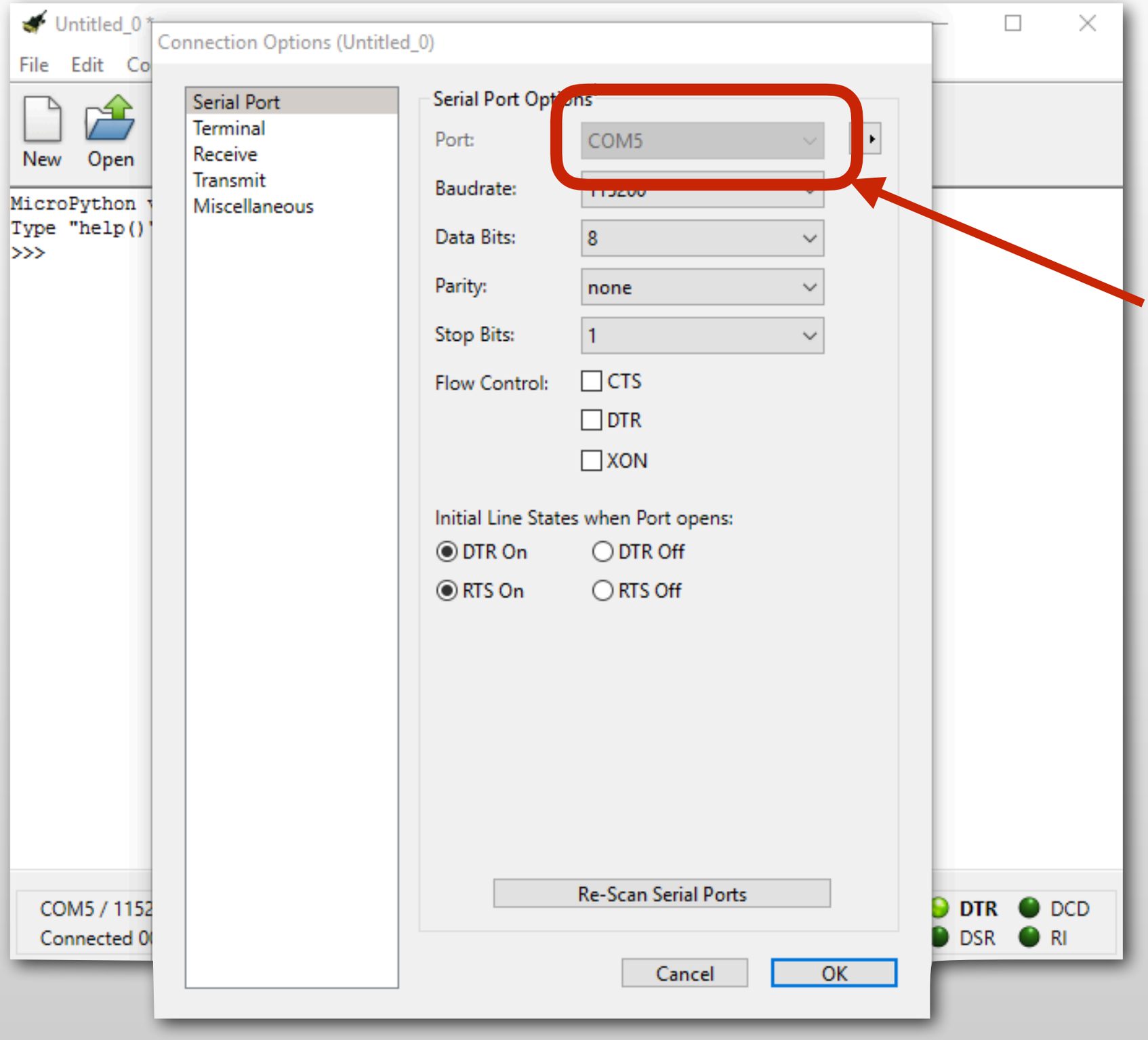
Here, `numRepeats` is equal to 3, meaning Hello would be printed to the REPL three times.

Connecting to the REPL

- Will need to use some serial terminal

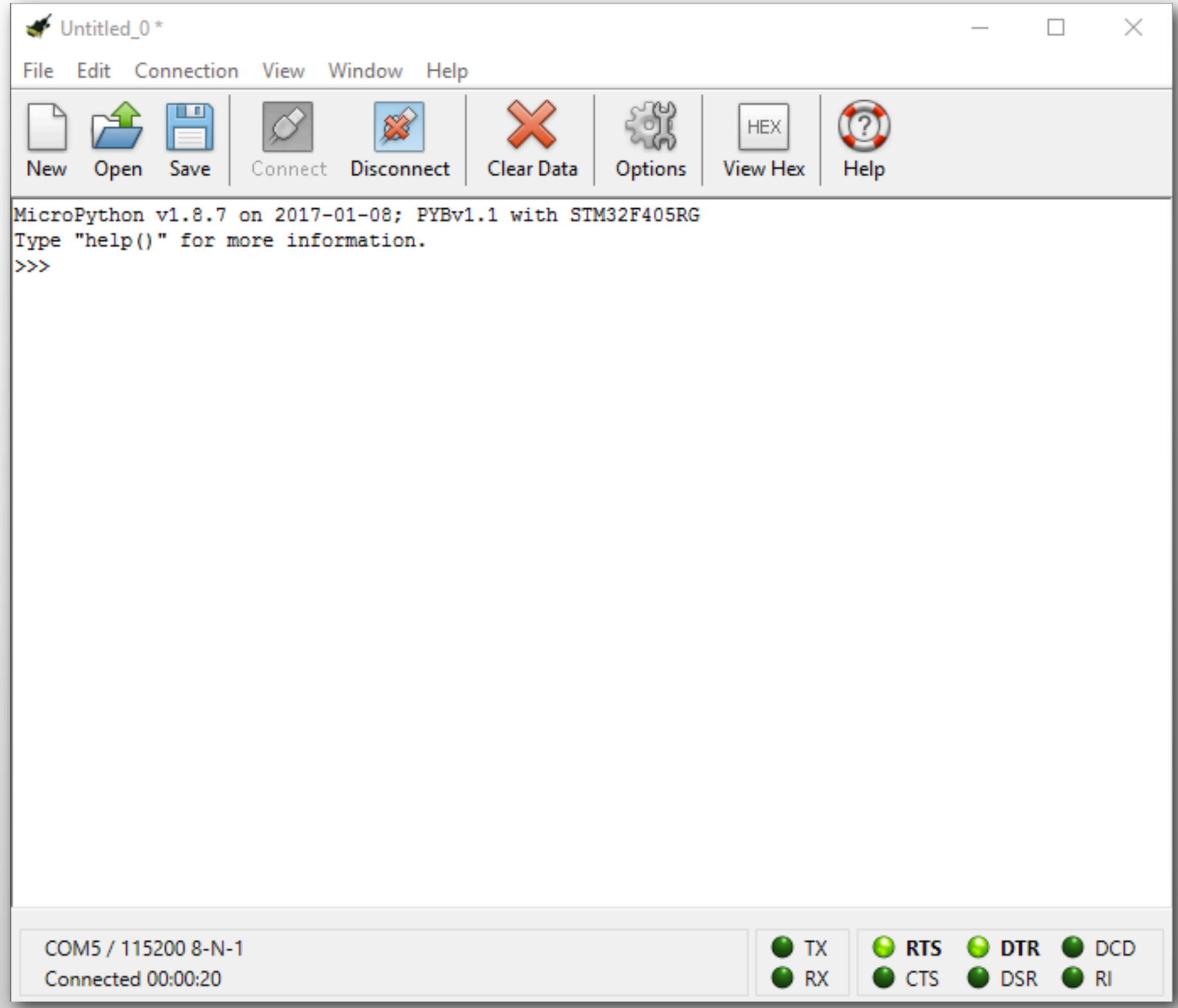
- Serial Settings
  - Baudrate = 115200bps
  - Date bits = 8 bits
  - Parity = None
  - Stop bits = 1
  - Flow control = none

CoolTerm Settings in Windows

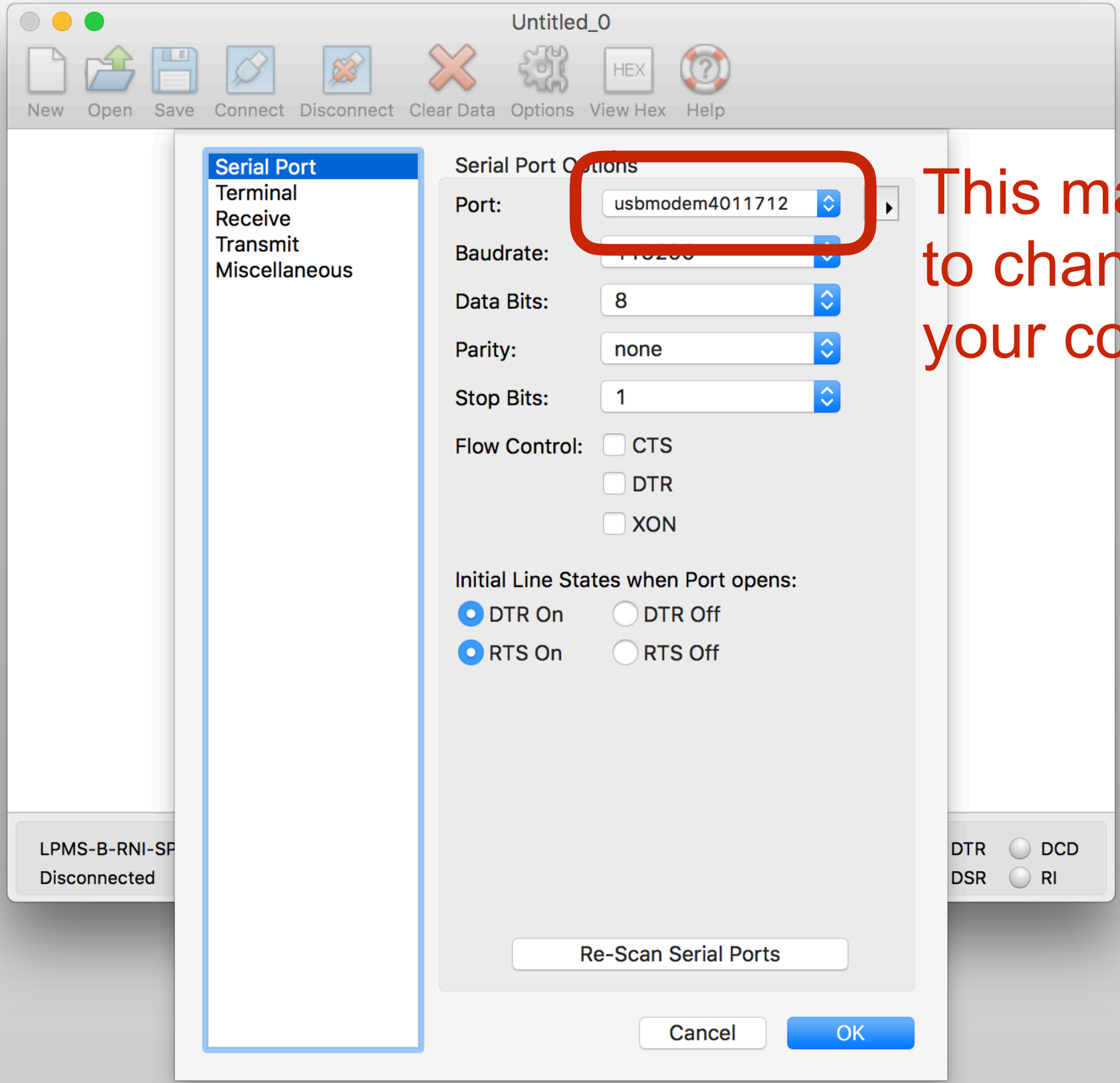


This may need to change for your computer

CoolTerm Terminal in Windows…

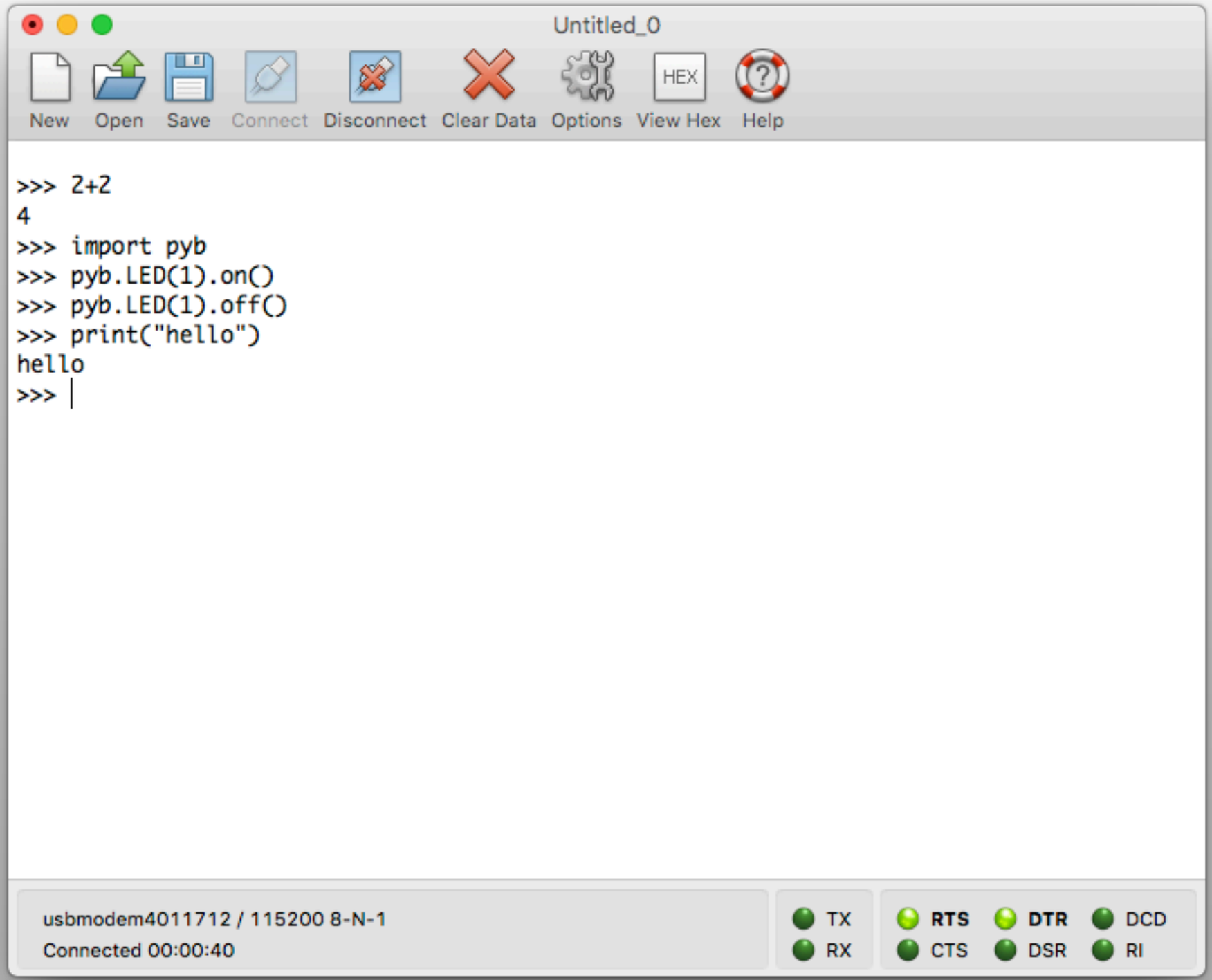


Cool-Term Settings on macOS



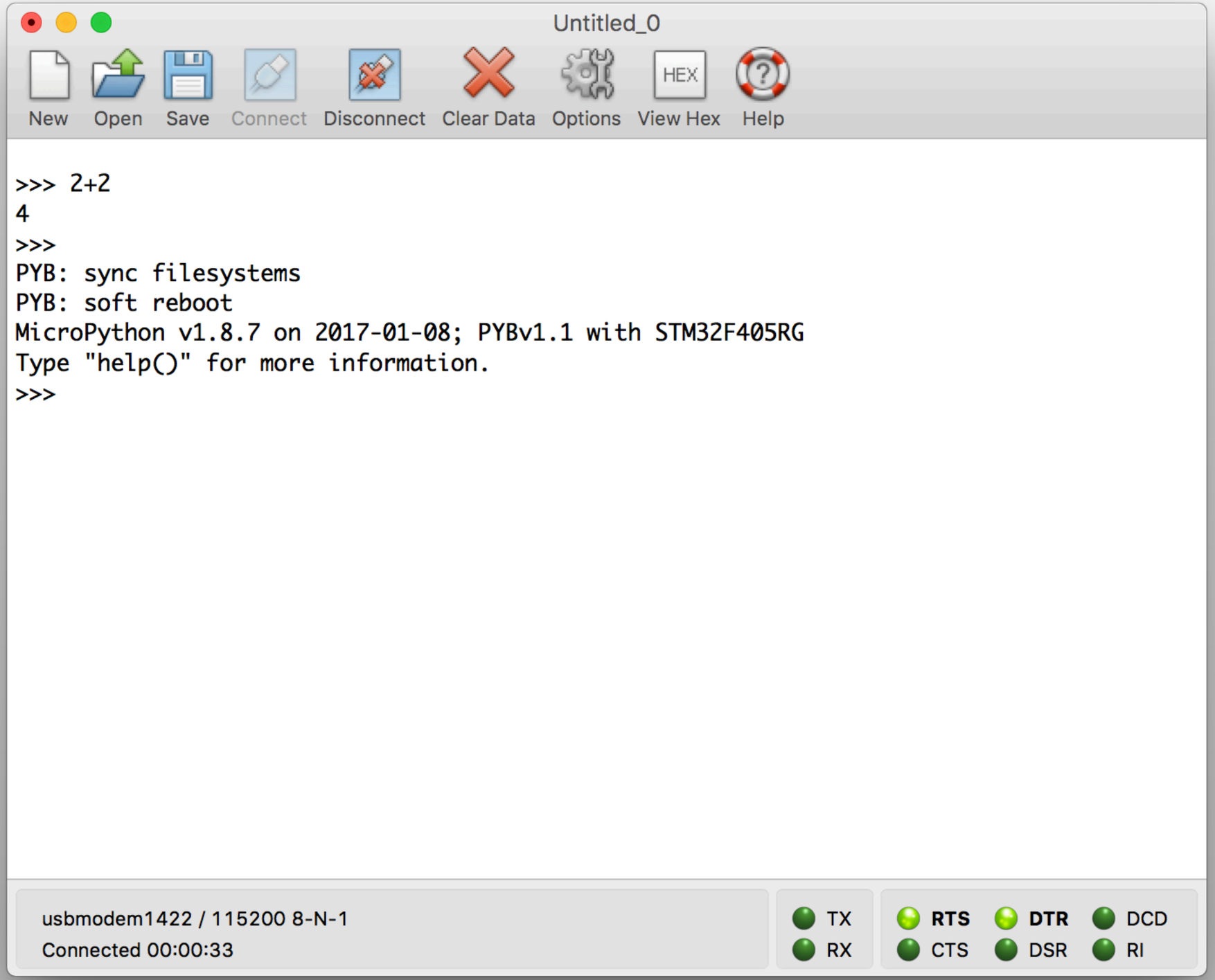
This may need to change for your computer

CoolTerm Connection on macOS



Special Commands in the REPL

- `Control-d` will perform a soft reboot

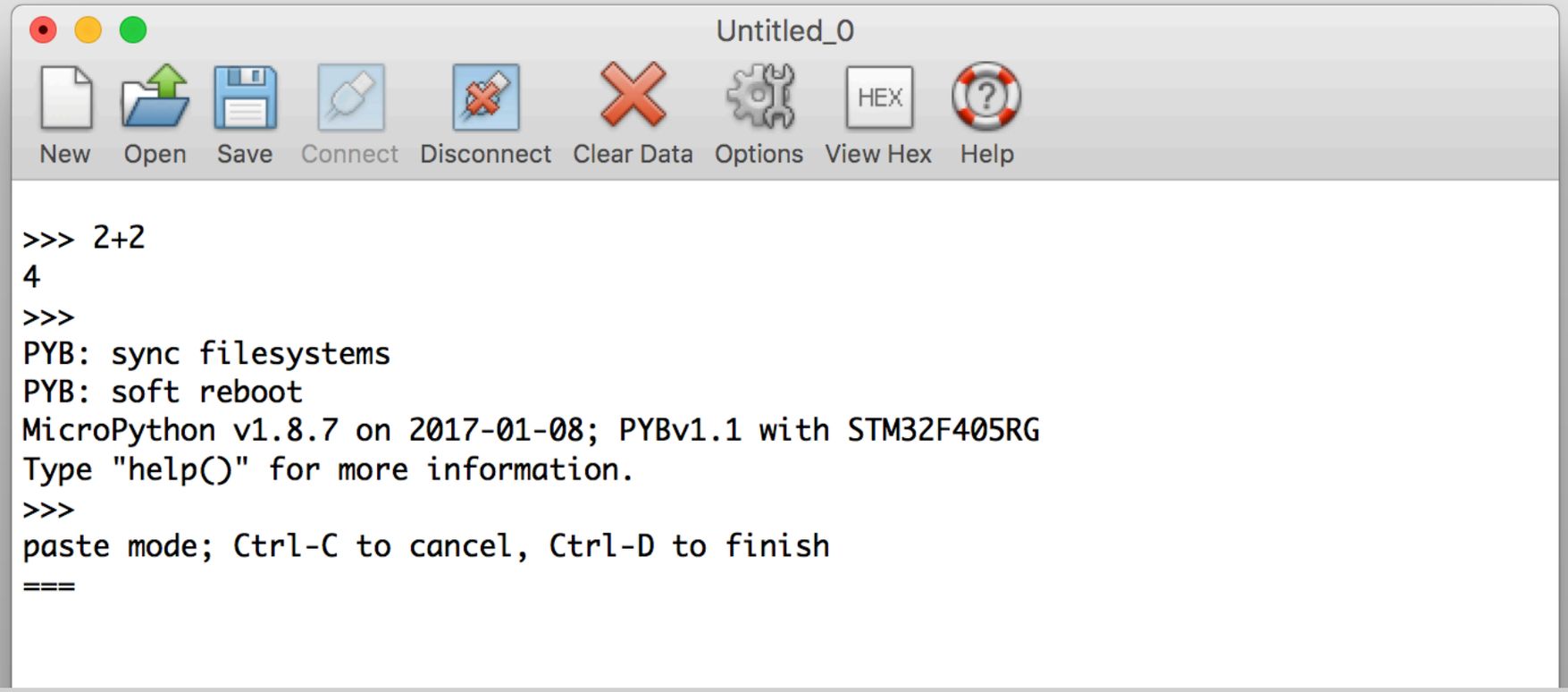


Special Commands in the REPL

- `Control-d` will perform a soft reboot
- `Control-c` will kill any running script

Special Commands in the REPL

- `Control-d` will perform a soft reboot
- `Control-c` will kill any running script
- `Control-e` will enter paste mode
  - Paste as usual
  - Use `Control-d` to exit paste mode



Recommended Workflow

- Connect the board to your computer and start the REPL in CoolTerm

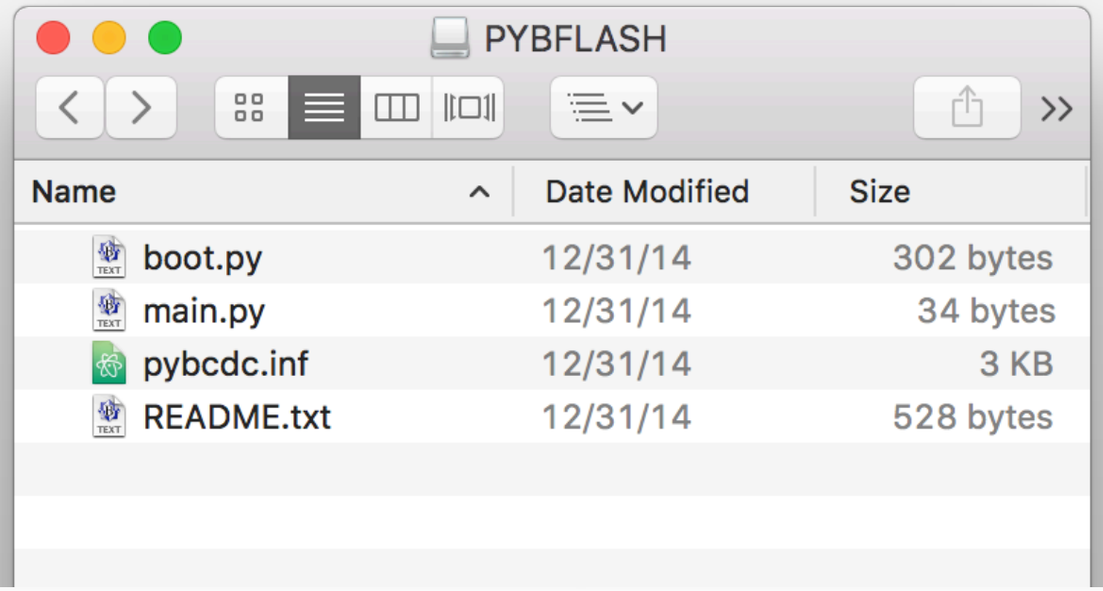
- Work on scripts (mostly `main.py` in MCHE201) in a local folder with Visual Studio Code

- Drag (or otherwise copy) edited versions to `PYBFLASH`

- `Control-d` in the REPL to perform a soft reboot and run edited `main.py`

MicroPython Files

- `boot.py`
  - Runs every time the pyboard boots
  - Use for setup and configuration



- `main.py`
  - Executed immediately after `boot.py`
  - Use for your “main” code
  - Can reference other files

`boot.py` and `main.py` are at the "root" of the `PYBFLASH` drive (*i.e.* They are not in a folder.)

Minimum Viable `boot.py`

Minimum "Normal" boot.py

```
# boot.py -- run on boot-up
# can run arbitrary Python, but best to
# keep it minimal

import machine
import pyb
```

These `import` statements make the code in those modules, libraries, or files available.

Using imports

Just prepend the variable or function you want to use with the “name” that you imported

```
# Import the pyboard functions
import pyb

# To use a function from pyb, put pyb.
# in front of the function name.
RED_LED = pyb.LED(1)
```

Using `imports`

Just prepend the variable or function you want to use with the “name” that you imported

```
# Import time module
import time


# sleep for 1 second
time.sleep(1)

# sleep for 500 milliseconds
time.sleep_ms(500)

# sleep for 10 microseconds
time.sleep_us(10)
```

Change (local-copy) `main.py`, to be...

```
# main.py -- put your code here!

# import the pyboard module
import pyb

# Turn on the 1st LED
pyb.LED(1).on()
```

To run...

- Copy `main.py` to the pyboard
- Open CoolTerm to REPL
- `Control-d` to perform a soft reboot and run edited `main.py`

Change `main.py`, to be...

```
# main.py -- put your code here!

import pyb    # import the pyboard module
import time   # import the time module

# Assign the 1st LED to variable RED_LED
RED_LED = pyb.LED(1)

# Now, we can control it using RED_LED
RED_LED.on()    # Turn the RED_LED on
time.sleep(5)   # Sleep 5 seconds
RED_LED.off()   # Turn the LED off
```

How could we do this 5 times?

```
import pyb  # import the pyboard module
import time # import the time module

# Assign the 1st LED to variable RED_LED
RED_LED = pyb.LED(1)

# This for loop will run 5 times
for counter in range(5):
  RED_LED.on()  # Turn the RED_LED on
  time.sleep(2) # Sleep 2 seconds while on
  RED_LED.off() # Turn the LED off
  time.sleep(2) # Sleep 2 seconds while off
```

An Alternate Solution

```
import pyb  # import the pyboard module
import time # import the time module

# Assign the 1st LED to variable RED_LED
RED_LED = pyb.LED(1)

# This for loop will run 10 times
for counter in range(10):
  RED_LED.toggle()  # Toggle the RED_LED on
  time.sleep(2)     # Sleep 2 seconds
```

Where can I find help?

- Full – `http://docs.micropython.org/en/latest/pyboard/`
- Quick Ref – `http://docs.micropython.org/en/latest/pyboard/pyboard/quickref.html`
- REPL specific – `http://docs.micropython.org/en/latest/pyboard/reference/repl.html`
- More links coming to class webpage
- If you don’t remember the syntax, look it up

help() in the REPL

Untitled_0

New Open Save Connect Disconnect Clear Data Options View Hex Help

```
>>> help()
Welcome to MicroPython!

For online help please visit http://micropython.org/help/.

Quick overview of commands for the board:
  pyb.info()    -- print some general information
  pyb.delay(n)  -- wait for n milliseconds
  pyb.millis()  -- get number of milliseconds since hard reset
  pyb.Switch()  -- create a switch object
                   Switch methods: (), callback(f)
  pyb.LED(n)    -- create an LED object for LED n (n=1,2,3,4)
                   LED methods: on(), off(), toggle(), intensity(<n>)
  pyb.Pin(pin)  -- get a pin, eg pyb.Pin('X1')
  pyb.Pin(pin, m, [p]) -- get a pin and configure it for IO mode m, pull mode p
                   Pin methods: init(..), value([v]), high(), low()
  pyb.ExtInt(pin, m, p, callback) -- create an external interrupt object
  pyb.ADC(pin)  -- make an analog object from a pin
                   ADC methods: read(), read_timed(buf, freq)
  pyb.DAC(port) -- make a DAC object
                   DAC methods: triangle(freq), write(n), write_timed(buf, freq)
  pyb.RTC()     -- make an RTC object; methods: datetime([val])
  pyb.rng()     -- get a 30-bit hardware random number
  pyb.Servo(n)  -- create Servo object for servo n (n=1,2,3,4)
```

usbmodem1422 / 115200 8-N-1
Connected 00:00:43

TX RX | **RTS** CTS **DTR** DSR DCD RI

Tab-completion in REPL

- Start your command, then hit tab to:
  - See options for completion
  - If only 1 option, fill in the complete function or variable name

Tab-completion in REPL

- Star
  - S
  - It ble
    n

Untitled_0

New Open Save Connect Disconnect Clear Data Options View Hex Help

```
>>> import pyb
>>> pyb.
__name__        bootloader      hard_reset      info
unique_id       freq            repl_info       wfi
disable_irq     enable_irq      stop            standby
main            repl_uart       usb_mode        hid_mouse
hid_keyboard    USB_VCP         USB_HID         have_cdc
hid             millis          elapsed_millis  micros
elapsed_micros  delay           udelay          sync
mount           Timer           rng             RTC
Pin             ExtInt          pwm             servo
Servo           Switch          Flash           SD
SDCard          LED             I2C             SPI
UART            CAN             ADC             ADCAll
DAC             Accel           LCD
>>> pyb.
```

usbmodem1422 / 115200 8-N-1
Connected 00:02:32

TX RX | RTS CTS DTR DSR DCD RI

Tab-completion in REPL

- Star
  - S
  - It ble
    n

Untitled_0

New Open Save Connect Disconnect Clear Data Options View Hex Help

```
>>> pyb.LED.
on              off             toggle          intensity
>>> pyb.LED.
```

usbmodem1422 / 115200 8-N-1
Connected 00:04:58

TX RX | RTS CTS DTR DSR DCD RI

Tab-completion in REPL

- Star
  - S
  - It ble
    n

Untitled_0

New Open Save Connect Disconnect Clear Data Options View Hex Help

```
>>> import time
>>> time.
__name__        localtime       mktime          time
sleep           sleep_ms        sleep_us        ticks_ms
ticks_us        ticks_cpu       ticks_add       ticks_diff
>>> time.
```

usbmodem1422 / 115200 8-N-1
Connected 00:05:38

TX RX | RTS CTS DTR DSR DCD RI

In-class Exercise 1

- Print the odd numbers between 1 and 27

- *Hint:* A for loop would be a good way to do this.

In-class Exercise 2

- Print the odd numbers between 1 and 27
- When the number is 13, print “Counter = 13... Bad Luck!!!” and turn on the red LED
- *Hint:* Modify/extend the method you used to solve Exercise 1.