Recap:
Git - Python

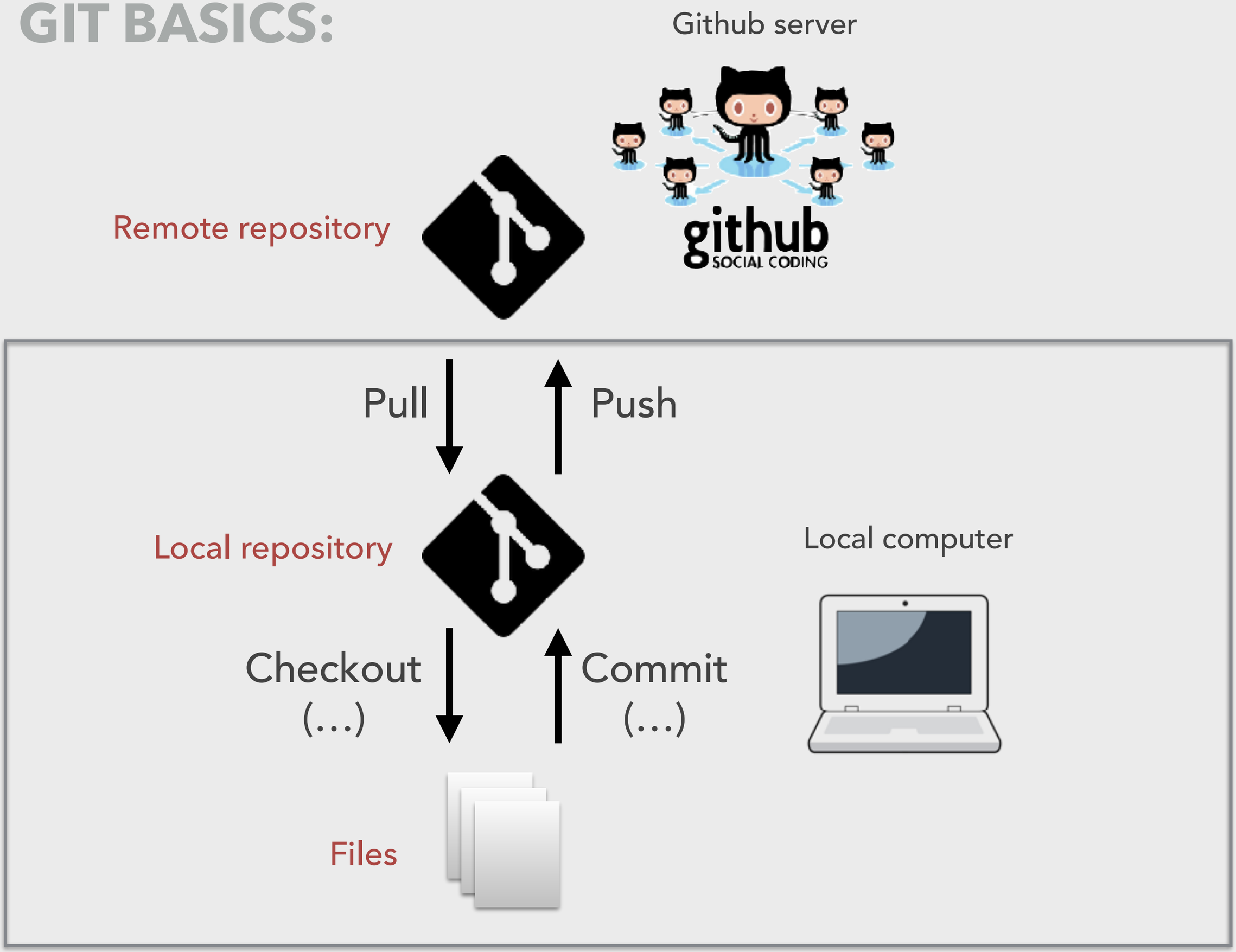
2018-10-05

Git
On 3 Slides

BASH BASICS:

- **cd**: Change current working directory
- **ls**: List files in current working directory
- **pwd**: Print current working directory to command line
- **git:** Operate git for repository of current working directory

GIT BASICS:


Github server
Remote repository
github
SOCIAL CODING
Pull
Push
Local repository
Local computer
Checkout
(…)
Commit
(…)
Files

GIT BASICS:

Cloning an existing project
$ git clone [URL]

Creating a new project
$ git init

Editing your files

Adding new files to repo
$ git add [filename]

Commit all new files
$ git commit -a -m “Useful message”

Accept merge message in vim by typing **:wq** and **RETURN**

Updating the local repo (“pulling”)
$ git pull

Updating the remote repo (“pushing”)
$ git push

Python Basics On 1 Slide

PYTHON BASICS:

- **Variable declarations**
  variable = value

- **Operators**
  - **Arithmetics** +, -, *, /, //, **, %
  - **Comparisons** ==, !=, >, <, <=, >=
  - **Membership** in, not in
  - **Logical** and, or, not

- **Data types**
  - **Floats** -2.342
  - **Integers** 1234
  - **Booleans** True and False
  - **Strings** “whatever” or ‘whatever’
  - **Lists** [item0 ,item1, item2]
  - **Dictionaries** {“key1”: item1, “key2”: item2}

- **If-statements**
  if condition:
  …
  else:
  …

- **Function decalarations**
  def function(argument1, argument2):
  …
  return value
- **Function calls**
  value = function(argument1, argument2)

- **Print function**
  print(value)

- **For-Loops**
  for item in iterable:
  …

- **Import**
  import package
  package.function()

Iterables:
Lists, Tuples, And Strings

Lists

- Lists are mutable ordered containers and can contain items of any type
  $ my_list = [1, 'my string', 1.234, [1, 2, 3]]
  $ my_list = [1, 2, 3, 4]
- New lists items can be added in different ways
  $ my_list = [1, 2, 3] + [4] => [1, 2, 3, 4]
  $ my_list = [1, 2, 3].append(4) => [1, 2, 3, 4]
- List items can be removed by index
  $ removed_item = my_list.pop(0) => 1 and [2,3,4]
- Lists can be sliced
  $ my_list[1:] => [2, 3, 4]
  $ my_list[:-1] => [1,2, 3]
- The order of lists can be reversed
  $ my_list[::-1] => [4, 3, 2, 1]
- Check the length of a list
  $ len(my_list) => 4
- Generate a list of a range of integers
  $ my_list = range(1, 5) => [1, 2, 3, 4]

Tuples

- Tuples are immutable ordered containers and can contain items of any type
  $ my_tuple = (1, ‘my string’, 1.234, [1, 2, 3])
  $ my_tuple = (1, 2, 3, 4)
- Tuples can be sliced
  $ my_tuple[1:] => (2, 3, 4)
  $ my_tuple[:-1] => (1,2, 3)
- The order of tuples can be reversed
  $ my_tuple[::-1] => (4, 3, 2, 1)
- Check the length of a tuple
  $ len(my_tuple) => 4

Strings

- Strings are mutable ordered sequences of characters
  $ my_string = 'whatever you want to write'
- Strings can be sliced the same way as lists and tuples
  $ my_string[9:] => 'you want to write'
  $ my_string[:-9] => 'whatever you want'
- Special characters
  '\n' new line
  '\t' tab space

Using Iterables In Loops

- Examples:
  - Grow a list in a loop

```
$ my_list = [1, 2, 3, 4]
$ my_new_list = []
$ for item in my_list:
$     my_new_list.append(2*item) => ?
```

  - Print every second character in a string

```
$ my_sting = ‘cover’
$ for i in range(0, len(my_string)):
$     if (i % 2) == 0:
$         print(my_string[i]) => ?
```

File I/O

Reading And Writing Files

- Open a file in either write ('w') or read ('r') mode

```
$ file_handle = open('file.txt', 'w')
$ file_handle = open('file.txt', 'r')
```

- Write lines from a list of strings (ending with '\n') to a file

```
$ file_handle = open('file.txt', 'w')
$ file_handle.writelines(list_of_strings)
```

- Read lines from a file to a list of strings (ending with '\n')

```
$ file_handle = open('file.txt', 'r')
$ list_of_strings = file_handle.readlines()
```

- Close file after you are done

```
$ file_handle.close()
```

- Use the with statement to implicitly open and close

```
$ with open('file.txt', 'w') as file_handle:
$    file_handle.writelines(list_of_strings)
```