CSci 127: Introduction to Computer Science

hunter.cuny.edu/csci

Today's Topics

- More Functions
- Github

Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any input parameters, surrounded by parenthesis:
  Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.

In Pairs or Triples:

Predict what the code will do:

```
#Greet loop example

def greetLoop(person):
    print("Greetings")
    for i in range(5):
        print("Hello", person)

greetLoop("Thomas")
```

```
# From "Teaching with Python" by John Zelle

def happy():
    print("Happy Birthday to you!")

def sing(P):
    happy()
    happy()
    print("Happy Birthday dear " + P + "!")
    happy()


sing("Fred")
sing("Thomas")
sing("Hunter")
```

Python Tutor

```
#Greet loop example

def greetLoop(person):
    print("Greetings")
    for i in range(5):
        print("Hello", person)

greetLoop("Thomas")
```

```
# From "Teaching with Python" by John Zelle

def happy():
    print("Happy Birthday to you!")

def sing(P):
    happy()
    happy()
    print("Happy Birthday dear " + P + "!")
    happy()


sing("Fred")
sing("Thomas")
sing("Hunter")
```

(Demo with pythonTutor)

Input Parameters & Return Values

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- Functions can have **input parameters.**
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).
- The "placeholders" in the function definition: **formal parameters.**
- The ones in the function call: **actual parameters.**
- Functions can also **return values** to where it was called.

In Pairs or Triples:

Predict what the code will do:

```
def prob4():
    verse = "jam tomorrow and jam yesterday,"
    print("The rule is,")
    c = mystery(verse)
    w = enigma(verse,c)
    print(c,w)
def mystery(v):
    print(v)
    c = v.count("jam")
    return(c)
def enigma(v,c):
    print("but never", v[-1])
    for i in range(c):
        print("jam")
    return("day.")
prob4()
```

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

Input Parameters

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- When called, the actual parameter values are copied to the formal parameters.
- All the commands inside the function are performed on the copies.
- The actual parameters do not change.
- The copies are discarded when the function is done.
- The time a variable exists is called its **scope**.

Input Parameters: What about Lists?

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

- When called, the actual parameter values are copied to the formal parameters.
- What is copied with a list?
- The address of the list, but not the individual elements.
- The actual parameters do not change, but the inside elements might.
- Easier to see with a demo.

Python Tutor

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

(Demo with pythonTutor)

In Pairs or Triples:

```
def bar(n):
    if n <= 8:
        return 1
    else:
        return 0

def foo(l):
    n = bar(l[-1])
    return l[n]
```

- What is the output of:

```
r = foo([1,2,3,4])
print("Return: ", r)
```

- What is the output of:

```
r = foo([1024,512,256,128])
print("Return: ", r)
```

Python Tutor

```
def bar(n):
    if n <= 8:
        return 1
    else:
        return 0

def foo(l):
    n = bar(l[-1])
    return l[n]
```

(Demo with pythonTutor)

In Pairs or Triples:

```
def prob4(amy, beth):
    if amy > 4:
        print("Easy case")
        kate = -1
    else:
        print("Complex case")
        kate = helper(amy,beth)
    return(kate)
```

```
def helper(meg,jo):
    s = ""
    for j in range(meg):
        print(j, ": ", jo[j])
        if j % 2 == 0:
            s = s + jo[j]
            print("Building s:", s)
    return(s)
```

- What is the output of:

```
r = prob4(4,"city")
print("Return: ", r)
```

- What is the output of:

```
r = prob4(2,"university")
print("Return: ", r)
```

Python Tutor

```
def prob4(amy, beth):
    if amy > 4:
        print("Easy case")
        kate = -1
    else:
        print("Complex case")
        kate = helper(amy,beth)
    return(kate)
```

```
def helper(meg,jo):
    s = ""
    for j in range(meg):
        print(j, ": ", jo[j])
        if j % 2 == 0:
            s = s + jo[j]
            print("Building s:", s)
    return(s)
```

(Demo with pythonTutor)

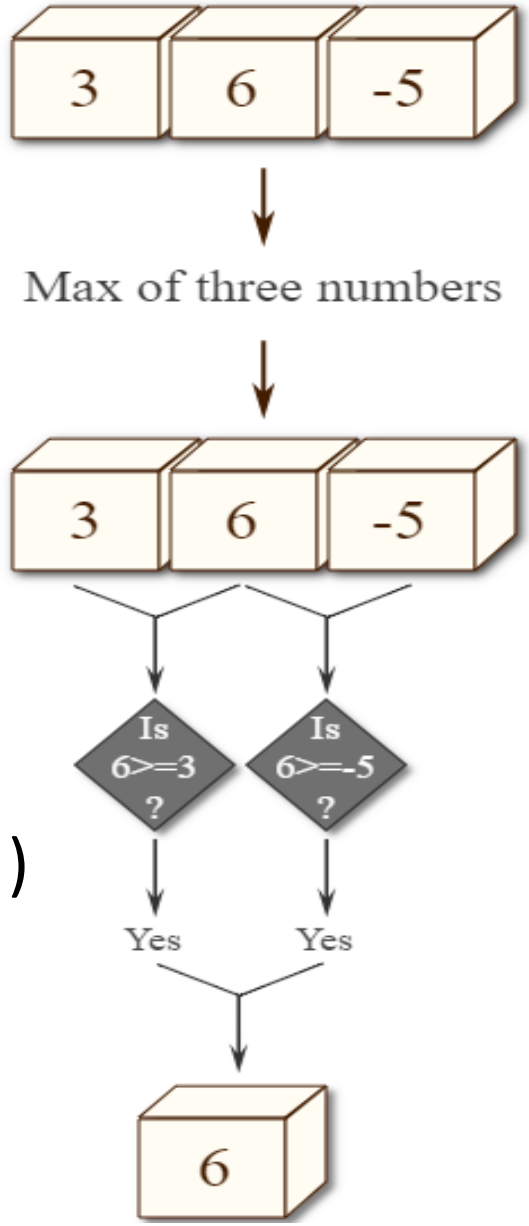
Python Exercises

Exercise – 1 (5 min)

Write a Python function to find the Max of three numbers.

Python Exercises

Exercise – 1 (5 min)

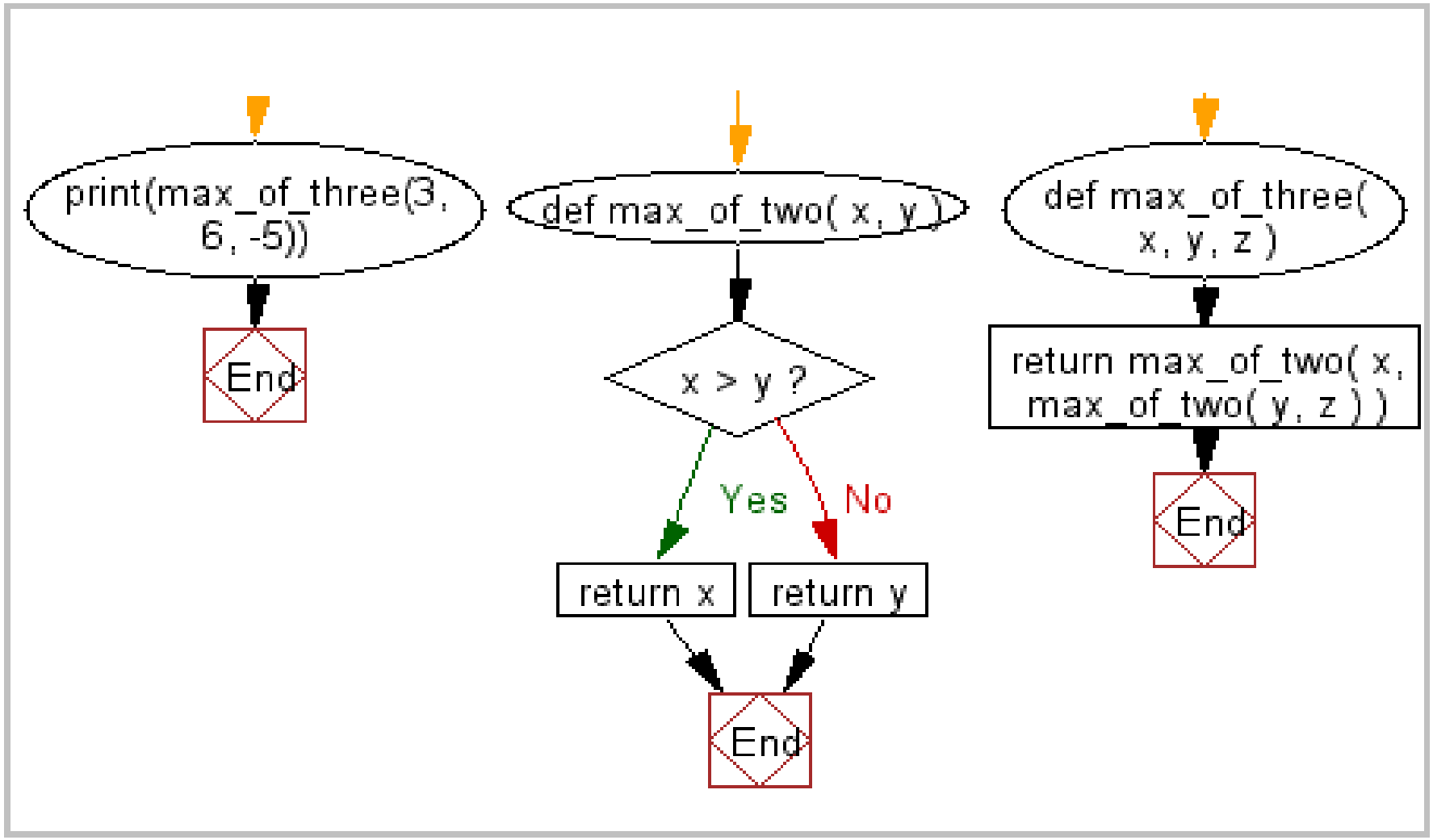



```
def max_of_two( x, y ):
    if x > y:
        return x
    return y
def max_of_three( x, y, z ):
    return max_of_two( x, max_of_two( y, z ) )

print(max_of_three(3, 6, -5))
```

Python Exercises

Exercise – 1 (5 min)



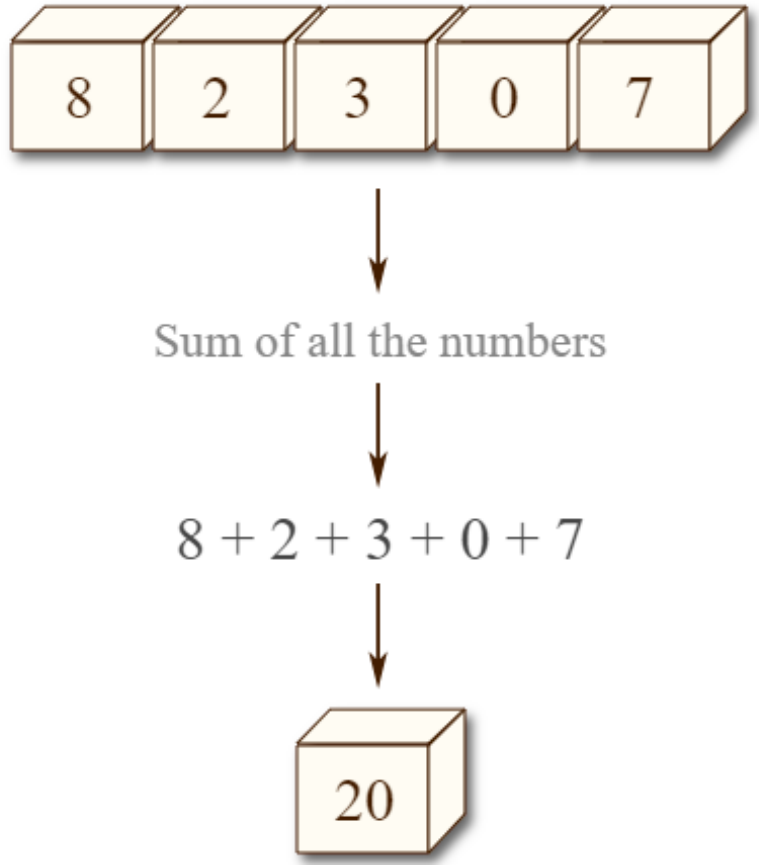
Python Exercises

Exercise – 2 (5 min)

Write a Python function to sum all the numbers in a list

Python Exercises

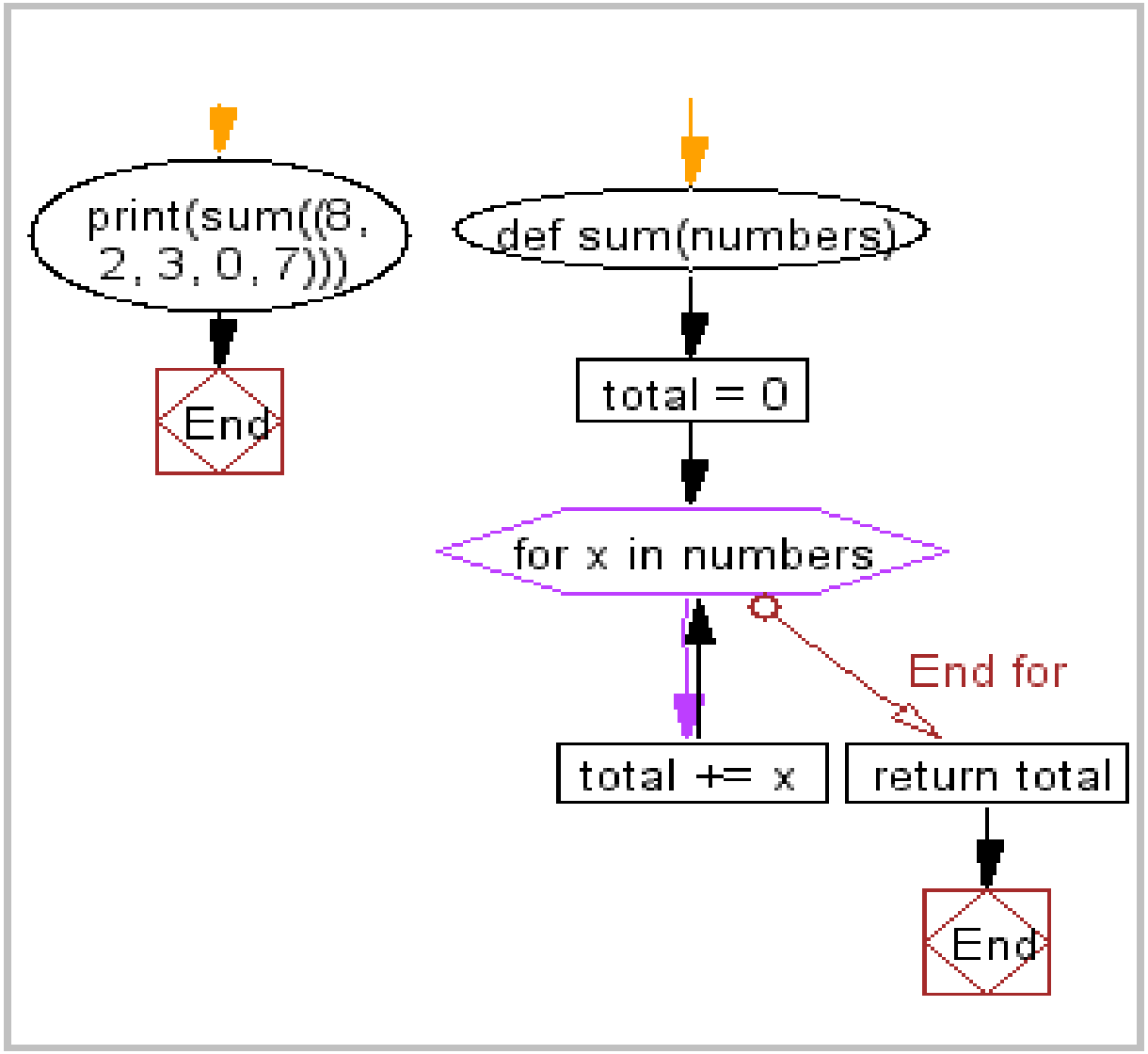
Exercise – 2 (5 min)




```
def sum(numbers):
    total = 0
    for x in numbers:
        total += x
    return total
print(sum([8, 2, 3, 0, 7]))
```

Python Exercises

Exercise – 2 (5 min)



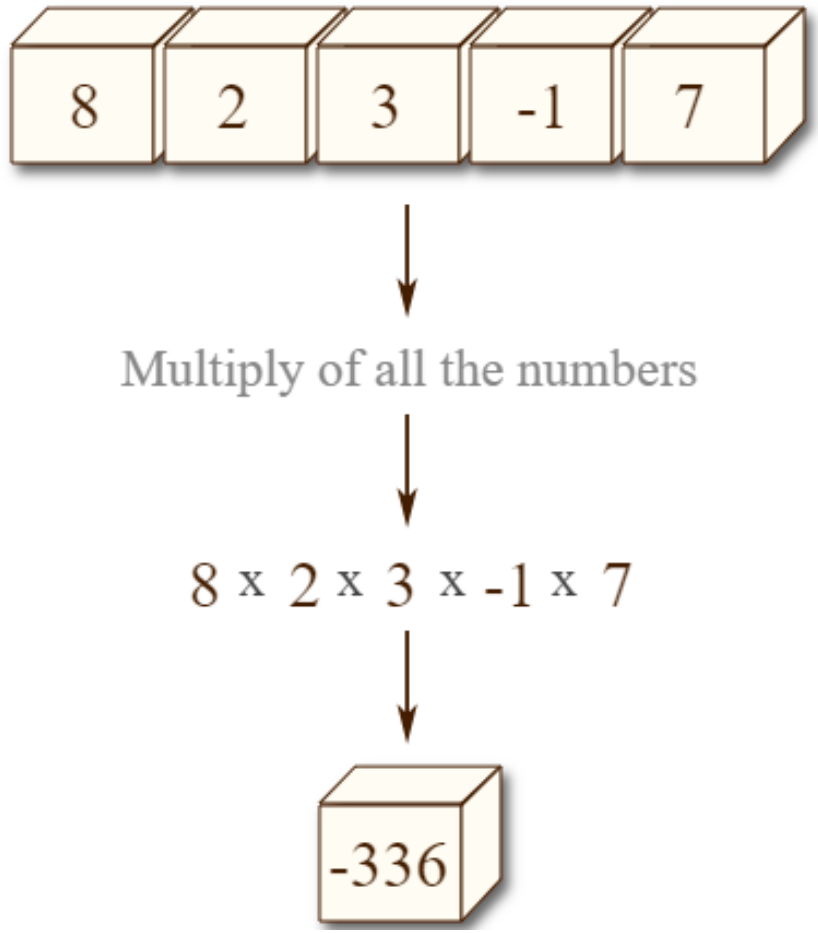
Python Exercises

Exercise – 3 (5 min)

Write a Python function to multiply all the numbers in a list

Python Exercises

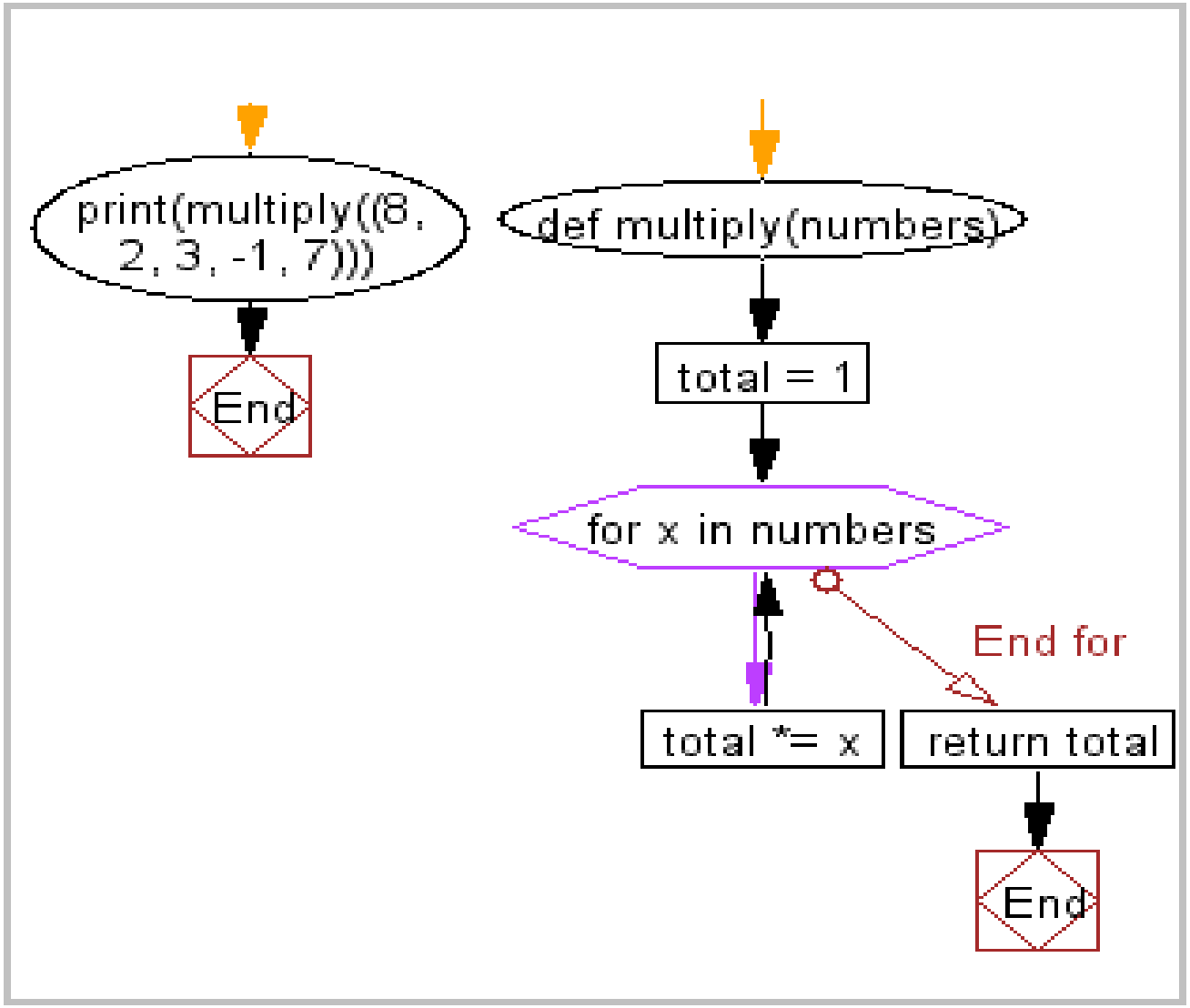
Exercise – 3 (5 min)




```
def multiply(numbers):
    total = 1
    for x in numbers:
        total *= x
    return total
print(multiply([8, 2, 3, -1, 7]))
```

Python Exercises

Exercise – 3 (5 min)



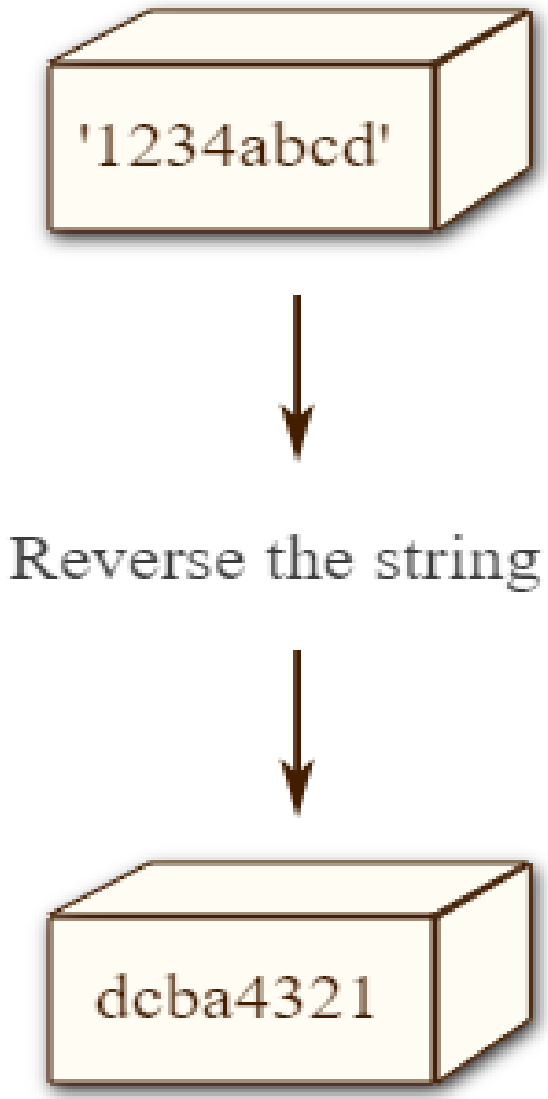
Python Exercises

Exercise – 4 (10 min)

Write a Python program to reverse a string

Python Exercises

Exercise – 4 (Solution)

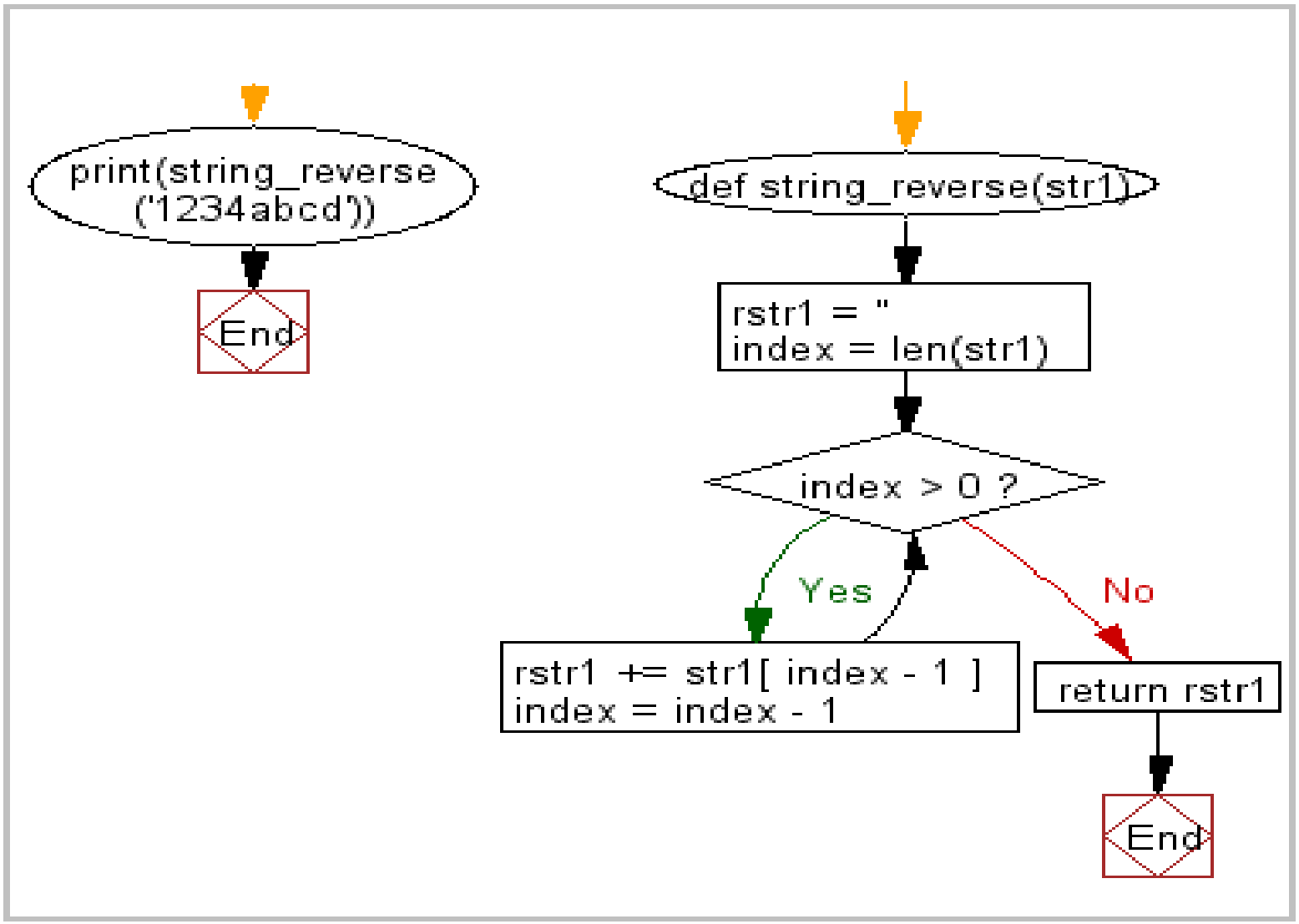



```
def string_reverse(str1):

    rstr1 = ''
    index = len(str1)
    while index > 0:
        rstr1 += str1[ index - 1 ]
        index = index - 1
    return rstr1
print(string_reverse('1234abcd'))
```

Python Exercises

Exercise – 4 (Solution)

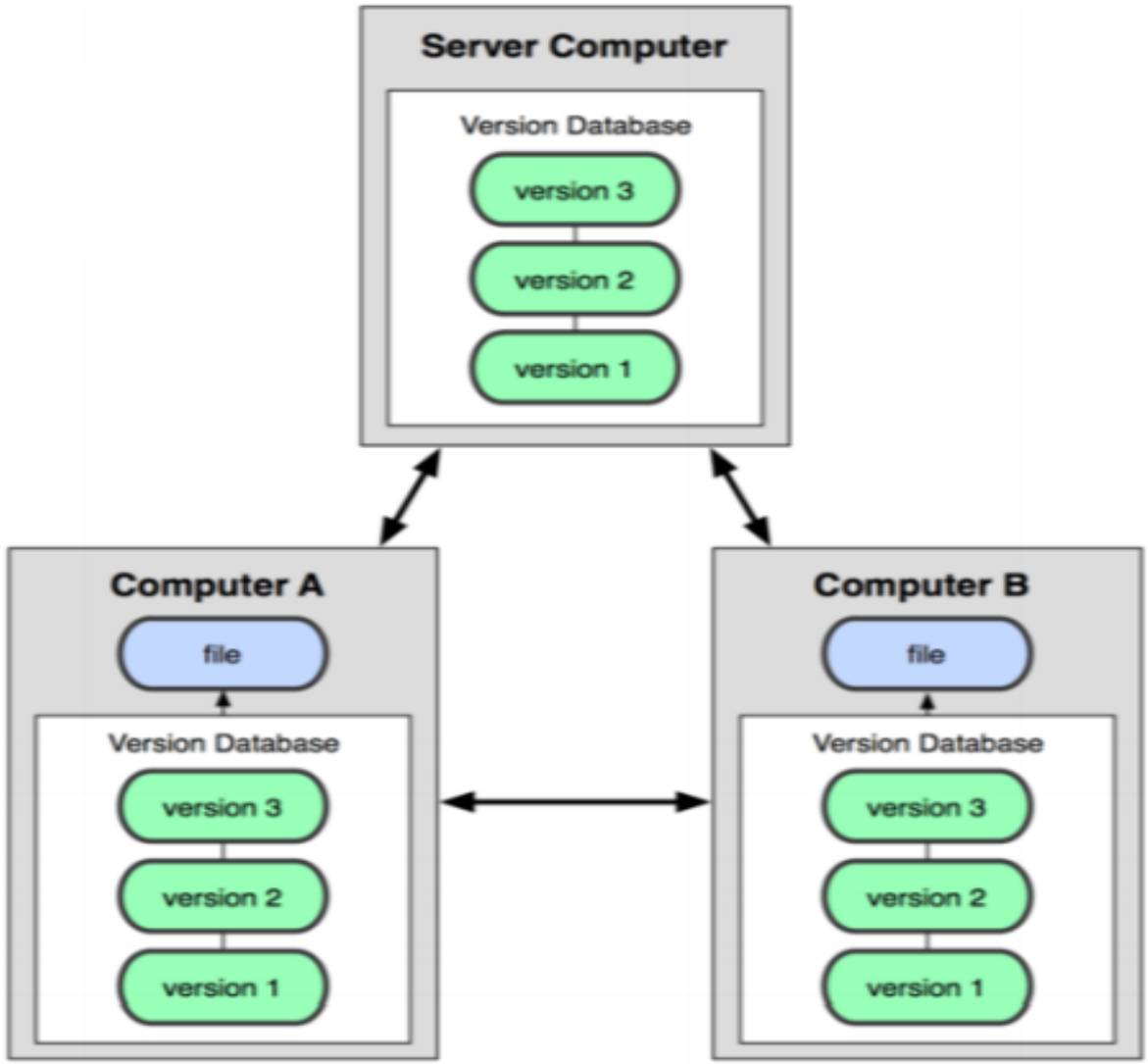


Github – what is it ?

- A central server repository (repo) holds the "official copy" of the code
  - the server maintains the version history of the repo
  - you "clone" it and "pull" changes from it
- Your local repo is a complete copy of everything on the remote server
  - yours is "just as good" as theirs
- Many operations are local:
  - commit changes to local repo
  - local repo keeps version history

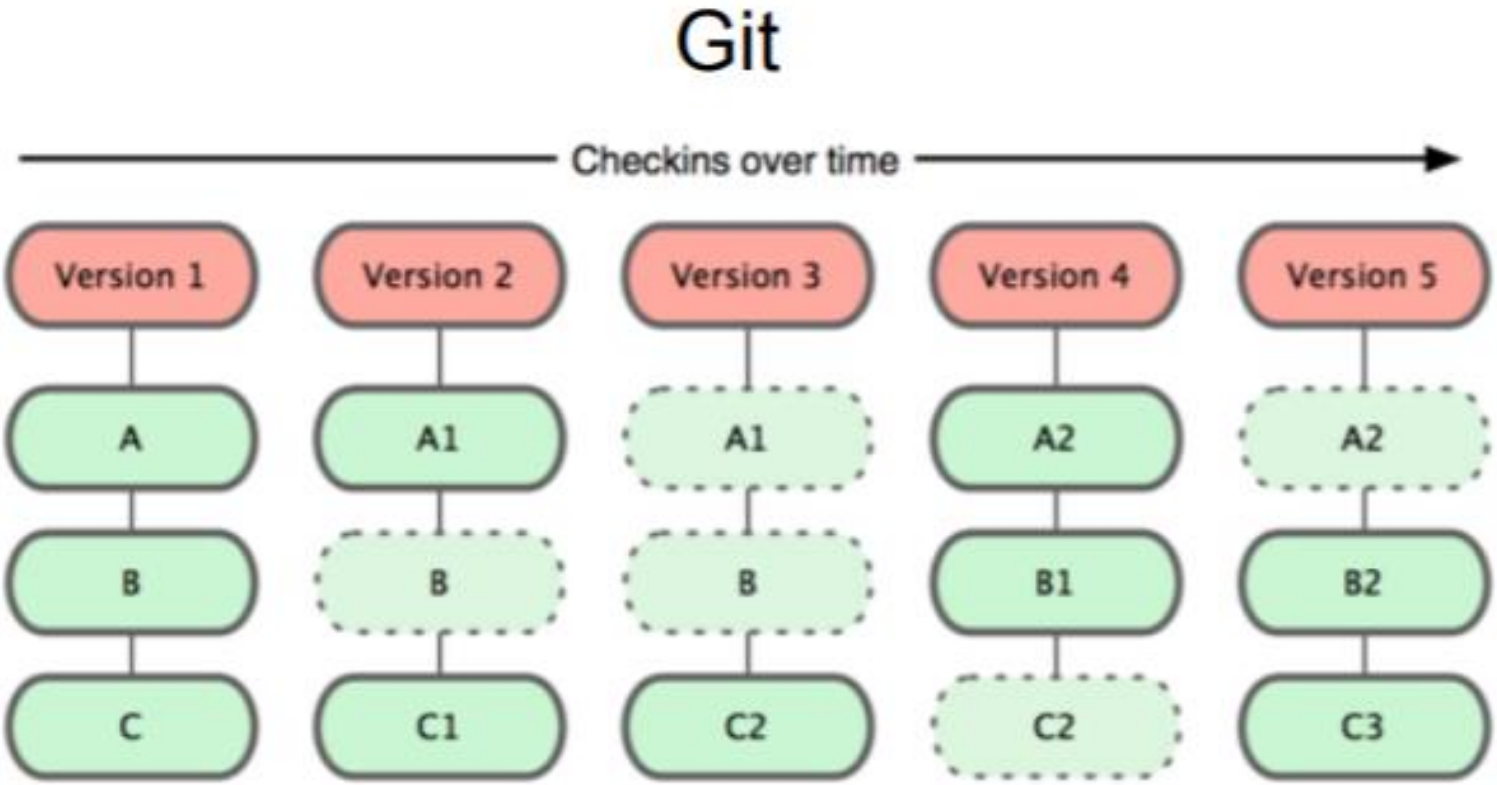
When you're ready, you can "push" changes back to server

Github


Server Computer
Version Database
version 3
version 2
version 1
Computer A
file
Version Database
version 3
version 2
version 1
Computer B
file
Version Database
version 3
version 2
version 1

Git/hub

Git keeps "snapshots" of the entire state of the project.



Git/hub – creating a git repo

Two common scenarios: (only do one of these)

1)To create a new local Git repo in your current directory:

– git init

- This will create a .git directory in your current directory.
- Then you can commit files in that directory into the repo:

– git add filename

– git commit –m "commit message"

2)To clone a remote repo to your current directory:

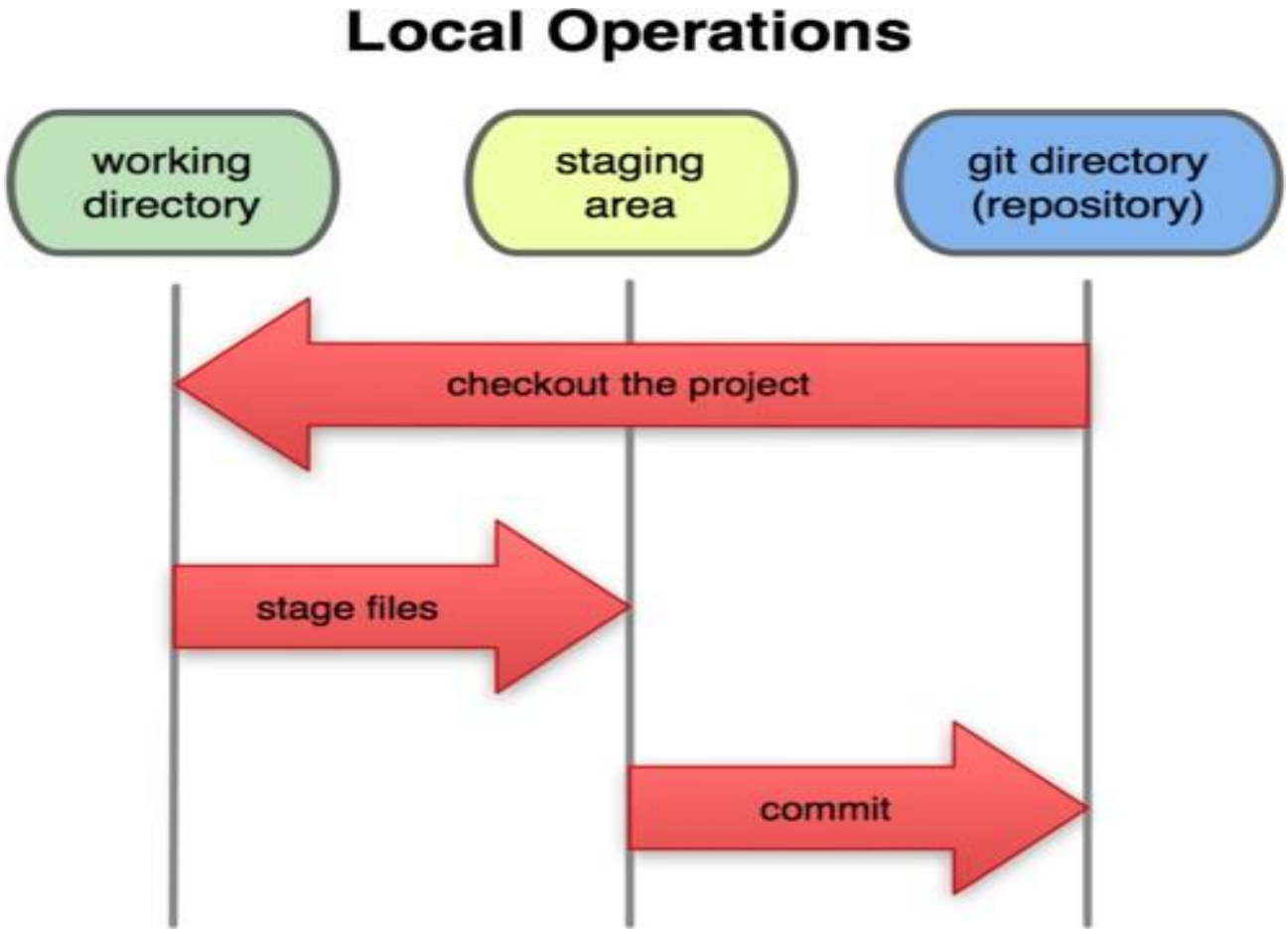
– git clone url localDirectoryName

- This will create the given local directory, containing a working copy of the files from the repo, and a .git directory (used to hold the staging area and your actual local repo)

Git/hub – useful commands

command	description
`git clone` ***`url [dir]`***	copy a Git repository so you can add to it
`git add` ***`file`***	adds file contents to the staging area
`git commit`	records a snapshot of the staging area
`git status`	view the status of your files in the working directory and staging area
`git diff`	shows diff of what is staged and what is modified but unstaged
`git help` ***`[command]`***	get help info about a particular command
`git pull`	fetch from a remote repo and try to merge into the current branch
`git push`	push your new branches and data to a remote repository
others: `init`, `reset`, `branch`, `checkout`, `merge`, `log`, `tag`	

Git/hub


Local Operations
working directory
staging area
git directory (repository)
checkout the project
stage files
commit

Add and commit a file

- The first time we ask a file to be tracked, *and every time before we commit a file*, we must add it to the staging area:
  - `git add hello.py`
    - Takes a snapshot of these files, adds them to the staging area.
    - "add" means "add to staging area" so it will be part of the next commit.
- To move staged changes into the repo, we commit:
  - `git commit –m "Adding new file hello.py"`
- To undo changes on a file before you have committed it:
  - `git reset HEAD -- filename` (unstages the file)
  - All these commands are acting on your local version of repo.

Git/hub - recap

Octocat

- Like Google docs for code...
- Used to share code, documents, etc.
- More formally: `git` is a version control protocol for tracking changes and versions of documents.
- Github provides hosting for repositories (‘**repos**’) of code.
- Also convenient place to host websites (i.e. `mmakkil1.github.io`).