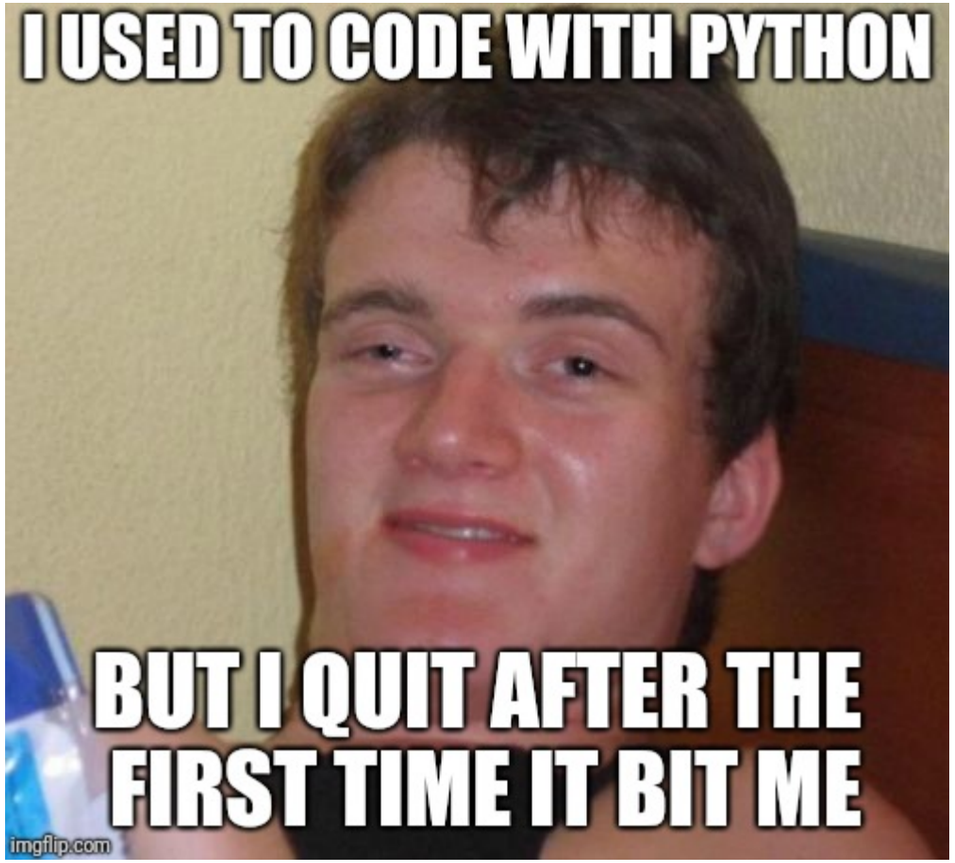
pythonclub 04



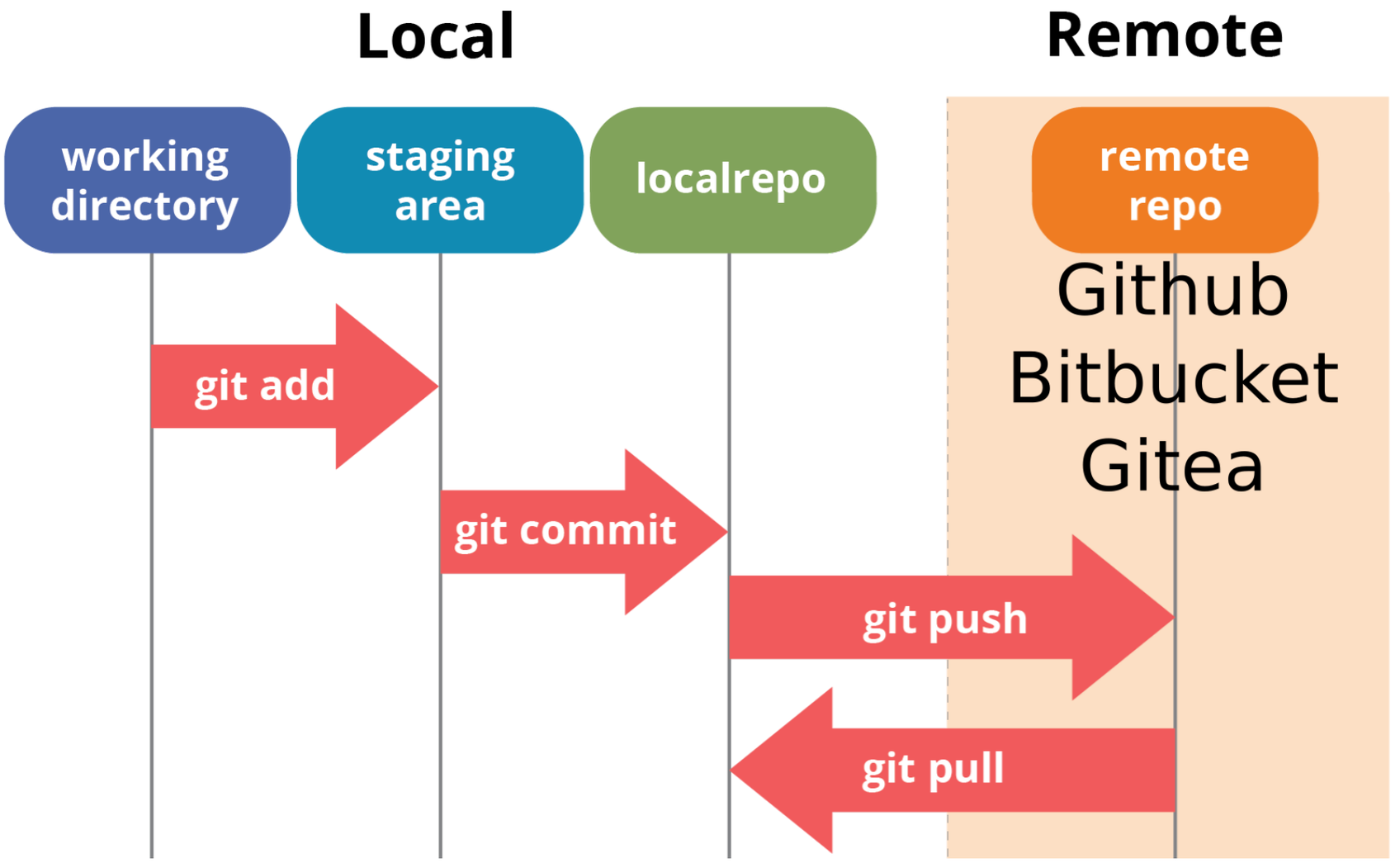
pythonclub 04

Start your VM, open your terminal (`Ctrl+Alt+T`) and try to type the commands shown on screen, we're going to go through:

- git
- regex

Next time: `PROJECT`

git


Local
Remote
working directory
staging area
localrepo
remote repo
Github
Bitbucket
Gitea
git add
git commit
git push
git pull

git: we do it

1. Let's create a repository in the pythhonclub
2. You are now going to pull this repository
    - Go to the repository's webpage: https://github.com/pythonclubmtl/
    - Click `Clone or Download` and copy the repo's link
    - Open your terminal, go to your `Repositories` folder, and input:
      `git clone <repository>`

In the next steps, you are going to send your first contribution to a repoistory.
When you feel that you should submit your code on the remote repository (github), you will have to prepare the data you are sending using commits:

1. We decide what we want to send
2. We put it in a box
3. We add note explaining why we're sending it, then we ship it

git: you do it

3. Create a file in your local repo called : `<yourname>.md`
4. Let's send this file to the distant repo
    - In the repo folder, input :
    i. Check what's up: `git status`
    ii. Put files you want to commit in a box: `git add <yourname>.md`
    iii. Add a note to explain why/what you are comitting:
    `git commit -m "My first commit ever"`
    iv. Send the data: `git push origin master`
    v. Pull to get latest commits: `git pull origin master`

- This the basic git workflow that we'll be using to collaborate (we'll update this later)
- It's usually better to pull before pulling
- Add a message in one of the `<yourname>.md` and push it back

git: explore

Go to the `pythonclubmtl/meetings` repo and click on `XX commits`:

- Click on any commit's name to check modifications
- Click on `Browse files` or `<>` to revert the repository as it was when that commit happened
- Try the split views to easily see updates
- Click next to a line number after clicking on a commit to add a question, remark or blame (tag me using `@ilyasst`)
- Notice those `Merge` commits, we will not expand on those know, but that is what happens when I work on the same file from two different computers

Github is just an interface, you can actually do all of this while offline from your terminal. The `.git` folder in each repo contains all the necessary data.

regular expressions: regexp

> A regular expression, regex or regexp is a sequence of characters that define a search pattern. Usually this pattern is used by string searching algorithms for "find" or "find and replace" operations on strings, or for input validation.

- Open regexone.com/lesson and go through **lessons 1 to 5**.
- Open your terminal and clone the repository: `pythonclubtmtl/learning_python3`
- We are going to use the grep command to find all occurrences of the word `python` in the file `002-using-pythonshell.md`. In the `learning_python3` folder from your terminal: `grep "python" 002-using-pythonshell.md`
- In the whole repository: `grep -r "python" <path>`
  - Reminder `.` means here (in the current folder)
- Use regular expressions instead of a string to find **all occurrences of double digit numbers** (ex: `42`, `51`,...)
- Use regular expressions to find **all occurrences of any letter followed by a single digit** (ex: `k3`)

regexp in python: try it

Let's find all double digits from a string (python console):

```
# regex pacakge
>>> import re
# We're looking for double digits only
>>> regex = r"[0-9][0-9]"
# some random text
>>> text = "Hi 42, it's me, 24"
# Get strings that fit regex (occurrences)
>>> matches = re.findall(regex, text)
>>> matches
# Get all occurrences with position
>>> matches = re.finditer(regex, text)
>>> for match in matches:
#       occurrence    , char # start , char # end
...     match.group(0), match.start(), match.end()
...
```

regexp: you do it now

- Open your "baby name parser" script from the previous session
- Modify your script to return:
  - All female names that contain (anywhere) the letter `c` followed by any letter, then the letter `a` (`c*a***` or `**c*a**` or `******c*a` or `...`), then find the most popular one

> Note: re.findall(regex, text, re.IGNORECASE) will make re case insensitive (as if all characters from the text are lower case.

- Next time, we will tokenize a corpus and work toward getting its keywords.