

---

## Gittle - Pythonic Git for Humans

Gittle is a high-level pure-python git library. It builds upon dulwich which provides most of the low-level machinery

### Install it

```
1 pip install gittle
```

### Examples :

#### Clone a repository

```
1 from gittle import Gittle
2
3 repo_path = '/tmp/gittle_bare'
4 repo_url = 'git://github.com/FriendCode/gittle.git'
5
6 repo = Gittle.clone(repo_url, repo_path)
```

With authentication (see Authentication section for more information) :

```
1 auth = GittleAuth(pkey=key)
2 Gittle.clone(repo_url, repo_path, auth=auth)
```

Or clone bare repository (no working directory) :

```
1 repo = Gittle.clone(repo_url, repo_path, bare=True)
```

#### Init repository from a path

```
1 repo = Gittle.init(path)
```

#### Get repository information

```
1 # Get list of objects
2 repo.commits
3
4 # Get list of branches
5 repo.branches
6
```

---

```
7 # Get list of modified files (in current working directory)
8 repo.modified_files
9
10 # Get diff between latest commits
11 repo.diff('HEAD', 'HEAD~1')
```

## Commit

```
1 # Stage single file
2 repo.stage('file.txt')
3
4 # Stage multiple files
5 repo.stage(['other1.txt', 'other2.txt'])
6
7 # Do the commit
8 repo.commit(name="Samy Pesse", email="samy@friendco.de", message="This
   is a commit")
```

## Pull

```
1 repo = Gittle(repo_path, origin_uri=repo_url)
2
3 # Authentication with RSA private key
4 key_file = open('/Users/Me/keys/rsa/private_rsa')
5 repo.auth(pkey=key_file)
6
7 # Do pull
8 repo.pull()
```

## Push

```
1 repo = Gittle(repo_path, origin_uri=repo_url)
2
3 # Authentication with RSA private key
4 key_file = open('/Users/Me/keys/rsa/private_rsa')
5 repo.auth(pkey=key_file)
6
7 # Do push
8 repo.push()
```

## Authentication for remote operations

---

```
1 # With a key
2 key_file = open('/Users/Me/keys/rsa/private_rsa')
3 repo.auth(pkey=key_file)
4
5 # With username and password
6 repo.auth(username="your_name", password="your_password")
```

## Branch

```
1 # Create branch off master
2 repo.create_branch('dev', 'master')
3
4 # Checkout the branch
5 repo.switch_branch('dev')
6
7 # Create an empty branch (like 'git checkout --orphan')
8 repo.create_orphan_branch('NewBranchName')
9
10 # Print a list of branches
11 print(repo.branches)
12
13 # Remove a branch
14 repo.remove_branch('dev')
15
16 # Print a list of branches
17 print(repo.branches)
```

## Get file version

```
1 versions = repo.get_file_versions('gittle/gittle.py')
2 print("Found %d versions out of a total of %d commits" % (len(versions)
    , repo.commit_count()))
```

## Get list of modified files (in current working directory)

```
1 repo.modified_files
```

## Count number of commits

```
1 repo.commit_count
```

---

## Get information for commits

List commits :

```
1 # Get 20 first commits
2 repo.commit_info(start=0, end=20)
```

With a given commit :

```
1 commit = "a2105a0d528bf770021de874baf72ce36f6c3ccc"
```

Diff with another commit :

```
1 old_commit = repo.get_previous_commit(commit, n=1)
2 print repo.diff(commit, old_commit)
```

Explore commit files using :

```
1 commit = "a2105a0d528bf770021de874baf72ce36f6c3ccc"
2
3 # Files tree
4 print repo.commit_tree(commit)
5
6 # List files in a subpath
7 print repo.commit_ls(commit, "testdir")
8
9 # Read a file
10 print repo.commit_file(commit, "testdir/test.txt")
```

## Create a GIT server

```
1 from gittle import GitServer
2
3 # Read only
4 GitServer('/', 'localhost').serve_forever()
5
6 # Read/Write
7 GitServer('/', 'localhost', perm='rw').serve_forever()
```

## Why implement Git in Python ?

### NEED FOR AWESOMENESS :

- Git is Awesome
- Python is Awesome

- 
- Automating Git isn't so Awesome

### **TO SOLVE MY OWN PROBLEMS AT FRIENDCODE :**

- Automate git repo management (push/pull, commit, etc ...)
- Scriptable and usable from Python
- Easy to use & good interoperability in a SOA environment

### **USE IT FOR :**

- Local
  - ☒ Common git operations (add, rm, mv, commit, log)
  - ☒ Branch operations (creating, switching, deleting)
- Remote
  - ☒ Fetching
  - ☒ Pushing
  - ☒ Pulling (needs merging)
- Merging
  - [-] Fast forward
  - [-] Recursive
  - [-] Merge branches
- Diff
  - ☒ Filter binary files

## **Building and uploading to PyPi**

```
1 python setup.py sdist bdist_egg upload
```