Quick Package Development in R and python

Theodore Bakanas

ODSC East - 2020-04-15

I’m Ted Bakanas, a Staff Data Scientist at Uptake

What motivated this workshop? At whom is it aimed?

This started out largely aimed at myself!

- Myself when I started as a data scientist:
  - cover gaps in knowledge which would have driven greater understanding of what I was using
- Myself now:
  - provide a template of package development and common code patterns in R and python
  - other examples

What will we cover?

The starterkits repo contains two packages for hitting the Swiss Public Transit API

- Repo and package basics
- Installation & uninstallation
- Functions & logging
- Iteration
- Documentation
- Getting your code to show up!
- Command line executable scripts
- Appendix: objects and APIs

Why do we want to package up our work?

Theme: teamwork and thought patterns

- encourages and enables collaboration
- forces us to think about parameterization
- forces us to think about reusability

What are some situations in which to think about making a package?

Theme: reusable

- encoding access to a database
- creating standard visualizations
- recurring data cleansing

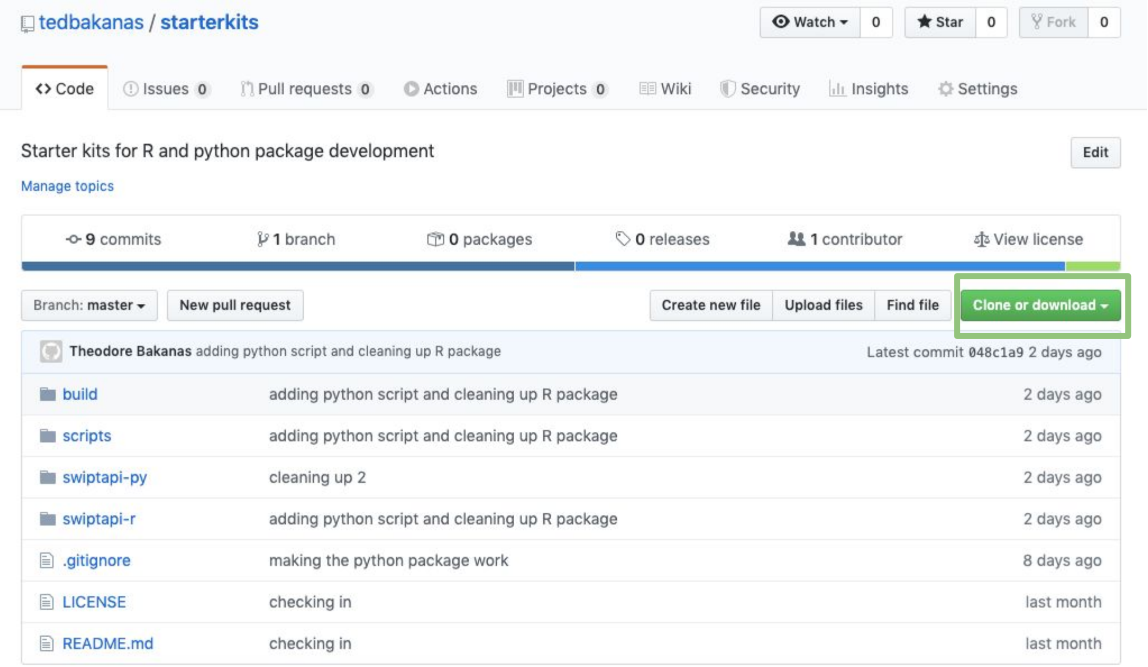
Getting set up to follow along:

Go this this github repo: https://github.com/tedbakanas/starterkits

Clone or download the zip file!

Some prerequisites:

- R installed
- python 3 installed
- a file browser/text editor
  - (sublime text, RStudio...)
- terminal



Starter kits for R and python package development

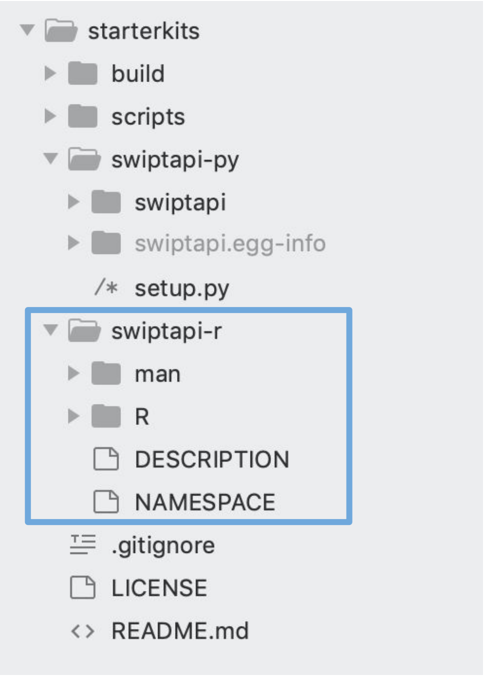
Theodore Bakanas adding python script and cleaning up R package — Latest commit 048c1a9 2 days ago

build	adding python script and cleaning up R package	2 days ago
scripts	adding python script and cleaning up R package	2 days ago
swiptapi-py	cleaning up 2	2 days ago
swiptapi-r	adding python script and cleaning up R package	2 days ago
.gitignore	making the python package work	8 days ago
LICENSE	checking in	last month
README.md	checking in	last month

How is this going to work?

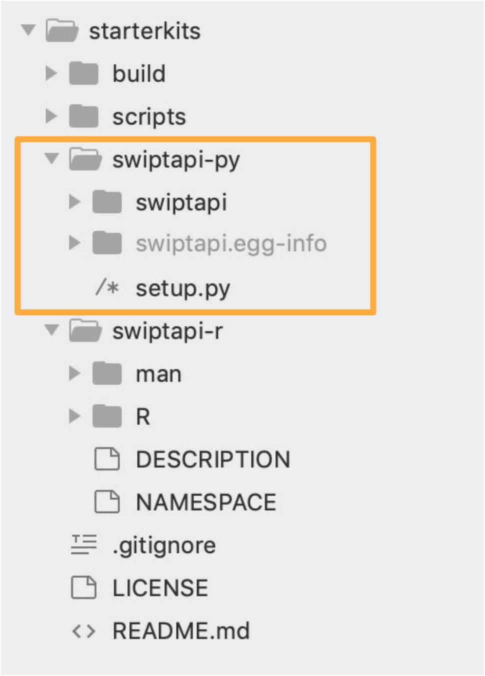
Lumping similar steps in R and python promotes understanding the concept as a whole

R



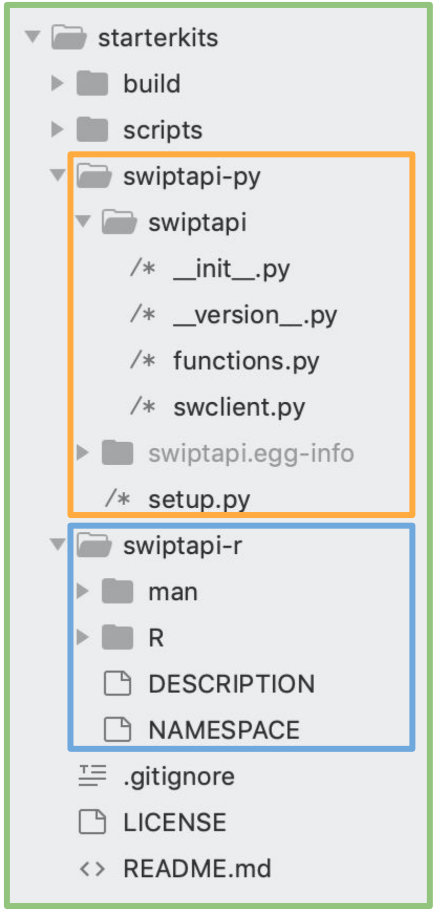
```
starterkits
  build
  scripts
  swiptapi-py
    swiptapi
    swiptapi.egg-info
    setup.py
  swiptapi-r
    man
    R
    DESCRIPTION
    NAMESPACE
  .gitignore
  LICENSE
  README.md
```

python



```
starterkits
  build
  scripts
  swiptapi-py
    swiptapi
    swiptapi.egg-info
    setup.py
  swiptapi-r
    man
    R
    DESCRIPTION
    NAMESPACE
  .gitignore
  LICENSE
  README.md
```

Repo & Package Basics



```
starterkits
  build
  scripts
  swiptapi-py
    swiptapi
      __init__.py
      __version__.py
      functions.py
      swclient.py
    swiptapi.egg-info
    setup.py
  swiptapi-r
    man
    R
    DESCRIPTION
    NAMESPACE
  .gitignore
  LICENSE
  README.md
```

Repository:

- a single git controlled suite of files and folders
  - .gitignore allows certain files to not be tracked by git (.egg-info)
- can contain multiple packages
- can hold other files and folders (build, scripts, README.md)

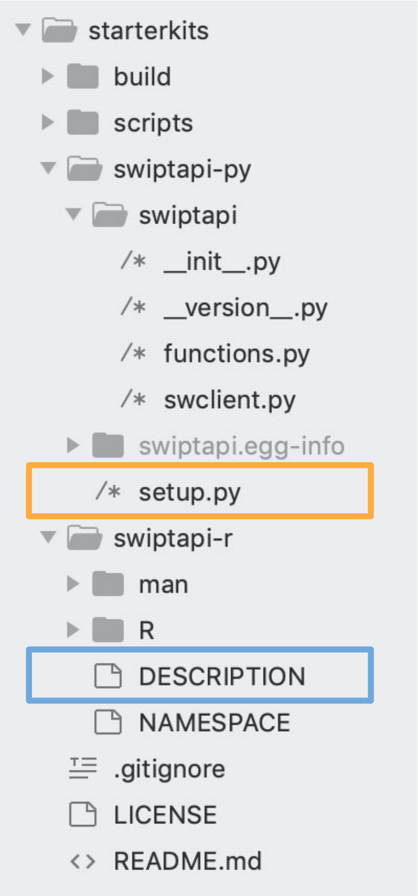
python

- “swiptapi” folder contains the .py files with the functions and objects
- setup.py contains the general package info
- __version__.py contains the package version
- __init__.py contains the outward facing names (more later)

R package:

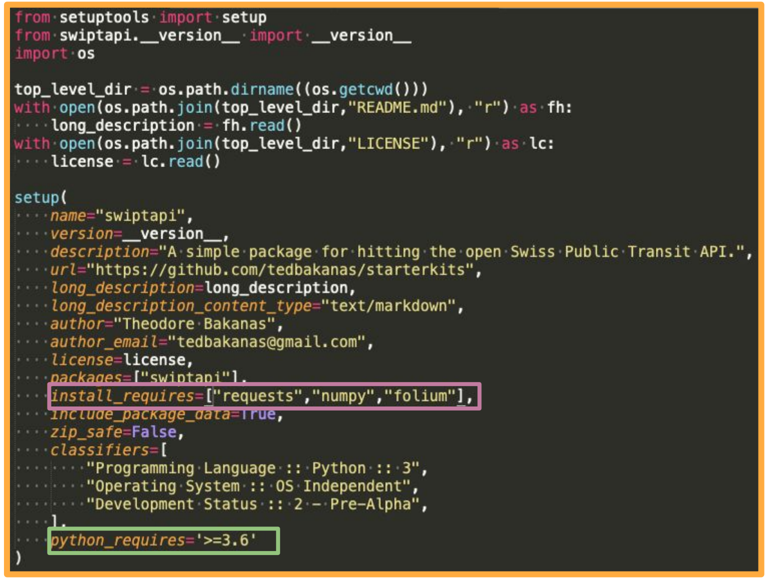
- “R” folder contains the .R files with the functions and objects
- “man” folder contains the docs (more on this later)
- NAMESPACE contains the functions imported and the outward facing names
- DESCRIPTION contains general package info

Repo & Package Basics

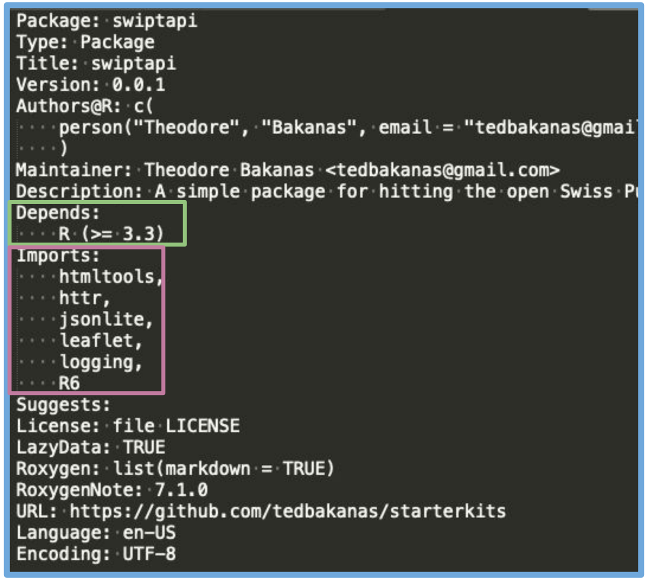


```
starterkits
    build
    scripts
    swiptapi-py
        swiptapi
            /* __init__.py
            /* __version__.py
            /* functions.py
            /* swclient.py
        swiptapi.egg-info
        /* setup.py
    swiptapi-r
        man
        R
        DESCRIPTION
        NAMESPACE
    .gitignore
    LICENSE
    <> README.md
```

- the setup.py file and the DESCRIPTION file serve equivalent purposes
  - setup.py is used more directly in installation
- provides general info (name, description, version, author, license...)
- specifies base language version
- specifies required packages

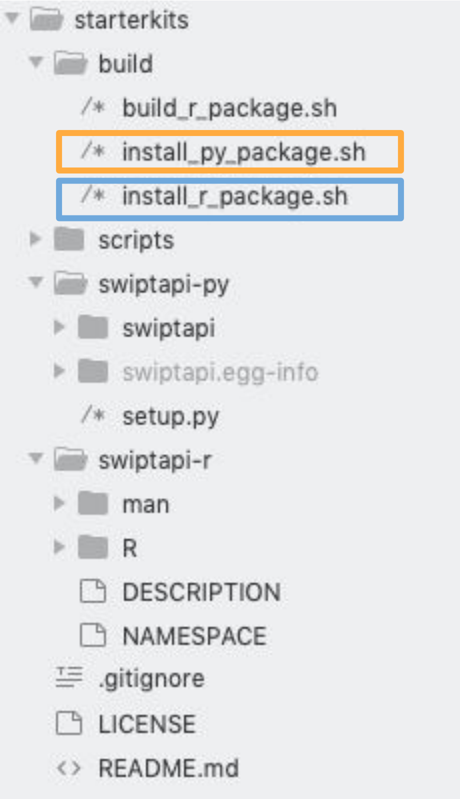


```python
from setuptools import setup
from swiptapi.__version__ import __version__
import os

top_level_dir = os.path.dirname((os.getcwd()))
with open(os.path.join(top_level_dir,"README.md"), "r") as fh:
    long_description = fh.read()
with open(os.path.join(top_level_dir,"LICENSE"), "r") as lc:
    license = lc.read()

setup(
    name="swiptapi",
    version=__version__,
    description="A simple package for hitting the open Swiss Public Transit API.",
    url="https://github.com/tedbakanas/starterkits",
    long_description=long_description,
    long_description_content_type="text/markdown",
    author="Theodore Bakanas",
    author_email="tedbakanas@gmail.com",
    license=license,
    packages=["swiptapi"],
    install_requires=["requests","numpy","folium"],
    include_package_data=True,
    zip_safe=False,
    classifiers=[
        "Programming Language :: Python :: 3",
        "Operating System :: OS Independent",
        "Development Status :: 2 - Pre-Alpha",
    ],
    python_requires='>=3.6'
)
```



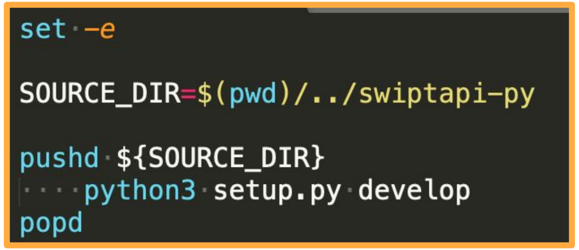
```
Package: swiptapi
Type: Package
Title: swiptapi
Version: 0.0.1
Authors@R: c(
    person("Theodore", "Bakanas", email = "tedbakanas@gmai
    )
Maintainer: Theodore Bakanas <tedbakanas@gmail.com>
Description: A simple package for hitting the open Swiss Pu
Depends:
    R (>= 3.3)
Imports:
    htmltools,
    httr,
    jsonlite,
    leaflet,
    logging,
    R6
Suggests:
License: file LICENSE
LazyData: TRUE
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.1.0
URL: https://github.com/tedbakanas/starterkits
Language: en-US
Encoding: UTF-8
```

Installation

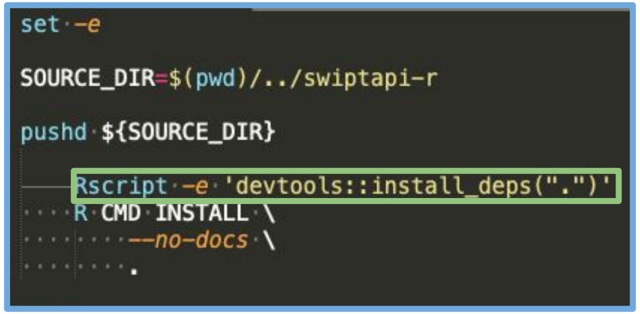


```
▼ starterkits
  ▼ build
      build_r_package.sh
      install_py_package.sh
      install_r_package.sh
  ▶ scripts
  ▼ swiptapi-py
    ▶ swiptapi
    ▶ swiptapi.egg-info
      setup.py
  ▼ swiptapi-r
    ▶ man
    ▶ R
      DESCRIPTION
      NAMESPACE
    .gitignore
    LICENSE
    README.md
```

- both R and python provide means of installing packages from source
  - python invokes setup.py file with the additional parameter “develop”
  - R uses “R CMD INSTALL” from within the package directory
- by wrapping these installations in shell scripts it is easier to call them in a repeatable manner
- both commands should be executed from the directory which contains the package.
  - “pushd” allows for the resetting of the working directory
- required packages (see previous slide)
  - python attempts to install required packages automatically
  - R we add a command to do this within the shell script

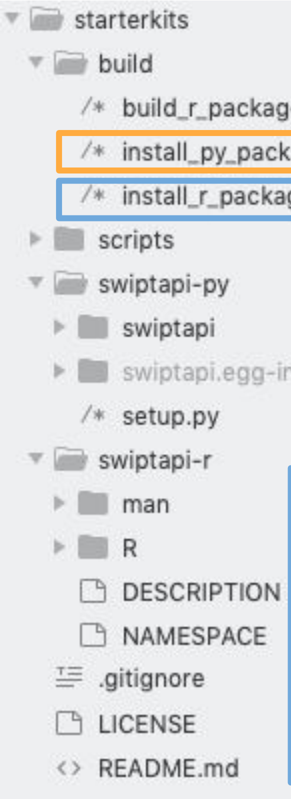


```
set -e

SOURCE_DIR=$(pwd)/../swiptapi-py

pushd ${SOURCE_DIR}
    python3 setup.py develop
popd
```



```
set -e

SOURCE_DIR=$(pwd)/../swiptapi-r

pushd ${SOURCE_DIR}
    Rscript -e 'devtools::install_deps(".")'
    R CMD INSTALL \
        --no-docs \
        .
```

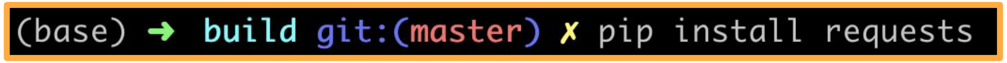
Installation



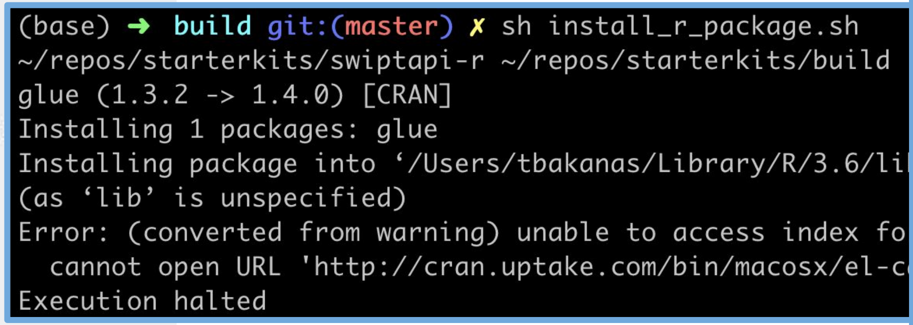
```
starterkits
  build
    build_r_package.sh
    install_py_package.sh
    install_r_package.sh
  scripts
  swiptapi-py
    swiptapi
    swiptapi.egg-info
    setup.py
  swiptapi-r
    man
    R
    DESCRIPTION
    NAMESPACE
  .gitignore
  LICENSE
  README.md
```

Often it doesn’t work!

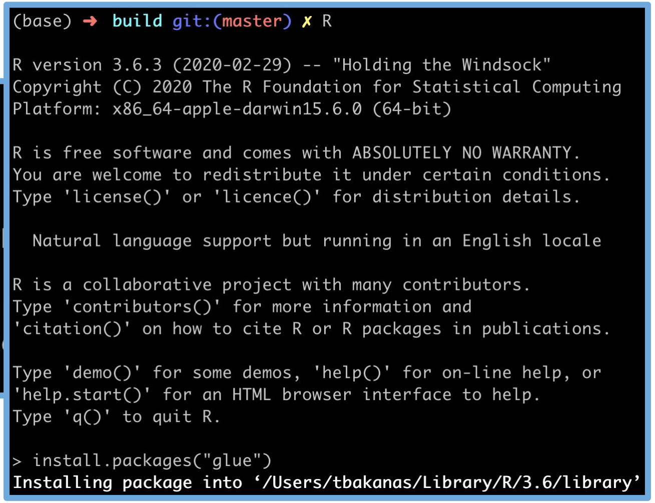
- read the error log
- if a dependency install errors out, try installing it directly
  - R: open R and use install.packages
  - py: pip install!



```
(base) ➜  build git:(master) ✗ pip install requests
```

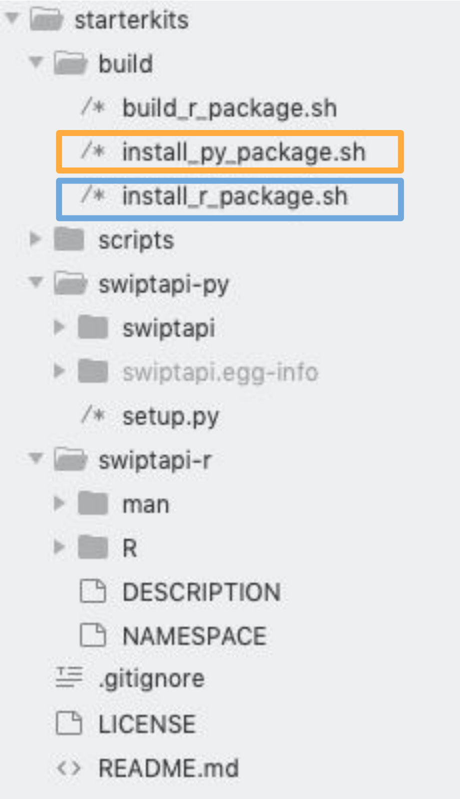


```
(base) ➜  build git:(master) ✗ sh install_r_package.sh
~/repos/starterkits/swiptapi-r ~/repos/starterkits/build
glue (1.3.2 -> 1.4.0) [CRAN]
Installing 1 packages: glue
Installing package into ‘/Users/tbakanas/Library/R/3.6/li
(as ‘lib’ is unspecified)
Error: (converted from warning) unable to access index fo
  cannot open URL 'http://cran.uptake.com/bin/macosx/el-c
Execution halted
```



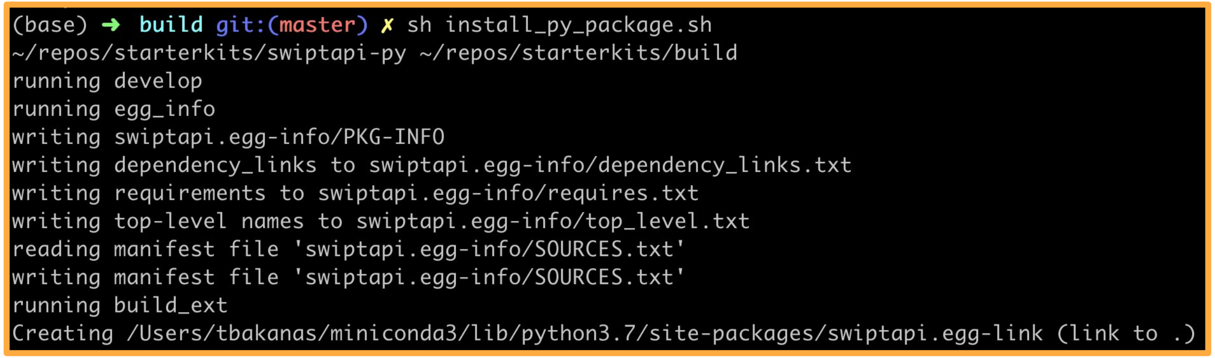
```
(base) ➜  build git:(master) ✗ R

R version 3.6.3 (2020-02-29) -- "Holding the Windsock"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> install.packages("glue")
Installing package into ‘/Users/tbakanas/Library/R/3.6/library’
```

Installation

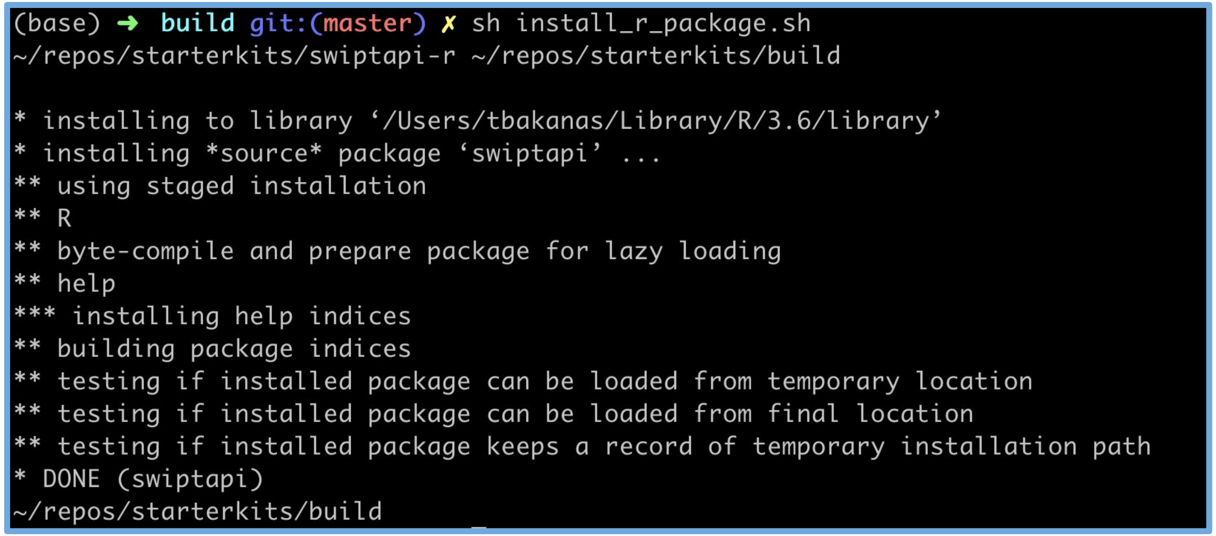


```
starterkits
  build
    build_r_package.sh
    install_py_package.sh
    install_r_package.sh
  scripts
  swiptapi-py
    swiptapi
    swiptapi.egg-info
    setup.py
  swiptapi-r
    man
    R
    DESCRIPTION
    NAMESPACE
  .gitignore
  LICENSE
  README.md
```

When it works:



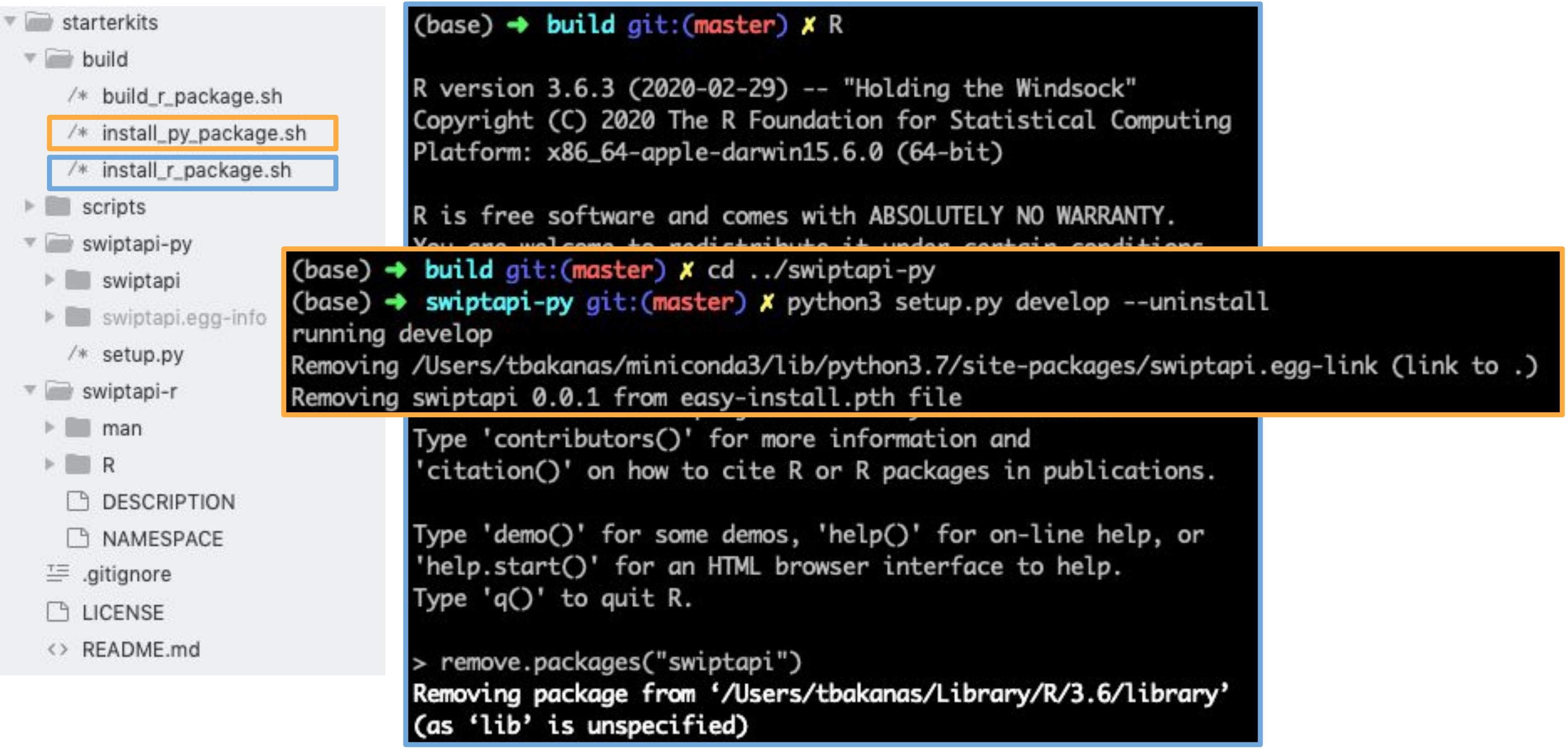
```
(base) ➜ build git:(master) ✗ sh install_py_package.sh
~/repos/starterkits/swiptapi-py ~/repos/starterkits/build
running develop
running egg_info
writing swiptapi.egg-info/PKG-INFO
writing dependency_links to swiptapi.egg-info/dependency_links.txt
writing requirements to swiptapi.egg-info/requires.txt
writing top-level names to swiptapi.egg-info/top_level.txt
reading manifest file 'swiptapi.egg-info/SOURCES.txt'
writing manifest file 'swiptapi.egg-info/SOURCES.txt'
running build_ext
Creating /Users/tbakanas/miniconda3/lib/python3.7/site-packages/swiptapi.egg-link (link to .)
```



```
(base) ➜ build git:(master) ✗ sh install_r_package.sh
~/repos/starterkits/swiptapi-r ~/repos/starterkits/build

* installing to library ‘/Users/tbakanas/Library/R/3.6/library’
* installing *source* package ‘swiptapi’ ...
** using staged installation
** R
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (swiptapi)
~/repos/starterkits/build
```

Un-installation

You can remove these later!



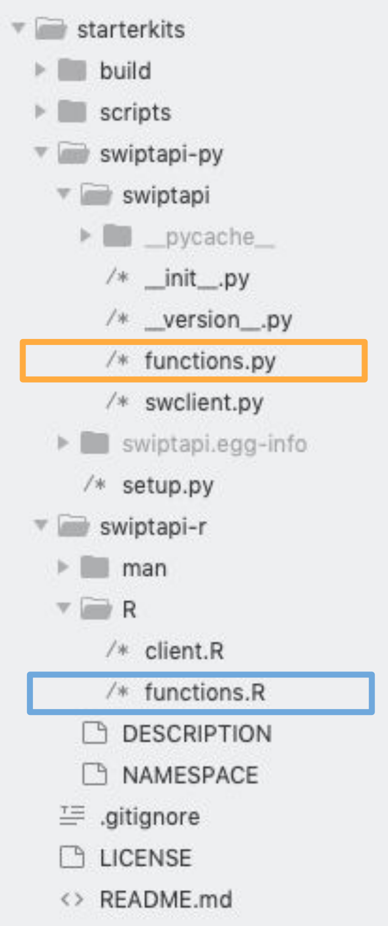
```
starterkits
  build
    build_r_package.sh
    install_py_package.sh
    install_r_package.sh
  scripts
  swiptapi-py
    swiptapi
    swiptapi.egg-info
    setup.py
  swiptapi-r
    man
    R
    DESCRIPTION
    NAMESPACE
  .gitignore
  LICENSE
  README.md
```

```
(base) ➜ build git:(master) ✗ R

R version 3.6.3 (2020-02-29) -- "Holding the Windsock"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions
```

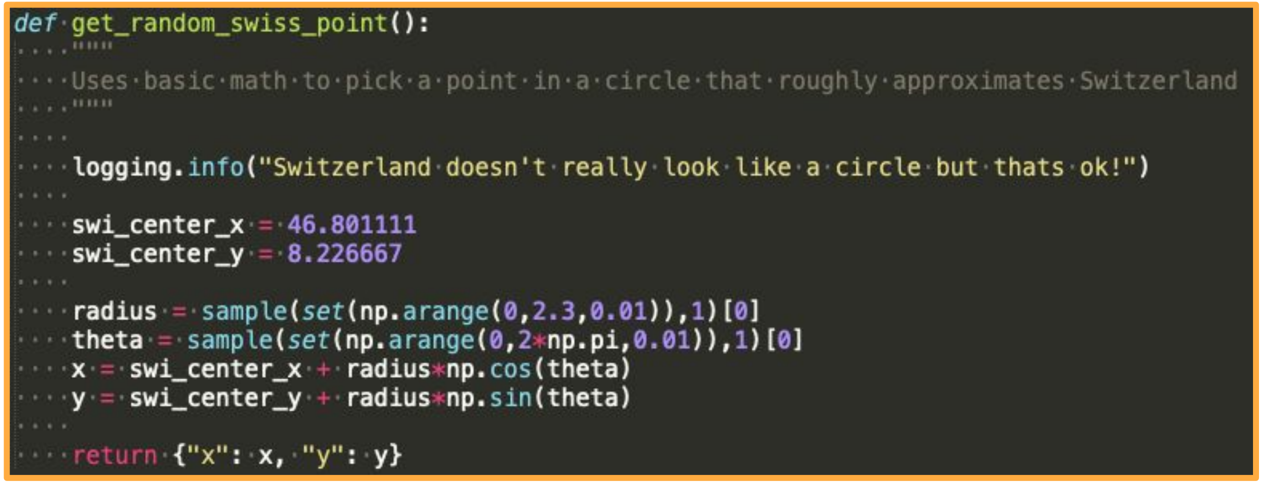
```
(base) ➜ build git:(master) ✗ cd ../swiptapi-py
(base) ➜ swiptapi-py git:(master) ✗ python3 setup.py develop --uninstall
running develop
Removing /Users/tbakanas/miniconda3/lib/python3.7/site-packages/swiptapi.egg-link (link to .)
Removing swiptapi 0.0.1 from easy-install.pth file
```

```
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> remove.packages("swiptapi")
Removing package from ‘/Users/tbakanas/Library/R/3.6/library’
(as ‘lib’ is unspecified)
```

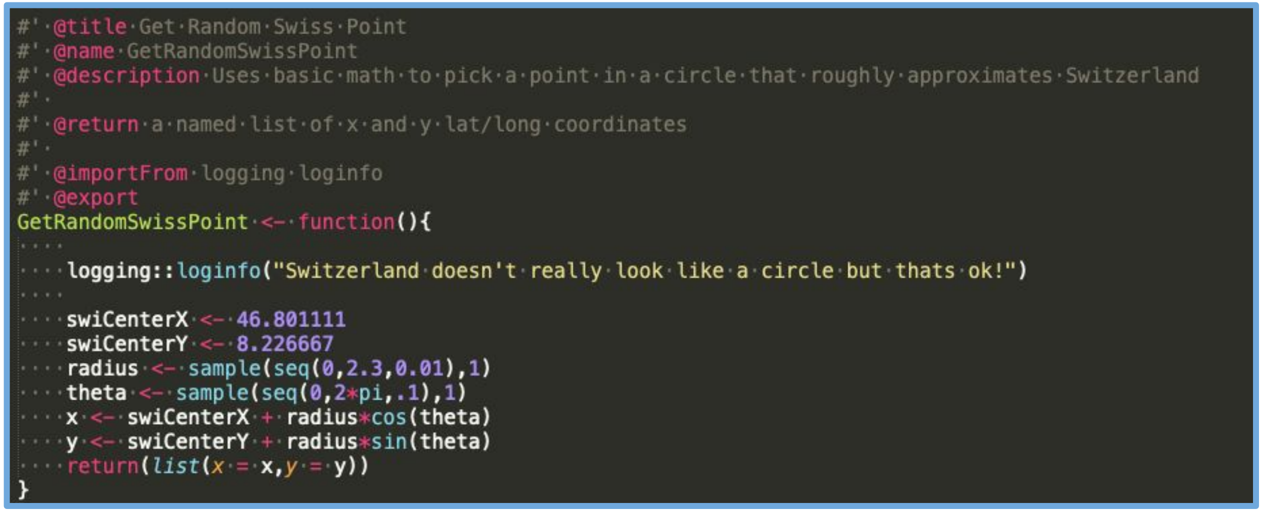
Functions



```
starterkits
  build
  scripts
  swiptapi-py
    swiptapi
      __pycache__
      __init__.py
      __version__.py
      functions.py
      swclient.py
    swiptapi.egg-info
    setup.py
  swiptapi-r
    man
    R
      client.R
      functions.R
    DESCRIPTION
    NAMESPACE
  .gitignore
  LICENSE
  README.md
```

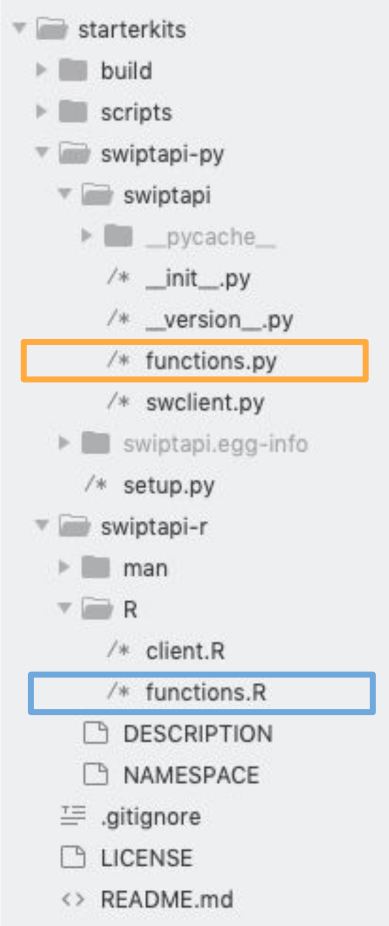


```python
def get_random_swiss_point():
    """
    Uses basic math to pick a point in a circle that roughly approximates Switzerland
    """

    logging.info("Switzerland doesn't really look like a circle but thats ok!")

    swi_center_x = 46.801111
    swi_center_y = 8.226667

    radius = sample(set(np.arange(0,2.3,0.01)),1)[0]
    theta = sample(set(np.arange(0,2*np.pi,0.01)),1)[0]
    x = swi_center_x + radius*np.cos(theta)
    y = swi_center_y + radius*np.sin(theta)

    return {"x": x, "y": y}
```

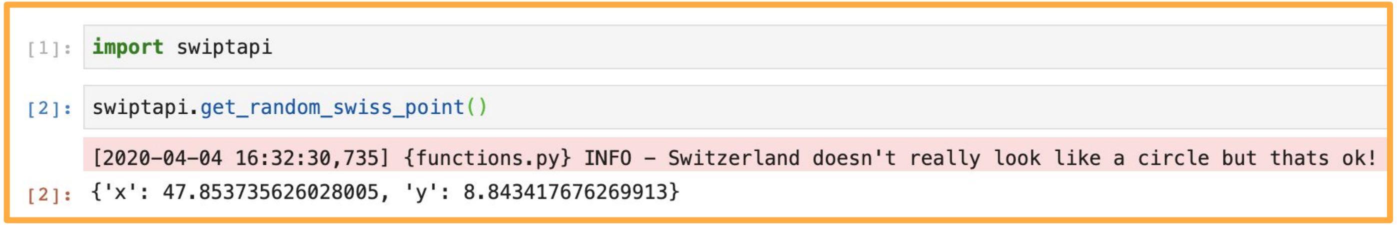


```r
#' @title Get Random Swiss Point
#' @name GetRandomSwissPoint
#' @description Uses basic math to pick a point in a circle that roughly approximates Switzerland
#'
#' @return a named list of x and y lat/long coordinates
#'
#' @importFrom logging loginfo
#' @export
GetRandomSwissPoint <- function(){

    logging::loginfo("Switzerland doesn't really look like a circle but thats ok!")

    swiCenterX <- 46.801111
    swiCenterY <- 8.226667
    radius <- sample(seq(0,2.3,0.01),1)
    theta <- sample(seq(0,2*pi,.1),1)
    x <- swiCenterX + radius*cos(theta)
    y <- swiCenterY + radius*sin(theta)
    return(list(x = x,y = y))
}
```

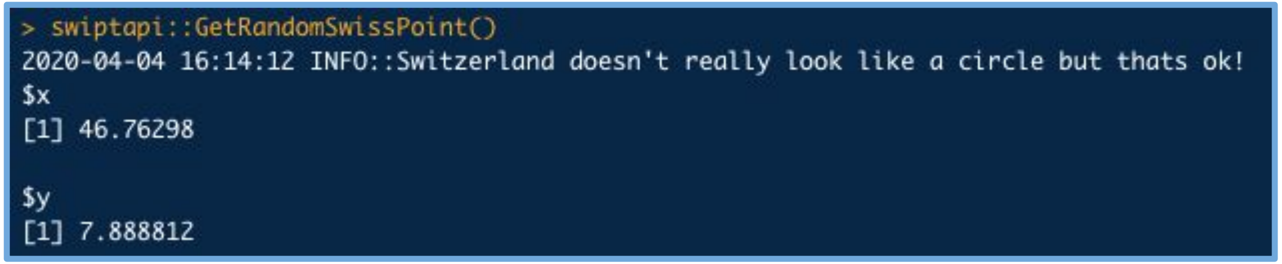
Functions



```
starterkits
  build
  scripts
  swiptapi-py
    swiptapi
      __pycache__
      __init__.py
      __version__.py
      functions.py
      swclient.py
    swiptapi.egg-info
    setup.py
  swiptapi-r
    man
    R
      client.R
      functions.R
    DESCRIPTION
    NAMESPACE
  .gitignore
  LICENSE
  README.md
```

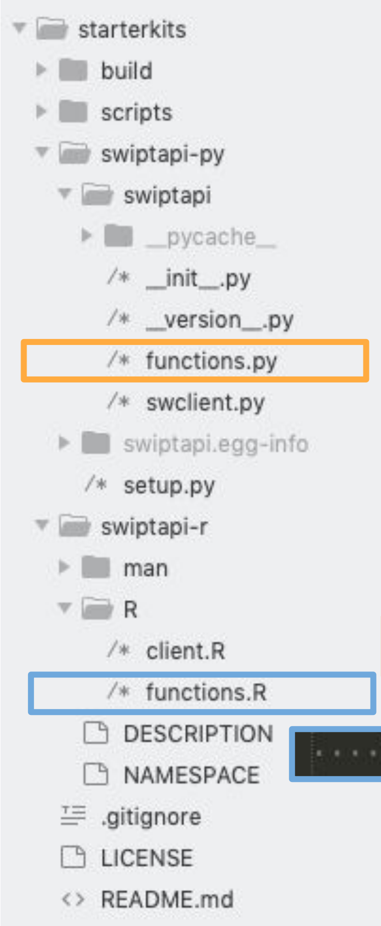


```
[1]: import swiptapi

[2]: swiptapi.get_random_swiss_point()

     [2020-04-04 16:32:30,735] {functions.py} INFO - Switzerland doesn't really look like a circle but thats ok!
[2]: {'x': 47.853735626028005, 'y': 8.843417676269913}
```



```
> swiptapi::GetRandomSwissPoint()
2020-04-04 16:14:12 INFO::Switzerland doesn't really look like a circle but thats ok!
$x
[1] 46.76298

$y
[1] 7.888812
```

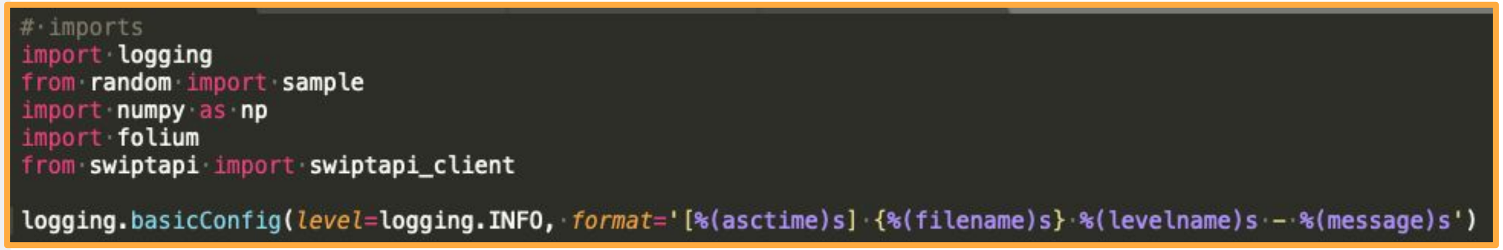
Logging



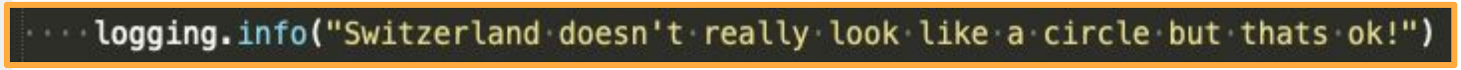
```
▼ starterkits
  ▶ build
  ▶ scripts
  ▼ swiptapi-py
    ▼ swiptapi
      ▶ __pycache__
        /* __init__.py
        /* __version__.py
        /* functions.py
        /* swclient.py
    ▶ swiptapi.egg-info
      /* setup.py
  ▼ swiptapi-r
    ▶ man
    ▼ R
        /* client.R
        /* functions.R
      DESCRIPTION
      NAMESPACE
  .gitignore
  LICENSE
  README.md
```

Adding logging to the package can be helpful to follow execution and for debugging

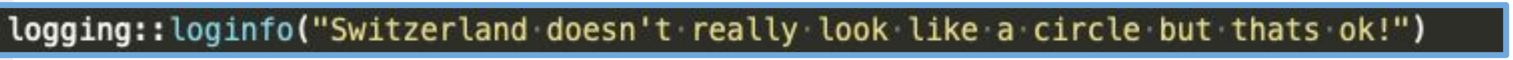
- there are many logging packages! (you could even get away with just “print()”)
- “logging” is a package in R and python
  - it is a base package in python
  - R: “A logging package emulating the Python logging package”
  - in python you should set the config, R you can just use directly



```python
# imports
import logging
from random import sample
import numpy as np
import folium
from swiptapi import swiptapi_client

logging.basicConfig(level=logging.INFO, format='[%(asctime)s] {%(filename)s} %(levelname)s - %(message)s')
```

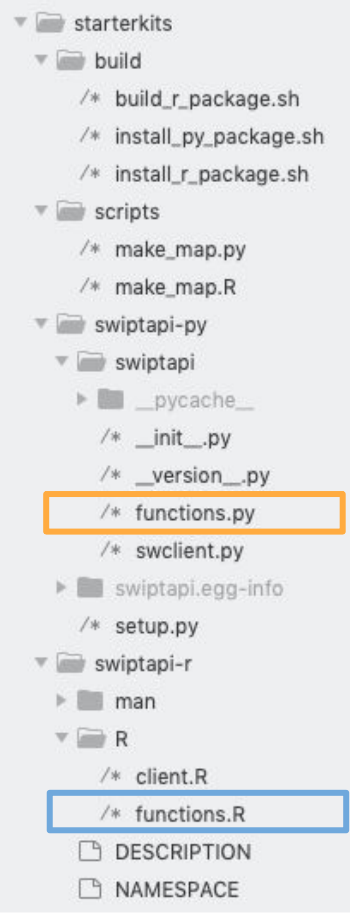


```python
    logging.info("Switzerland doesn't really look like a circle but thats ok!")
```

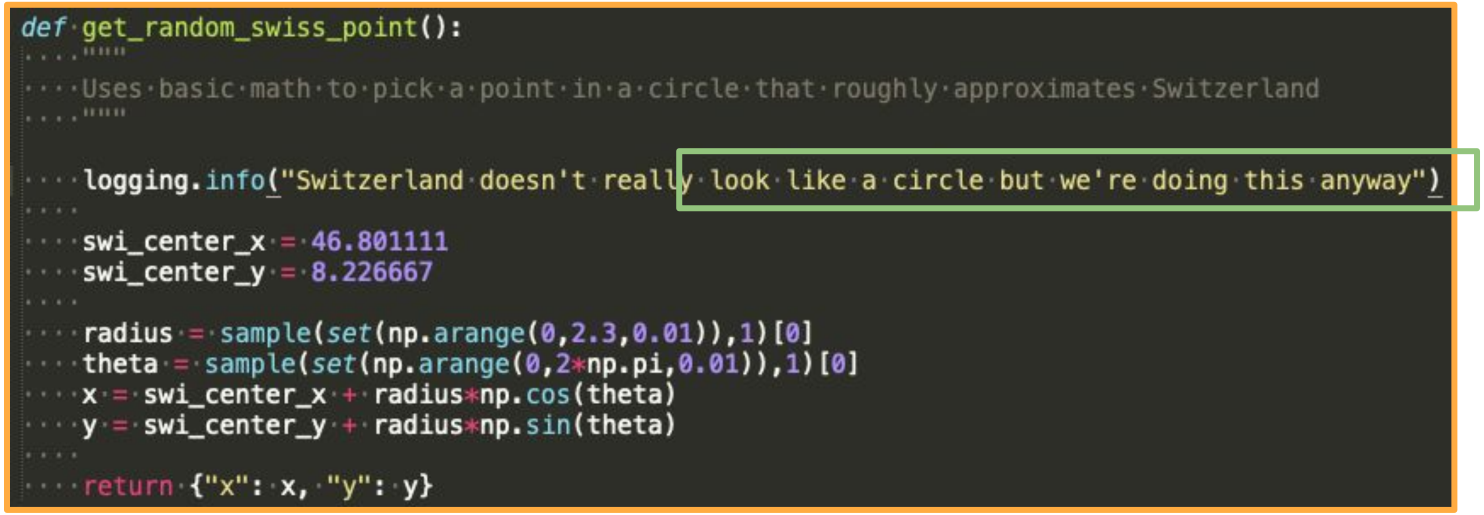


```r
    logging::loginfo("Switzerland doesn't really look like a circle but thats ok!")
```

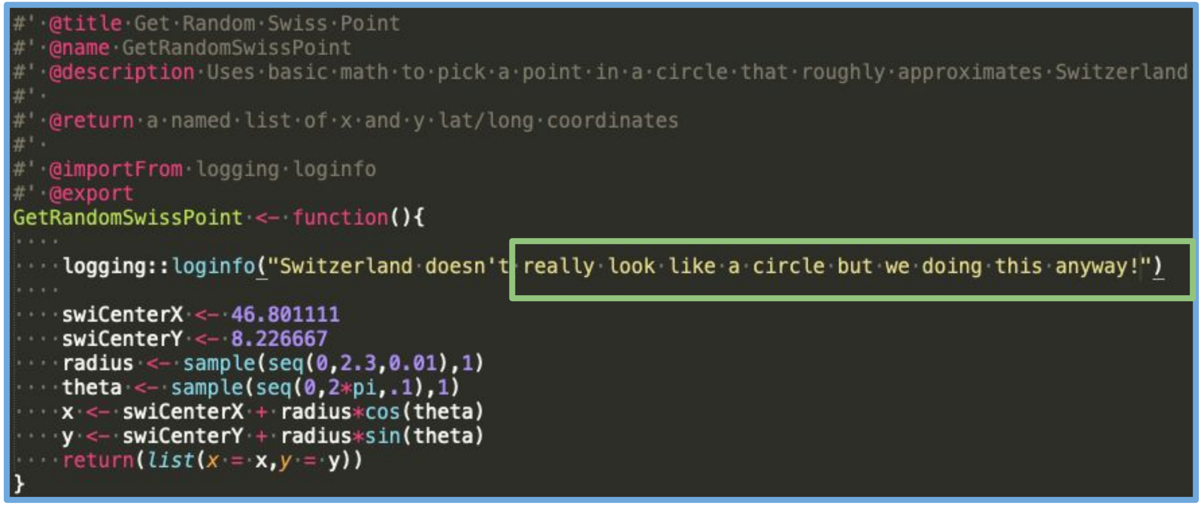
Iteration



```
▾ starterkits
  ▾ build
      /* build_r_package.sh
      /* install_py_package.sh
      /* install_r_package.sh
  ▾ scripts
      /* make_map.py
      /* make_map.R
  ▾ swiptapi-py
    ▾ swiptapi
      ▸ __pycache__
        /* __init__.py
        /* __version__.py
        /* functions.py
        /* swclient.py
    ▸ swiptapi.egg-info
      /* setup.py
  ▾ swiptapi-r
    ▸ man
    ▾ R
        /* client.R
        /* functions.R
      DESCRIPTION
      NAMESPACE
```



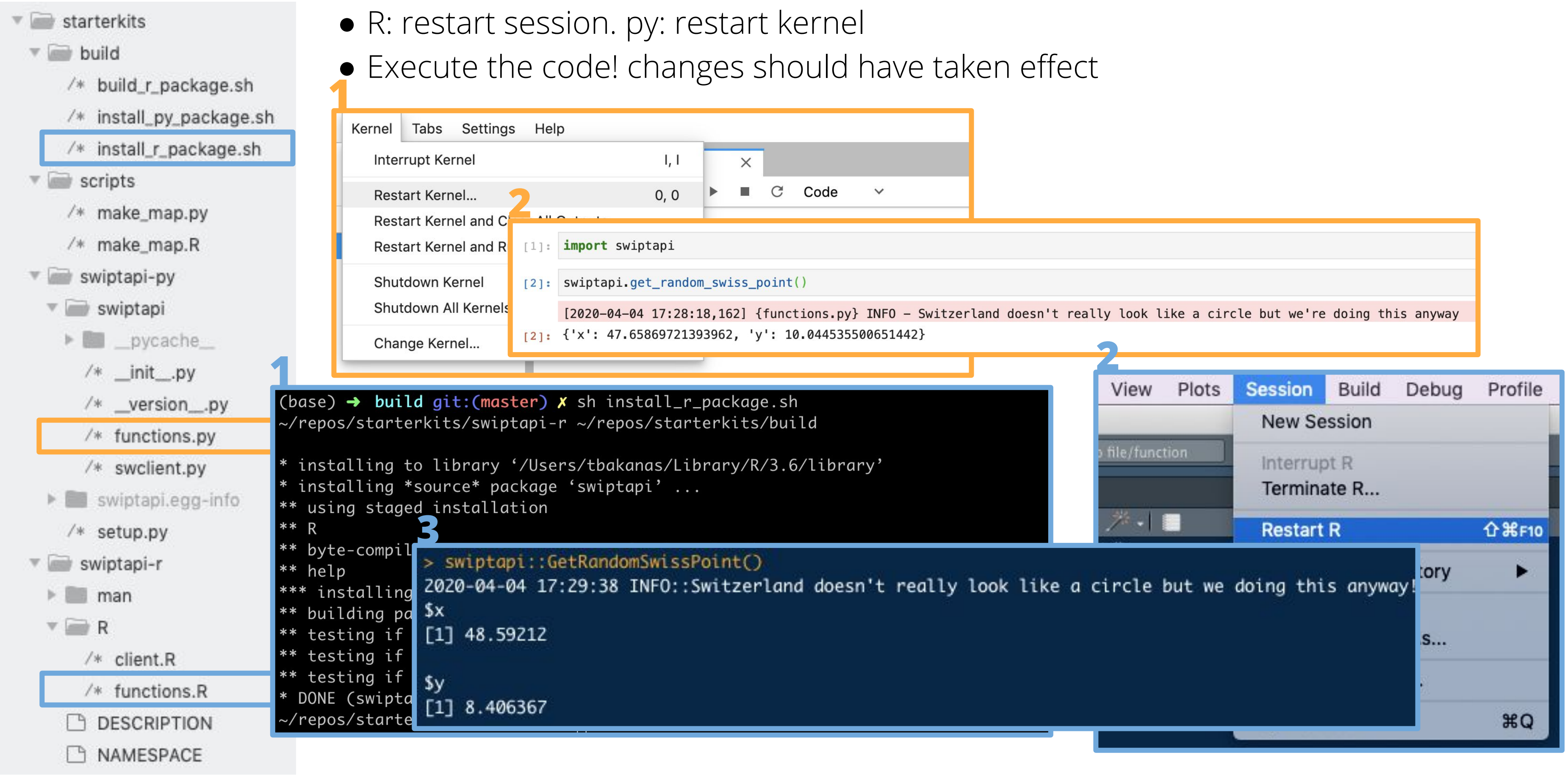
```python
def get_random_swiss_point():
    """
    Uses basic math to pick a point in a circle that roughly approximates Switzerland
    """

    logging.info("Switzerland doesn't really look like a circle but we're doing this anyway")

    swi_center_x = 46.801111
    swi_center_y = 8.226667

    radius = sample(set(np.arange(0,2.3,0.01)),1)[0]
    theta = sample(set(np.arange(0,2*np.pi,0.01)),1)[0]
    x = swi_center_x + radius*np.cos(theta)
    y = swi_center_y + radius*np.sin(theta)

    return {"x": x, "y": y}
```



```r
#' @title Get Random Swiss Point
#' @name GetRandomSwissPoint
#' @description Uses basic math to pick a point in a circle that roughly approximates Switzerland
#'
#' @return a named list of x and y lat/long coordinates
#'
#' @importFrom logging loginfo
#' @export
GetRandomSwissPoint <- function(){

    logging::loginfo("Switzerland doesn't really look like a circle but we doing this anyway!")

    swiCenterX <- 46.801111
    swiCenterY <- 8.226667
    radius <- sample(seq(0,2.3,0.01),1)
    theta <- sample(seq(0,2*pi,.1),1)
    x <- swiCenterX + radius*cos(theta)
    y <- swiCenterY + radius*sin(theta)
    return(list(x = x,y = y))
}
```

Iteration

- Save the file
- R: install package again. py: if installed with 'develop' flag, then reinstall unnecessary
- R: restart session. py: restart kernel
- Execute the code! changes should have taken effect



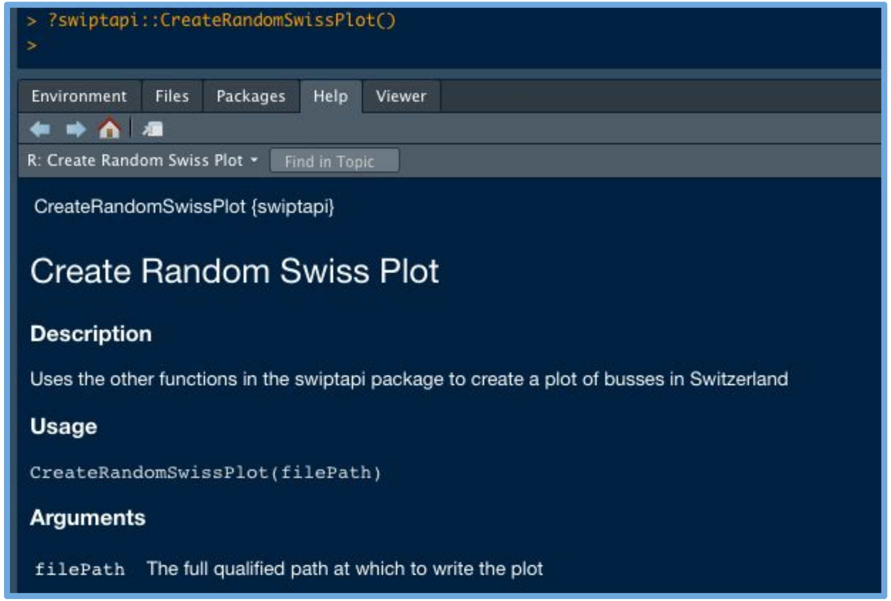
```
[1]: import swiptapi

[2]: swiptapi.get_random_swiss_point()

     [2020-04-04 17:28:18,162] {functions.py} INFO - Switzerland doesn't really look like a circle but we're doing this anyway

[2]: {'x': 47.65869721393962, 'y': 10.044535500651442}
```

```
(base) ➜  build git:(master) ✗ sh install_r_package.sh
~/repos/starterkits/swiptapi-r ~/repos/starterkits/build

* installing to library '/Users/tbakanas/Library/R/3.6/library'
* installing *source* package 'swiptapi' ...
** using staged installation
** R
** byte-compil
** help
*** installing
** building pa
** testing if
** testing if
** testing if
* DONE (swipta
~/repos/starte
```

```
> swiptapi::GetRandomSwissPoint()
2020-04-04 17:29:38 INFO::Switzerland doesn't really look like a circle but we doing this anyway!
$x
[1] 48.59212

$y
[1] 8.406367
```

Documentation

- providing documentation around a package is important to ensure it is usable
- there are many ways to go about documentation! I will touch on some simpler ones
- R: roxygen (important because it also helps build the package)
- py: docstrings



```
> ?swiptapi::CreateRandomSwissPlot()
>
```

CreateRandomSwissPlot {swiptapi}

Create Random Swiss Plot

Description

Uses the other functions in the swiptapi package to create a plot of busses in Switzerland

Usage

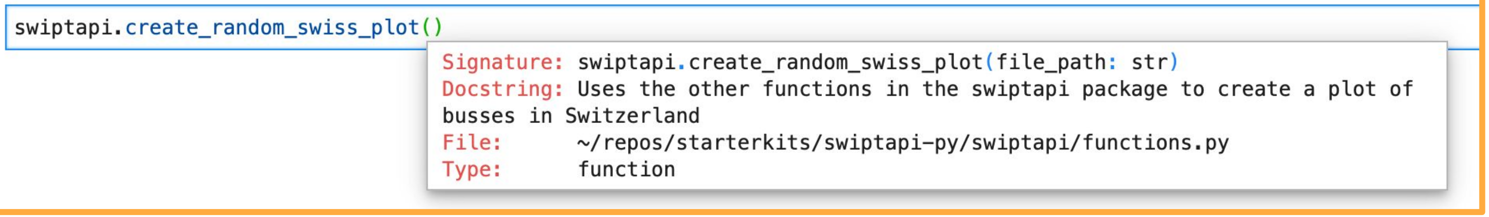
```
CreateRandomSwissPlot(filePath)
```

Arguments

`filePath` The full qualified path at which to write the plot

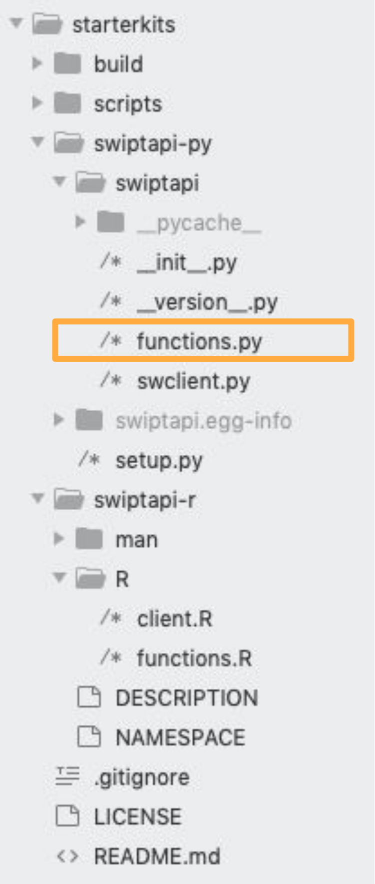
R: you can use ?myFunctionName to pull up the roxygen docs

py: you can use shift+tab to pull up the docstring and signature or help()

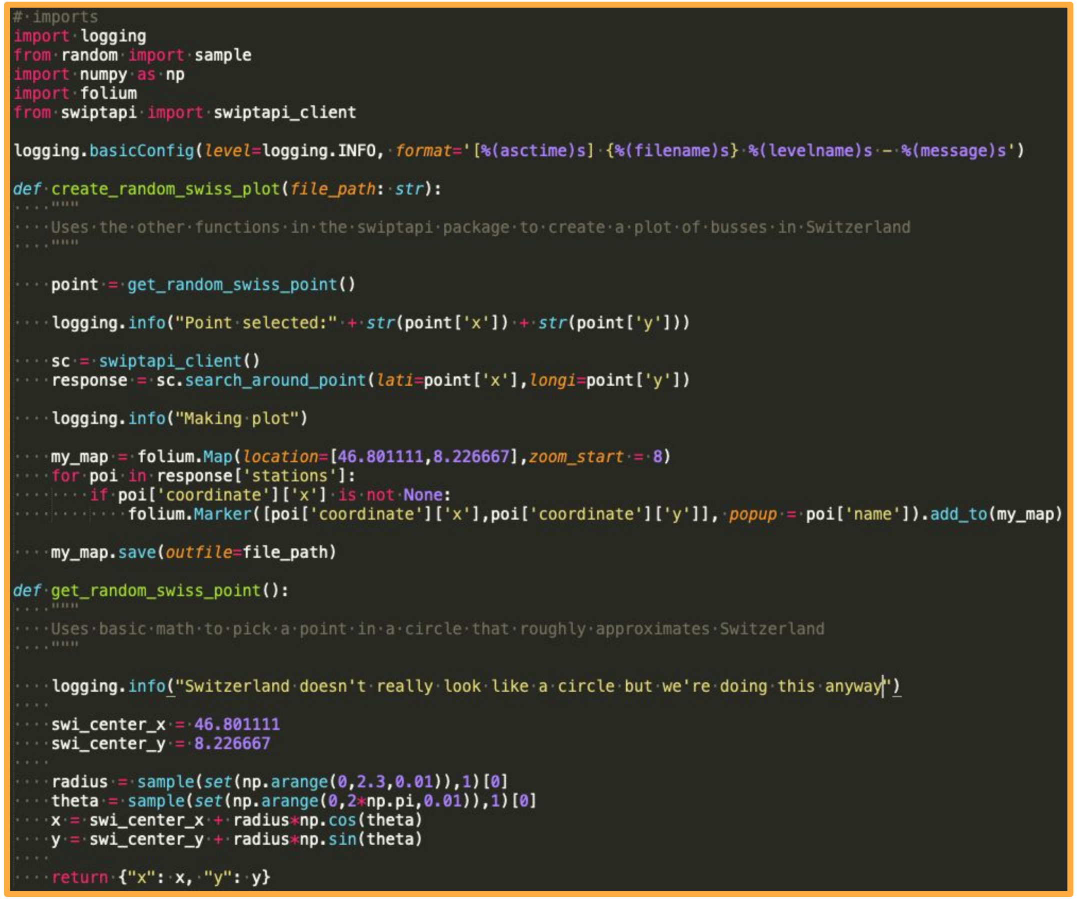


```
swiptapi.create_random_swiss_plot()
Signature: swiptapi.create_random_swiss_plot(file_path: str)
Docstring: Uses the other functions in the swiptapi package to create a plot of
busses in Switzerland
File:      ~/repos/starterkits/swiptapi-py/swiptapi/functions.py
Type:      function
```

Doc Strings - Python

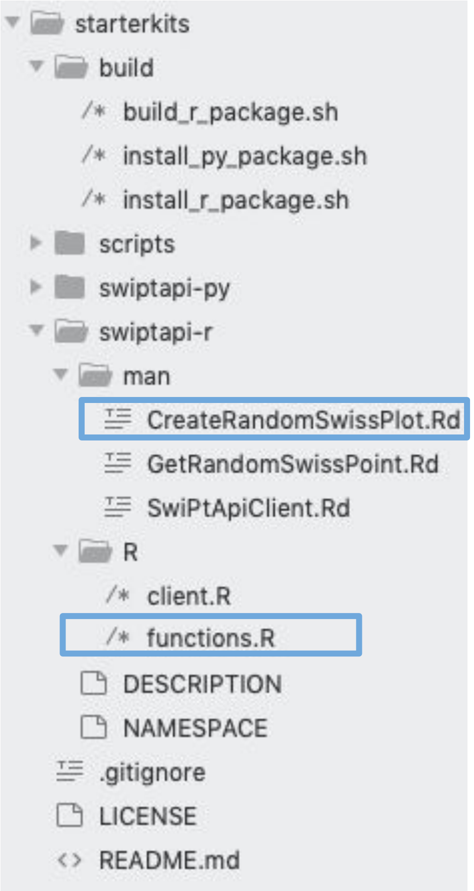


```
starterkits
  build
  scripts
  swiptapi-py
    swiptapi
      __pycache__
      __init__.py
      __version__.py
      functions.py
      swclient.py
    swiptapi.egg-info
    setup.py
  swiptapi-r
    man
    R
      client.R
      functions.R
    DESCRIPTION
    NAMESPACE
  .gitignore
  LICENSE
  README.md
```

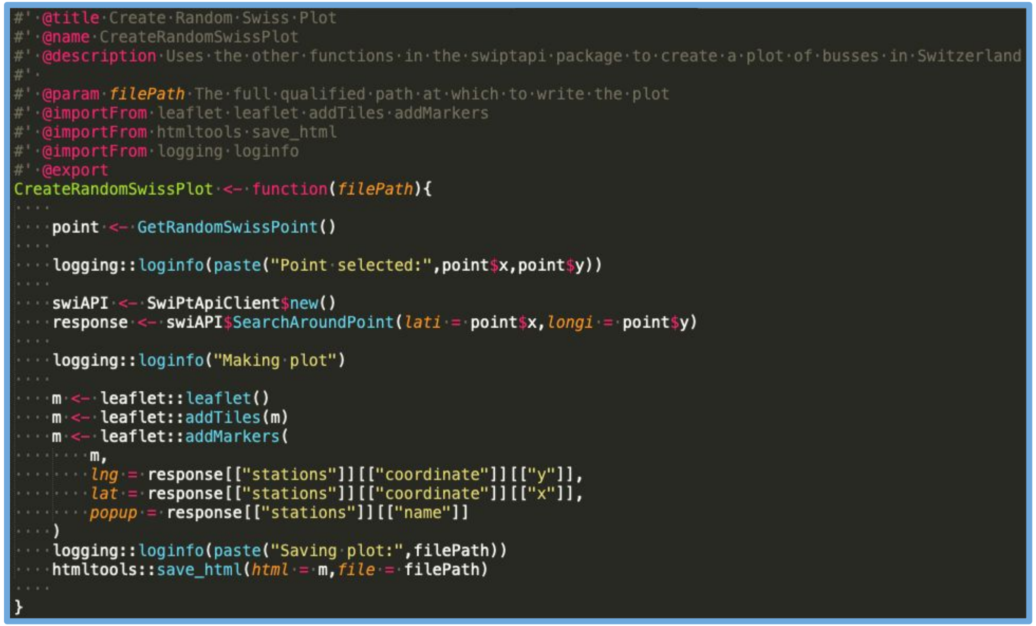


```python
# imports
import logging
from random import sample
import numpy as np
import folium
from swiptapi import swiptapi_client

logging.basicConfig(level=logging.INFO, format='[%(asctime)s] {%(filename)s} %(levelname)s - %(message)s')

def create_random_swiss_plot(file_path: str):
    """
    Uses the other functions in the swiptapi package to create a plot of busses in Switzerland
    """

    point = get_random_swiss_point()

    logging.info("Point selected:" + str(point['x']) + str(point['y']))

    sc = swiptapi_client()
    response = sc.search_around_point(lati=point['x'],longi=point['y'])

    logging.info("Making plot")

    my_map = folium.Map(location=[46.801111,8.226667],zoom_start = 8)
    for poi in response['stations']:
        if poi['coordinate']['x'] is not None:
            folium.Marker([poi['coordinate']['x'],poi['coordinate']['y']], popup = poi['name']).add_to(my_map)

    my_map.save(outfile=file_path)

def get_random_swiss_point():
    """
    Uses basic math to pick a point in a circle that roughly approximates Switzerland
    """

    logging.info("Switzerland doesn't really look like a circle but we're doing this anyway")

    swi_center_x = 46.801111
    swi_center_y = 8.226667

    radius = sample(set(np.arange(0,2.3,0.01)),1)[0]
    theta = sample(set(np.arange(0,2*np.pi,0.01)),1)[0]
    x = swi_center_x + radius*np.cos(theta)
    y = swi_center_y + radius*np.sin(theta)

    return {"x": x, "y": y}
```

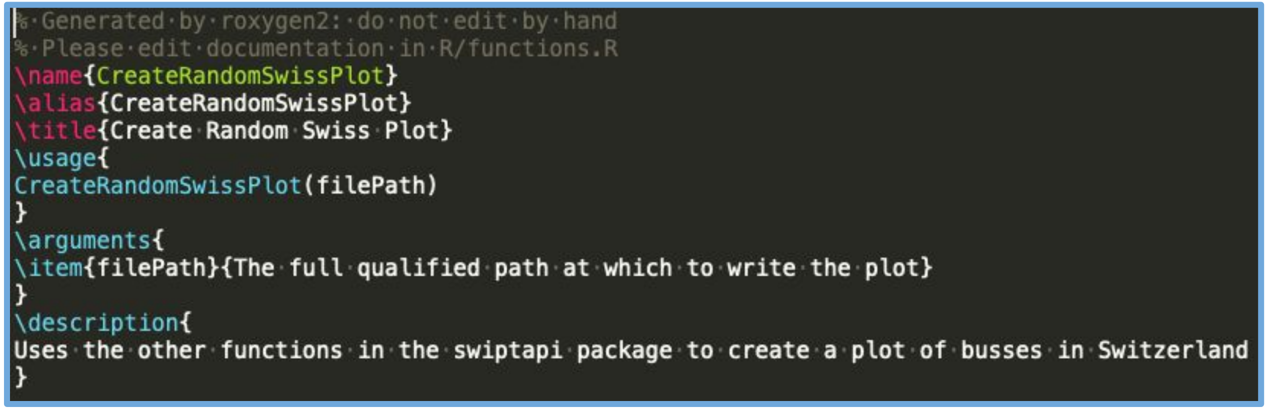
roxygen - R



- starterkits
  - build
    - build_r_package.sh
    - install_py_package.sh
    - install_r_package.sh
  - scripts
  - swiptapi-py
  - swiptapi-r
    - man
      - CreateRandomSwissPlot.Rd
      - GetRandomSwissPoint.Rd
      - SwiPtApiClient.Rd
    - R
      - client.R
      - functions.R
    - DESCRIPTION
    - NAMESPACE
  - .gitignore
  - LICENSE
  - README.md

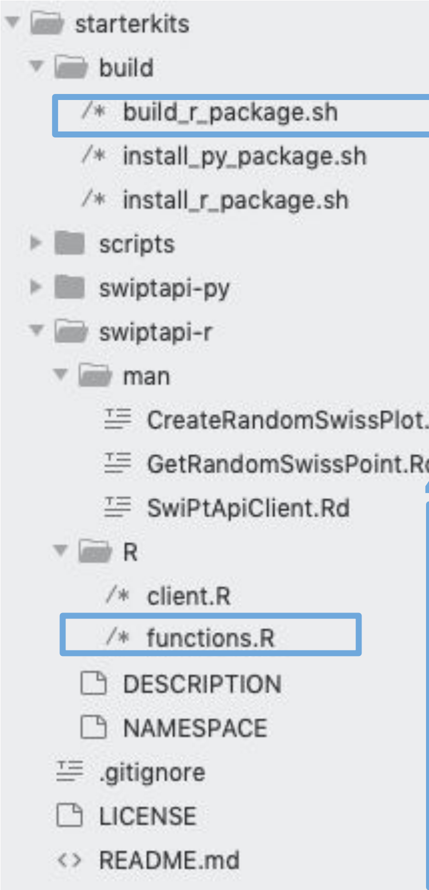


```r
#' @title Create Random Swiss Plot
#' @name CreateRandomSwissPlot
#' @description Uses the other functions in the swiptapi package to create a plot of busses in Switzerland
#' 
#' @param filePath The full qualified path at which to write the plot
#' @importFrom leaflet leaflet addTiles addMarkers
#' @importFrom htmltools save_html
#' @importFrom logging loginfo
#' @export
CreateRandomSwissPlot <- function(filePath){
    
    point <- GetRandomSwissPoint()
    
    logging::loginfo(paste("Point selected:",point$x,point$y))
    
    swiAPI <- SwiPtApiClient$new()
    response <- swiAPI$SearchAroundPoint(lati = point$x, longi = point$y)
    
    logging::loginfo("Making plot")
    
    m <- leaflet::leaflet()
    m <- leaflet::addTiles(m)
    m <- leaflet::addMarkers(
        m,
        lng = response[["stations"]][["coordinate"]][["y"]],
        lat = response[["stations"]][["coordinate"]][["x"]],
        popup = response[["stations"]][["name"]]
    )
    logging::loginfo(paste("Saving plot:",filePath))
    htmltools::save_html(html = m, file = filePath)
    
}
```



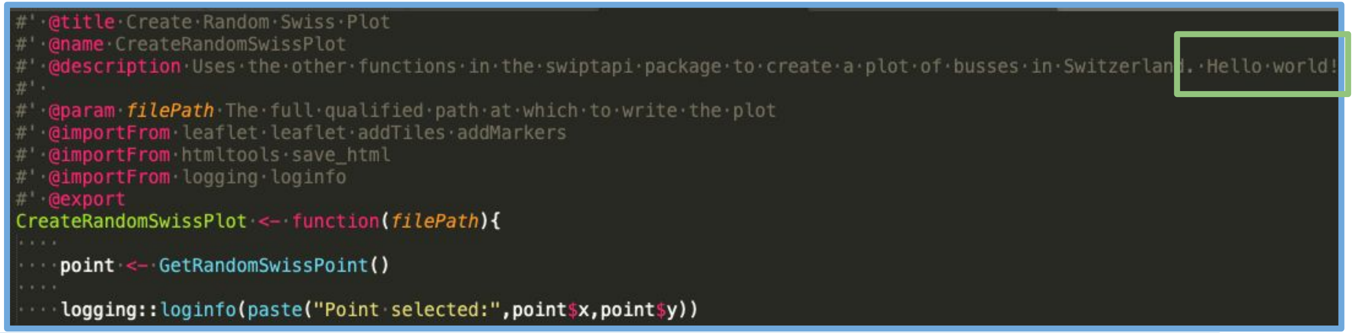
```
% Generated by roxygen2: do not edit by hand
% Please edit documentation in R/functions.R
\name{CreateRandomSwissPlot}
\alias{CreateRandomSwissPlot}
\title{Create Random Swiss Plot}
\usage{
CreateRandomSwissPlot(filePath)
}
\arguments{
\item{filePath}{The full qualified path at which to write the plot}
}
\description{
Uses the other functions in the swiptapi package to create a plot of busses in Switzerland
}
```

roxygen - R



```
▼ starterkits
  ▼ build
      build_r_package.sh
      install_py_package.sh
      install_r_package.sh
  ▶ scripts
  ▶ swiptapi-py
  ▼ swiptapi-r
    ▼ man
        CreateRandomSwissPlot.Rd
        GetRandomSwissPoint.Rd
        SwiPtApiClient.Rd
    ▼ R
        client.R
        functions.R
      DESCRIPTION
      NAMESPACE
    .gitignore
    LICENSE
    README.md
```

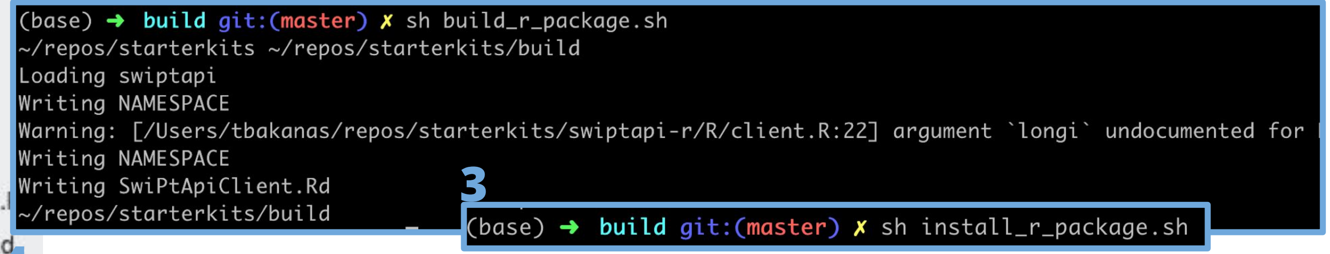
1



```r
#' @title Create Random Swiss Plot
#' @name CreateRandomSwissPlot
#' @description Uses the other functions in the swiptapi package to create a plot of busses in Switzerland. Hello world!
#'
#' @param filePath The full qualified path at which to write the plot
#' @importFrom leaflet leaflet addTiles addMarkers
#' @importFrom htmltools save_html
#' @importFrom logging loginfo
#' @export
CreateRandomSwissPlot <- function(filePath){

    point <- GetRandomSwissPoint()

    logging::loginfo(paste("Point selected:",point$x,point$y))
```

2

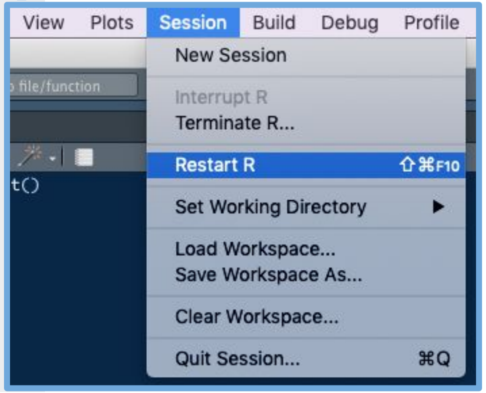
3



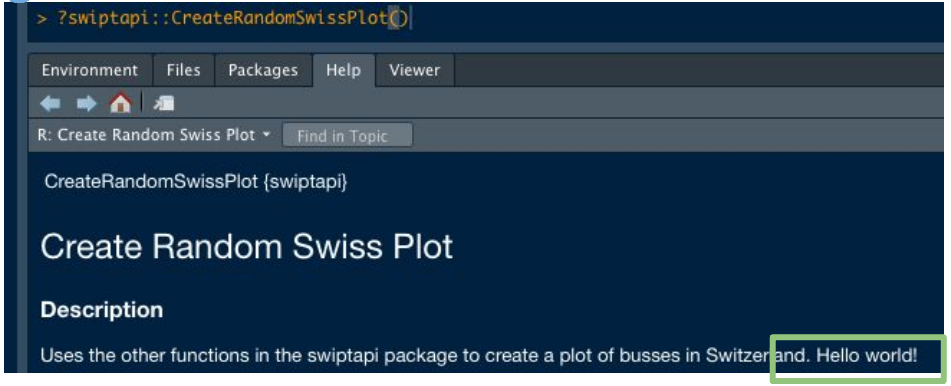
```
(base) ➜  build git:(master) ✗ sh build_r_package.sh
~/repos/starterkits ~/repos/starterkits/build
Loading swiptapi
Writing NAMESPACE
Warning: [/Users/tbakanas/repos/starterkits/swiptapi-r/R/client.R:22] argument `longi` undocumented for
Writing NAMESPACE
Writing SwiPtApiClient.Rd
~/repos/starterkits/build
```

```
(base) ➜  build git:(master) ✗ sh install_r_package.sh
```

4



5



```
> ?swiptapi::CreateRandomSwissPlot()
```

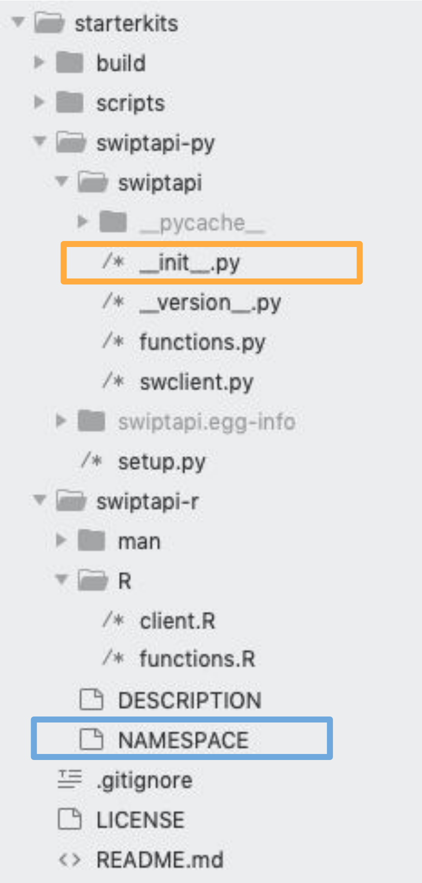
CreateRandomSwissPlot {swiptapi}

Create Random Swiss Plot

Description

Uses the other functions in the swiptapi package to create a plot of busses in Switzerland. Hello world!

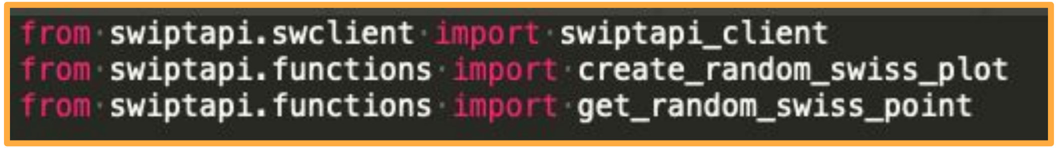
Getting your code to show up!



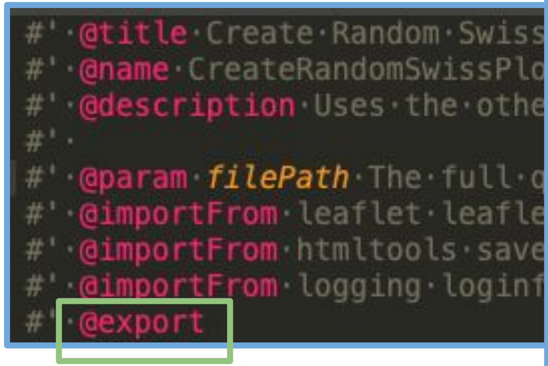
```
starterkits
  build
  scripts
  swiptapi-py
    swiptapi
      __pycache__
      __init__.py
      __version__.py
      functions.py
      swclient.py
    swiptapi.egg-info
    setup.py
  swiptapi-r
    man
    R
      client.R
      functions.R
    DESCRIPTION
    NAMESPACE
  .gitignore
  LICENSE
  README.md
```

Getting a function or object to show up in a package can be deceptively tricky!

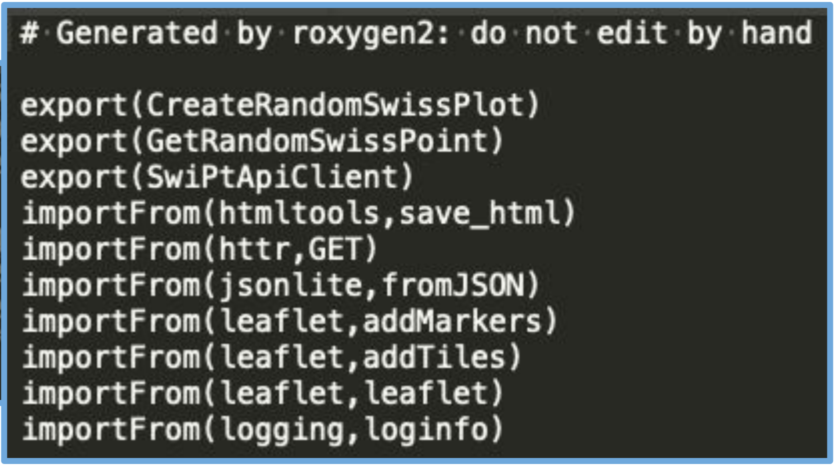
- R: roxygen helps with this.
  - when tagged with the @export roxygen will write the function into the NAMESPACE
- py: all about the "__init__.py" files
  - the folder "swiptapi" is a module
  - the from … import … statements effectively push functions up a level



```
from swiptapi.swclient import swiptapi_client
from swiptapi.functions import create_random_swiss_plot
from swiptapi.functions import get_random_swiss_point
```

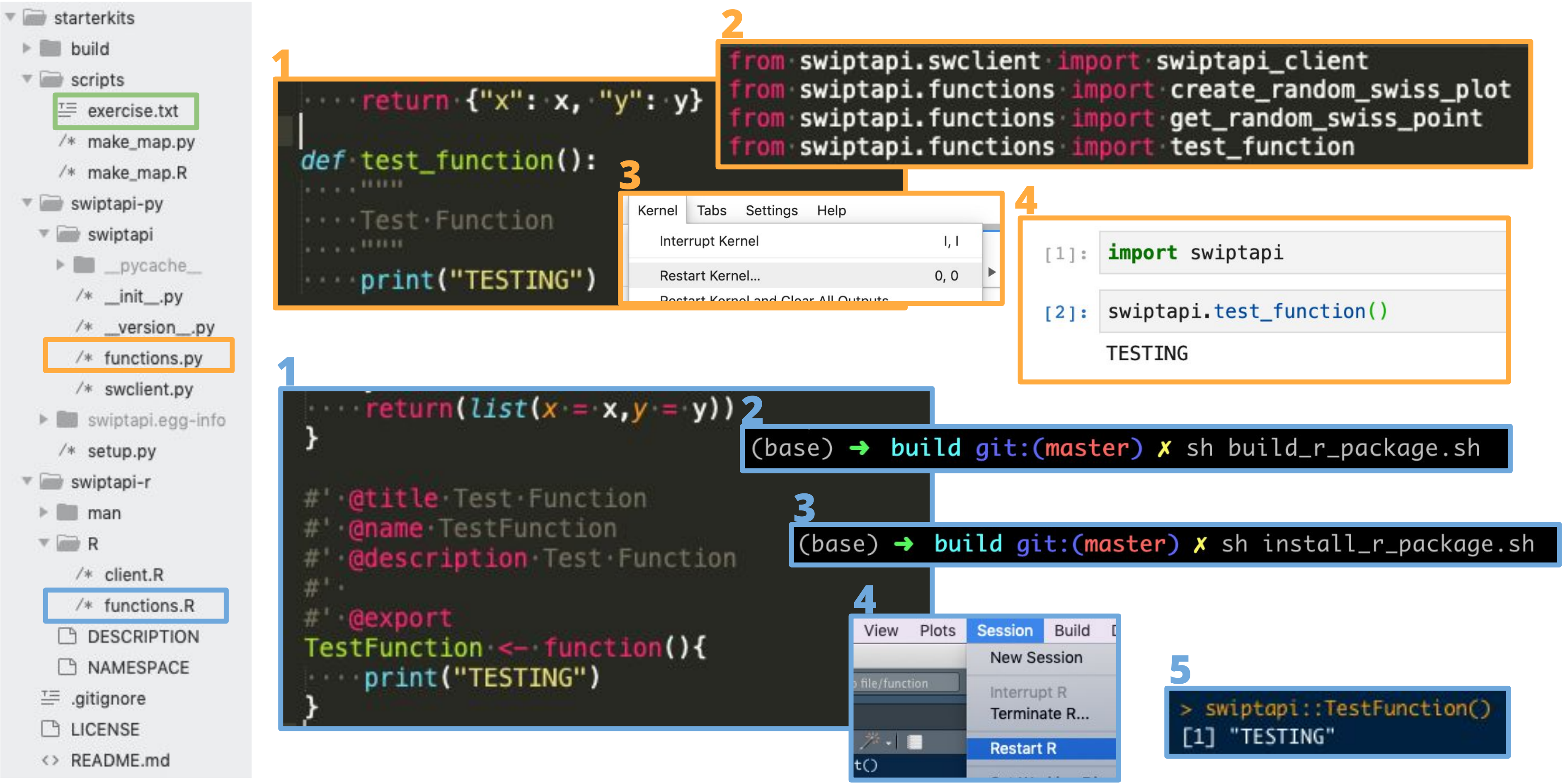


```
#' @title Create Random Swiss
#' @name CreateRandomSwissPlo
#' @description Uses the othe
#'
#' @param filePath The full q
#' @importFrom leaflet leafle
#' @importFrom htmltools save
#' @importFrom logging loginf
#' @export
```



```
# Generated by roxygen2: do not edit by hand

export(CreateRandomSwissPlot)
export(GetRandomSwissPoint)
export(SwiPtApiClient)
importFrom(htmltools,save_html)
importFrom(httr,GET)
importFrom(jsonlite,fromJSON)
importFrom(leaflet,addMarkers)
importFrom(leaflet,addTiles)
importFrom(leaflet,leaflet)
importFrom(logging,loginfo)
```

Getting your code to show up!



1

```
    return {"x": x, "y": y}

def test_function():
    """
    Test Function
    """
    print("TESTING")
```

2

```
from swiptapi.swclient import swiptapi_client
from swiptapi.functions import create_random_swiss_plot
from swiptapi.functions import get_random_swiss_point
from swiptapi.functions import test_function
```

4

```
[1]: import swiptapi

[2]: swiptapi.test_function()
     TESTING
```

1

```
    return(list(x = x,y = y))
}

#' @title Test Function
#' @name TestFunction
#' @description Test Function
#'
#' @export
TestFunction <- function(){
    print("TESTING")
}
```

2

```
(base) ➜  build git:(master) ✗ sh build_r_package.sh
```

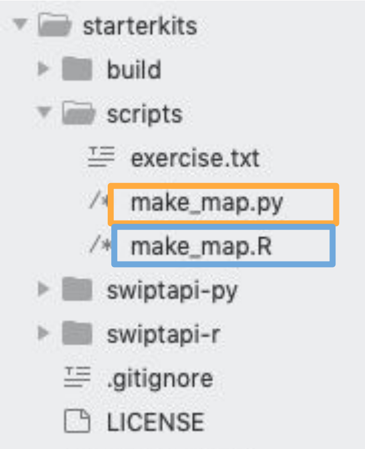
3

```
(base) ➜  build git:(master) ✗ sh install_r_package.sh
```

5

```
> swiptapi::TestFunction()
[1] "TESTING"
```

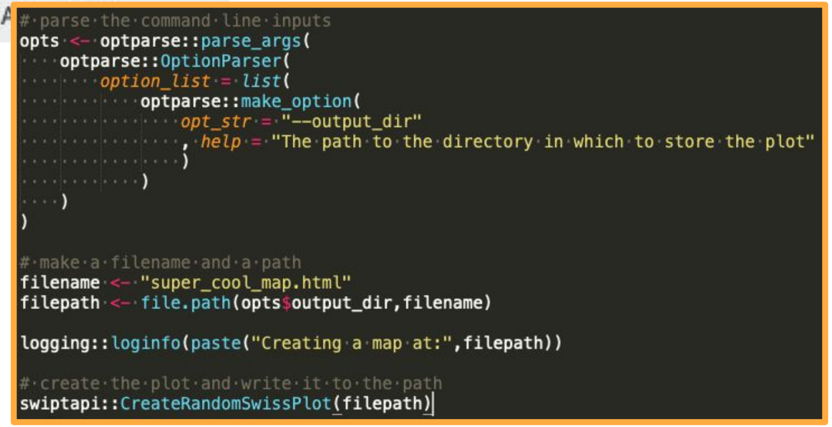
Making a command line executable script



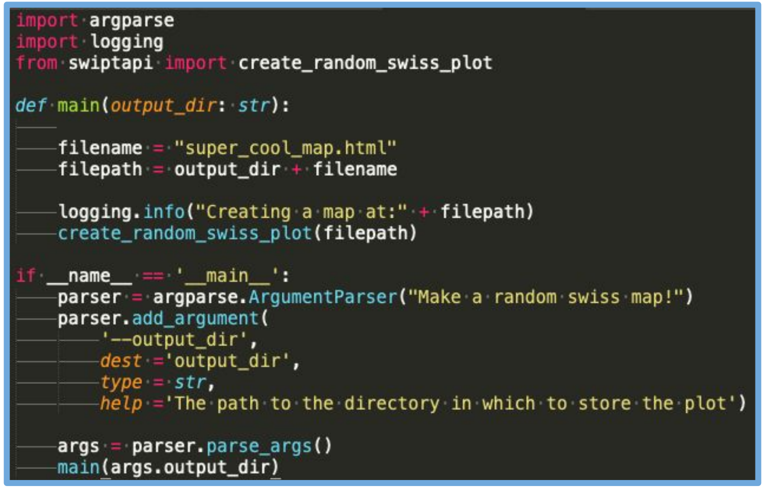
starterkits
build
scripts
exercise.txt
make_map.py
make_map.R
swiptapi-py
swiptapi-r
.gitignore
LICENSE
REA

While technically not package development understanding how to make a R or python script accept command line arguments is a useful tool

- can be used to invoke scripts without having to open RStudio or Jupyter
- allows programs to interact easily with schedulers like CRON or airflow
- R: code executes in order. Put the opts at the top.
- py: usually follows the __main__ convention seen below with the argument parsing taking place after the if statement

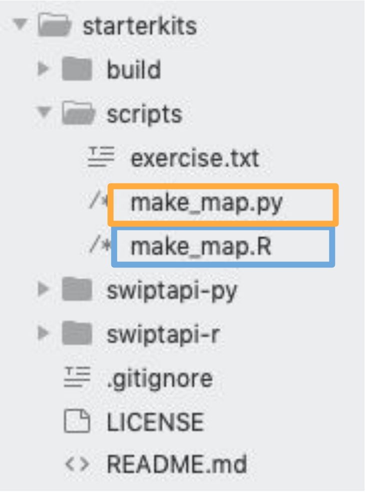


```r
# parse the command line inputs
opts <- optparse::parse_args(
    optparse::OptionParser(
        option_list = list(
            optparse::make_option(
                opt_str = "--output_dir"
                , help = "The path to the directory in which to store the plot"
                )
            )
    )
)

# make a filename and a path
filename <- "super_cool_map.html"
filepath <- file.path(opts$output_dir,filename)

logging::loginfo(paste("Creating a map at:",filepath))

# create the plot and write it to the path
swiptapi::CreateRandomSwissPlot(filepath)
```

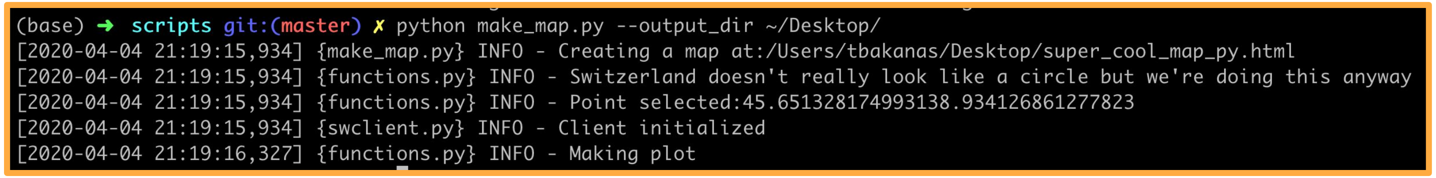


```python
import argparse
import logging
from swiptapi import create_random_swiss_plot

def main(output_dir: str):

    filename = "super_cool_map.html"
    filepath = output_dir + filename

    logging.info("Creating a map at:" + filepath)
    create_random_swiss_plot(filepath)

if __name__ == '__main__':
    parser = argparse.ArgumentParser("Make a random swiss map!")
    parser.add_argument(
        '--output_dir',
        dest ='output_dir',
        type = str,
        help ='The path to the directory in which to store the plot')

    args = parser.parse_args()
    main(args.output_dir)
```

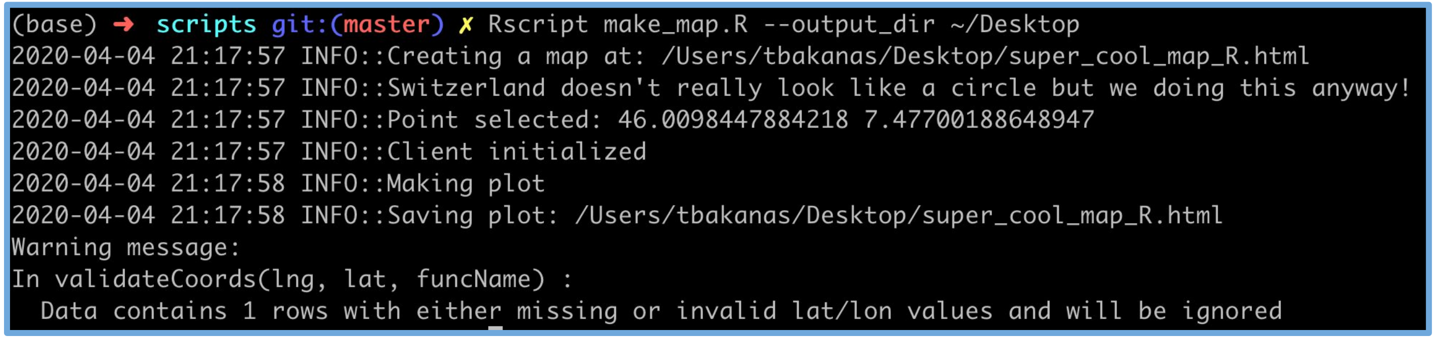
Making a command line executable script



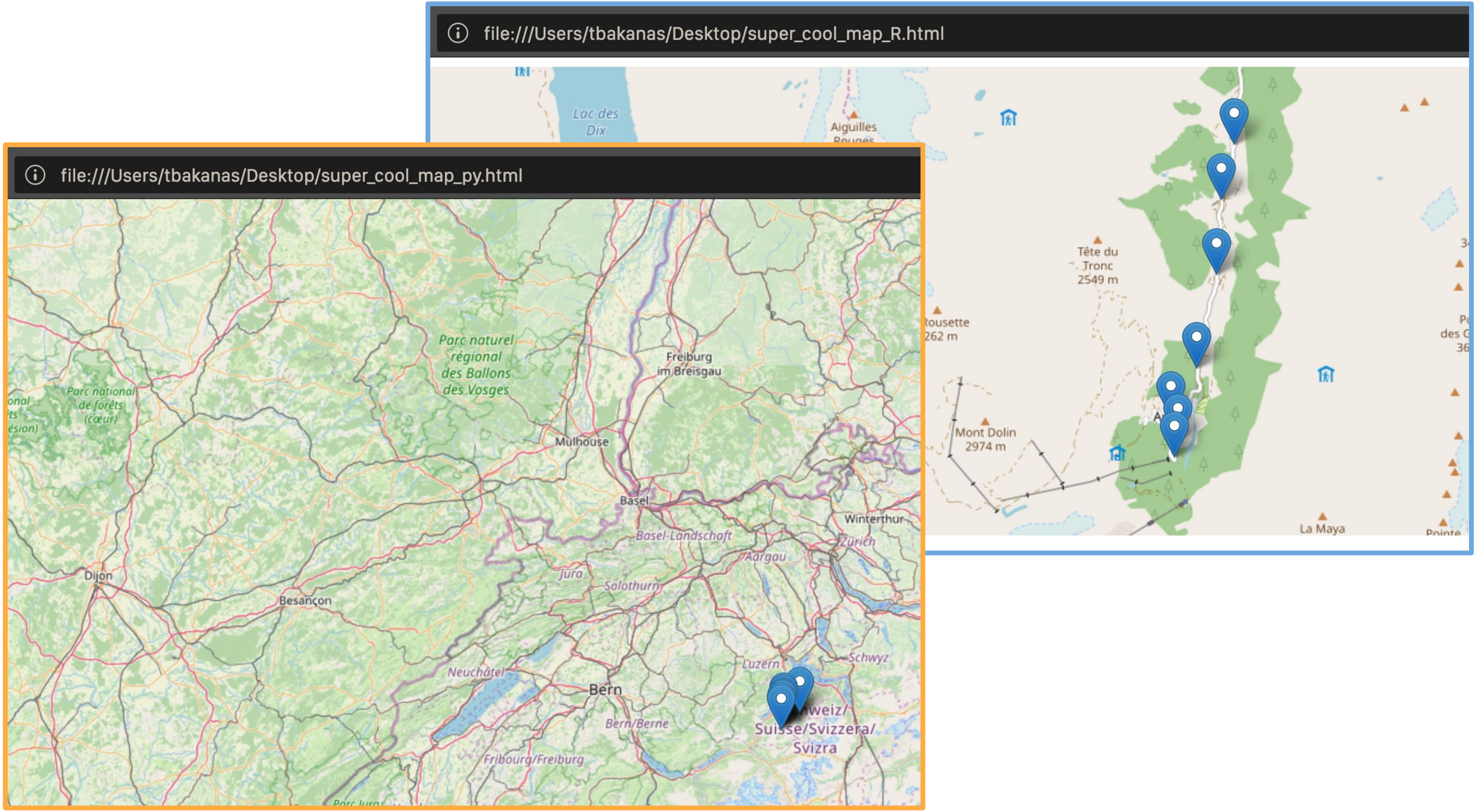
```
starterkits
  build
  scripts
    exercise.txt
    make_map.py
    make_map.R
  swiptapi-py
  swiptapi-r
  .gitignore
  LICENSE
  README.md
```



```
(base) ➜  scripts git:(master) ✗ python make_map.py --output_dir ~/Desktop/
[2020-04-04 21:19:15,934] {make_map.py} INFO - Creating a map at:/Users/tbakanas/Desktop/super_cool_map_py.html
[2020-04-04 21:19:15,934] {functions.py} INFO - Switzerland doesn't really look like a circle but we're doing this anyway
[2020-04-04 21:19:15,934] {functions.py} INFO - Point selected:45.651328174993138.934126861277823
[2020-04-04 21:19:15,934] {swclient.py} INFO - Client initialized
[2020-04-04 21:19:16,327] {functions.py} INFO - Making plot
```



```
(base) ➜  scripts git:(master) ✗ Rscript make_map.R --output_dir ~/Desktop
2020-04-04 21:17:57 INFO::Creating a map at: /Users/tbakanas/Desktop/super_cool_map_R.html
2020-04-04 21:17:57 INFO::Switzerland doesn't really look like a circle but we doing this anyway!
2020-04-04 21:17:57 INFO::Point selected: 46.0098447884218 7.47700188648947
2020-04-04 21:17:57 INFO::Client initialized
2020-04-04 21:17:58 INFO::Making plot
2020-04-04 21:17:58 INFO::Saving plot: /Users/tbakanas/Desktop/super_cool_map_R.html
Warning message:
In validateCoords(lng, lat, funcName) :
  Data contains 1 rows with either missing or invalid lat/lon values and will be ignored
```



file:///Users/tbakanas/Desktop/super_cool_map_R.html

file:///Users/tbakanas/Desktop/super_cool_map_py.html

Thank you

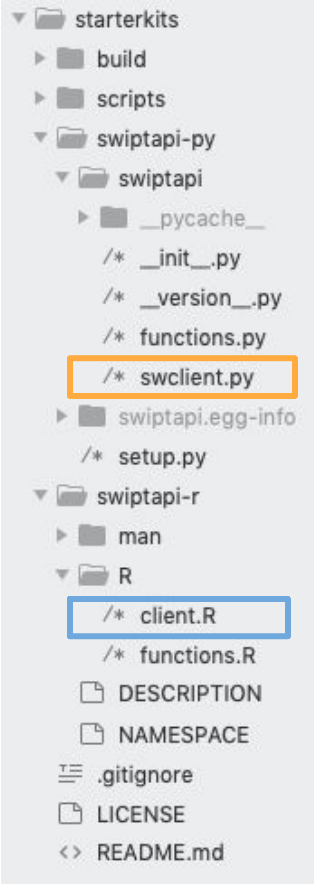
Feel free to reach out to me with any feedback, questions, or just to say hello

tedbakanas@gmail.com

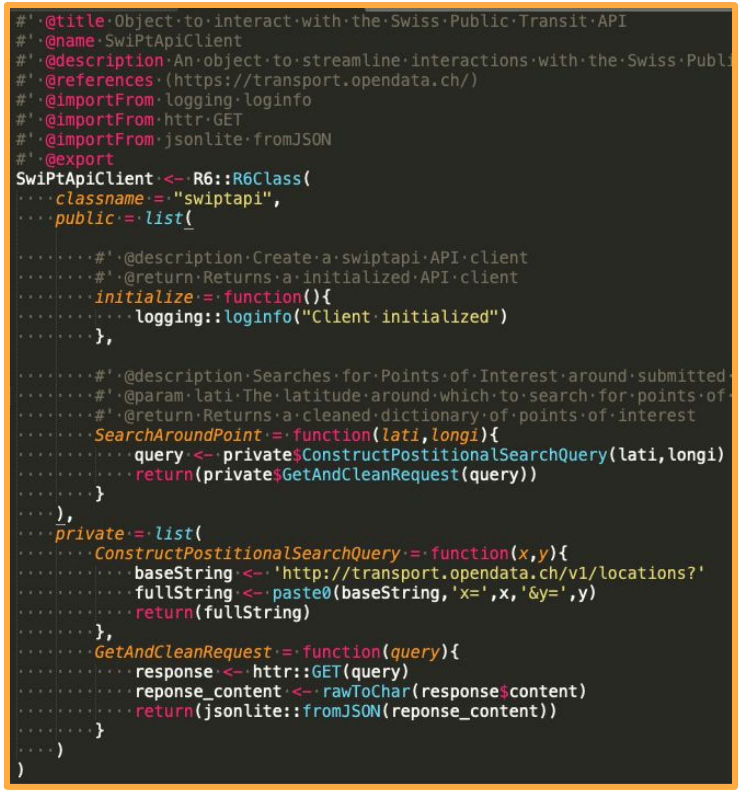
https://www.linkedin.com/in/theodore-bakanas-b839714

Objects

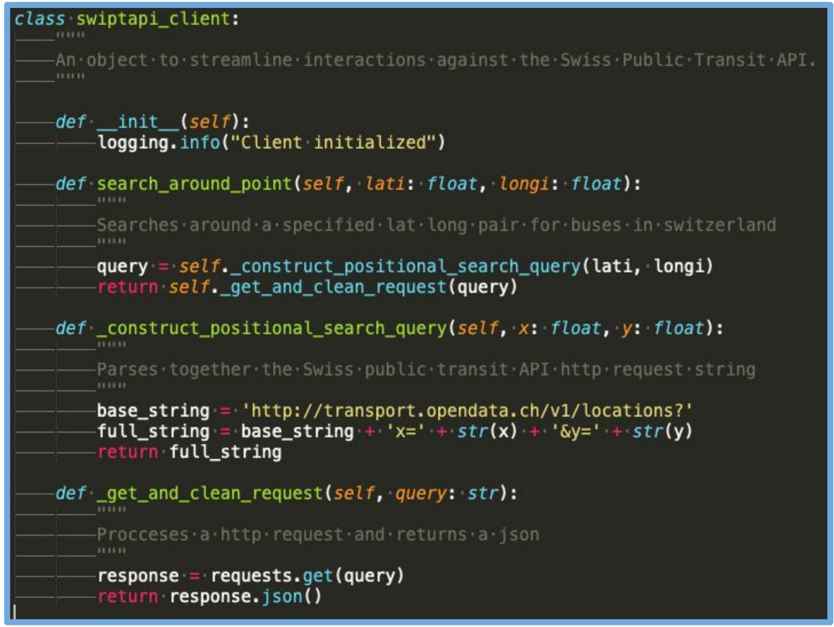
- __init__ and initialize in py and R are roughly equivalent
- documentation can be approached in the same ways as functions (roxygen and docstrings)
- R: treats everything as lists. Public and private methods are two different lists
- py: public and private methods are distinguished by leading underscores (_)



```
starterkits
  build
  scripts
  swiptapi-py
    swiptapi
      __pycache__
      __init__.py
      __version__.py
      functions.py
      swclient.py
    swiptapi.egg-info
    setup.py
  swiptapi-r
    man
    R
      client.R
      functions.R
    DESCRIPTION
    NAMESPACE
  .gitignore
  LICENSE
  README.md
```

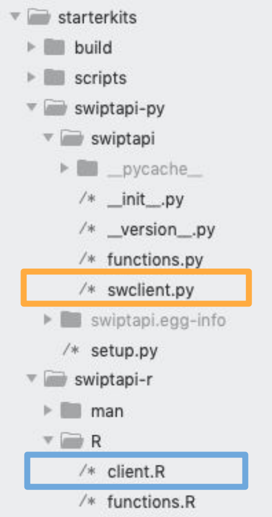


```r
#' @title Object to interact with the Swiss Public Transit API
#' @name SwiPtApiClient
#' @description An object to streamline interactions with the Swiss Publi
#' @references (https://transport.opendata.ch/)
#' @importFrom logging loginfo
#' @importFrom httr GET
#' @importFrom jsonlite fromJSON
#' @export
SwiPtApiClient <- R6::R6Class(
    classname = "swiptapi",
    public = list(

        #' @description Create a swiptapi API client
        #' @return Returns a initialized API client
        initialize = function(){
            logging::loginfo("Client initialized")
        },

        #' @description Searches for Points of Interest around submitted
        #' @param lati The latitude around which to search for points of
        #' @return Returns a cleaned dictionary of points of interest
        SearchAroundPoint = function(lati,longi){
            query <- private$ConstructPostitionalSearchQuery(lati,longi)
            return(private$GetAndCleanRequest(query))
        }
    ),
    private = list(
        ConstructPostitionalSearchQuery = function(x,y){
            baseString <- 'http://transport.opendata.ch/v1/locations?'
            fullString <- paste0(baseString,'x=',x,'&y=',y)
            return(fullString)
        },
        GetAndCleanRequest = function(query){
            response <- httr::GET(query)
            reponse_content <- rawToChar(response$content)
            return(jsonlite::fromJSON(reponse_content))
        }
    )
)
```



```python
class swiptapi_client:
    """
    An object to streamline interactions against the Swiss Public Transit API.
    """

    def __init__(self):
        logging.info("Client initialized")

    def search_around_point(self, lati: float, longi: float):
        """
        Searches around a specified lat long pair for buses in switzerland
        """
        query = self._construct_positional_search_query(lati, longi)
        return self._get_and_clean_request(query)

    def _construct_positional_search_query(self, x: float, y: float):
        """
        Parses together the Swiss public transit API http request string
        """
        base_string = 'http://transport.opendata.ch/v1/locations?'
        full_string = base_string + 'x=' + str(x) + '&y=' + str(y)
        return full_string

    def _get_and_clean_request(self, query: str):
        """
        Procceses a http request and returns a json
        """
        response = requests.get(query)
        return response.json()
```

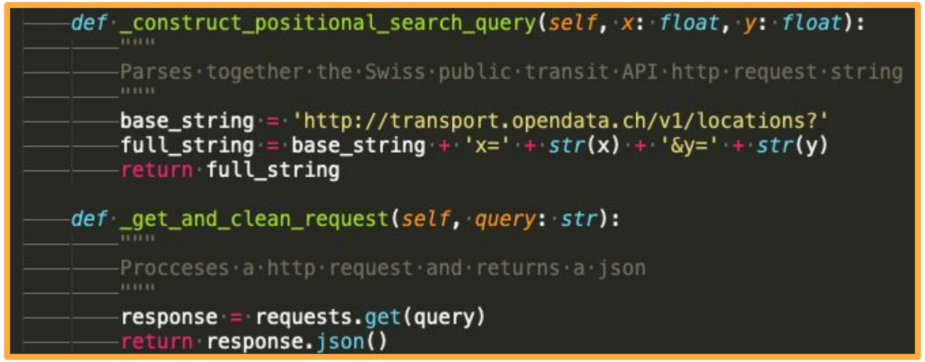
Hitting APIs



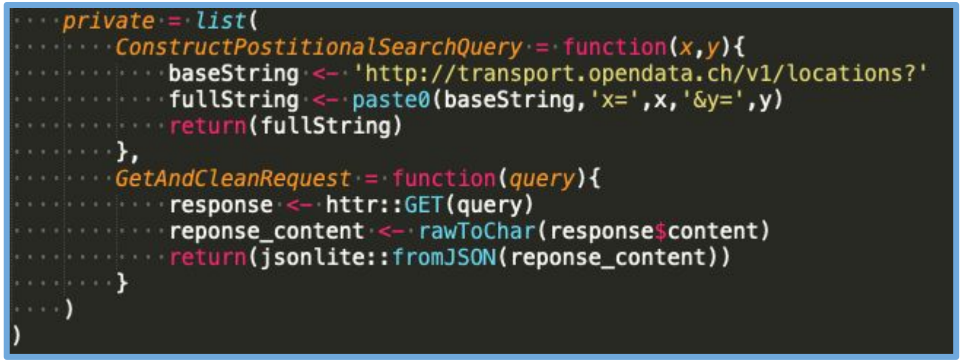
```
▼ starterkits
  ▶ build
  ▶ scripts
  ▼ swiptapi-py
    ▼ swiptapi
      ▶ __pycache__
        /* __init__.py
        /* __version__.py
        /* functions.py
        /* swclient.py
    ▶ swiptapi.egg-info
      /* setup.py
  ▼ swiptapi-r
    ▶ man
    ▼ R
        /* client.R
        /* functions.R
```

Streamlining interaction with an API is a great use of a package! Both R and python provide multiple package submitting http requests to an API.

- Packages used in this repo:
  - R: httr
  - py: requests
- Exploring an API
  - find the documentation! https://transport.opendata.ch/docs.html
  - play around with different GET requests
  - save your response! you can usually extract it to a dictionary/json



```python
    def _construct_positional_search_query(self, x: float, y: float):
        """
        Parses together the Swiss public transit API http request string
        """
        base_string = 'http://transport.opendata.ch/v1/locations?'
        full_string = base_string + 'x=' + str(x) + '&y=' + str(y)
        return full_string

    def _get_and_clean_request(self, query: str):
        """
        Procceses a http request and returns a json
        """
        response = requests.get(query)
        return response.json()
```



```r
    private = list(
        ConstructPostitionalSearchQuery = function(x,y){
            baseString <- 'http://transport.opendata.ch/v1/locations?'
            fullString <- paste0(baseString,'x=',x,'&y=',y)
            return(fullString)
        },
        GetAndCleanRequest = function(query){
            response <- httr::GET(query)
            reponse_content <- rawToChar(response$content)
            return(jsonlite::fromJSON(reponse_content))
        }
    )
)
```