Wikidata Lab X

Writing bots on Wikidata

[[User:Mike Peel]]

NeuroMat, São Paulo, Brazil

17 September 2018

Writing bot code

Figuring out what to edit

Running the bot

WIKIDATA

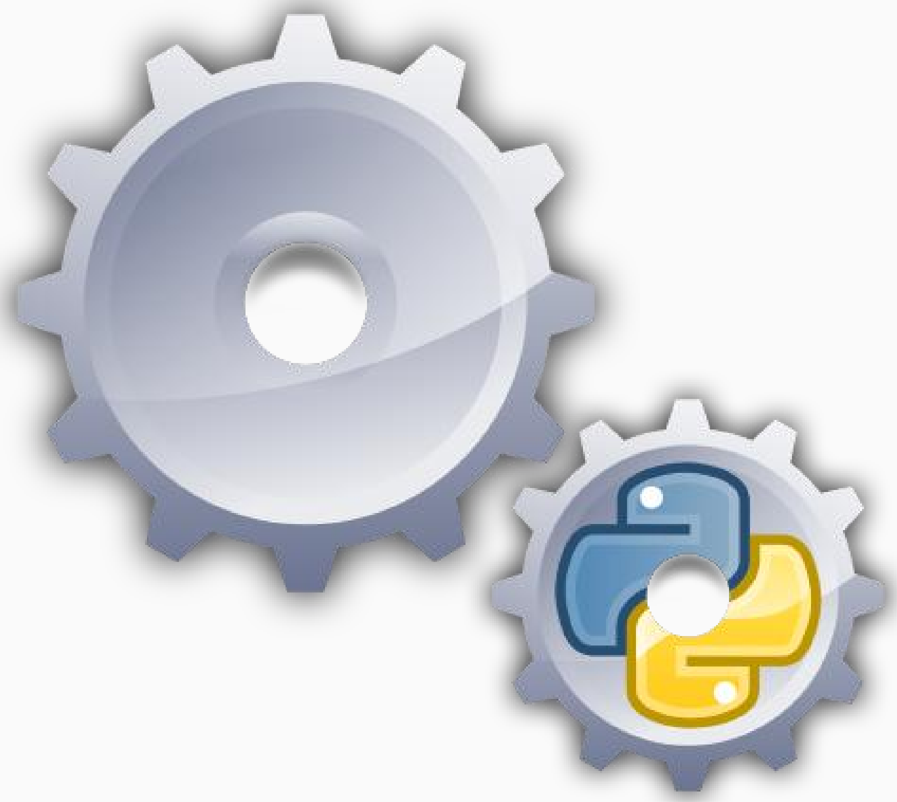
Caveats

I am still learning how to write bots!

There may be easier ways than those I will show here...

There are definitely better ways to write code!

... however, this has worked for the last 2 million edits!



https://www.mediawiki.org/wiki/Manual:Pywikibot

Pywikibot

Python framework to edit MediaWiki content

Need Python 2 or 3

‘pip install pywikibot’

What can be done?

Create a page

Edit a page

Delete a page

Undelete a page

Search

... and lots more!

Edits need to be 100% accurate

Focus on routine tasks that are easily described

Keep things modular/simple as much as possible

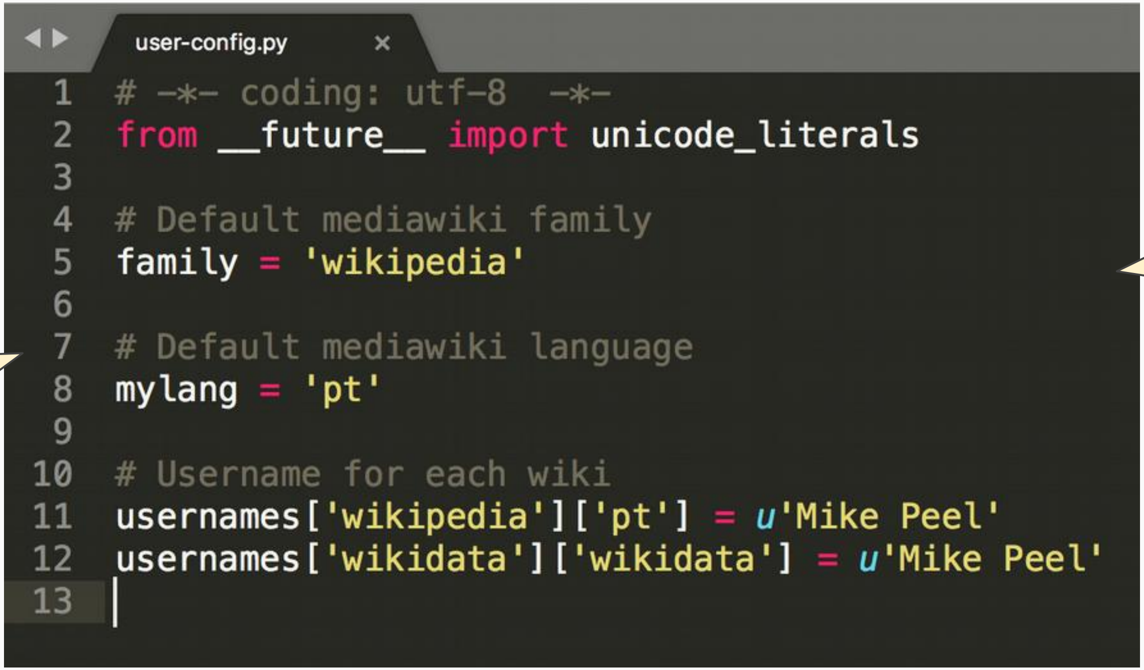
Set up the pywikibot user configuration file

Initial definitions

Language

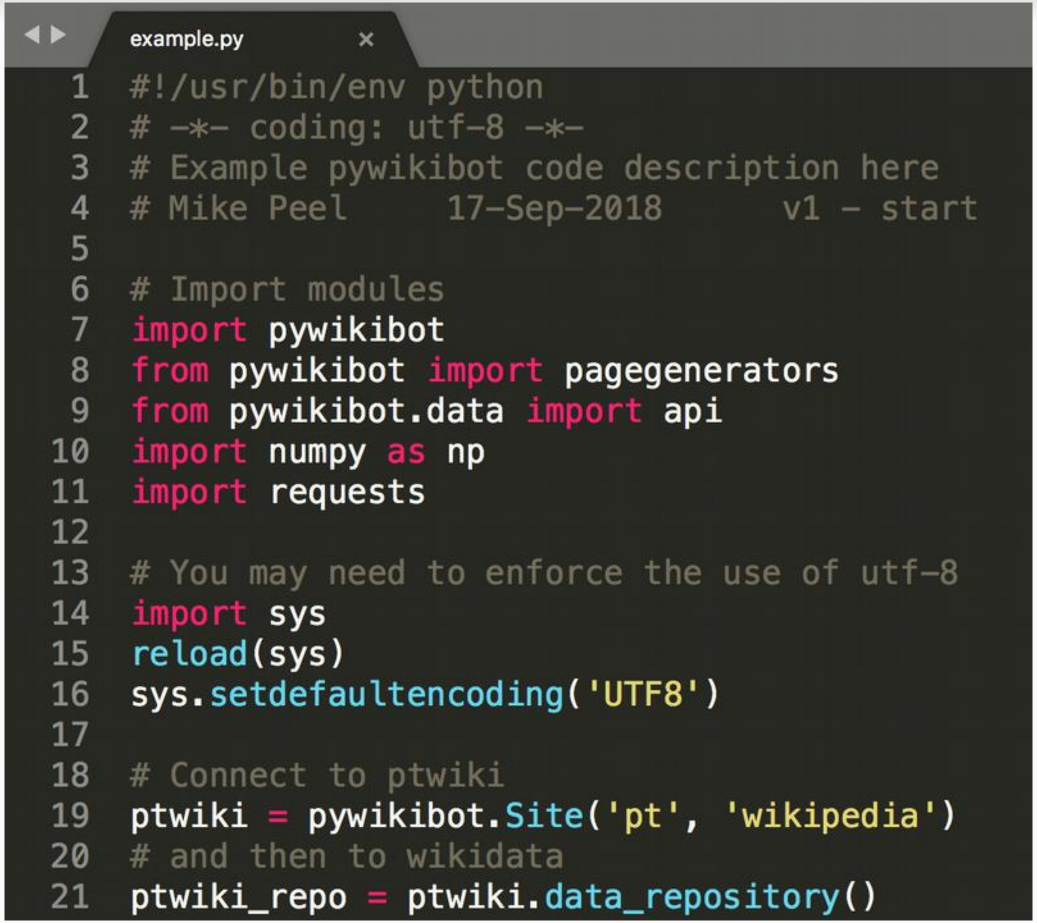
MediaWiki family

Usernames



```python
# -*- coding: utf-8  -*-
from __future__ import unicode_literals

# Default mediawiki family
family = 'wikipedia'

# Default mediawiki language
mylang = 'pt'

# Username for each wiki
usernames['wikipedia']['pt'] = u'Mike Peel'
usernames['wikidata']['wikidata'] = u'Mike Peel'

```

Starting the code and connecting to ptwiki and Wikidata



```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Example pywikibot code description here
# Mike Peel     17-Sep-2018       v1 - start

# Import modules
import pywikibot
from pywikibot import pagegenerators
from pywikibot.data import api
import numpy as np
import requests

# You may need to enforce the use of utf-8
import sys
reload(sys)
sys.setdefaultencoding('UTF8')

# Connect to ptwiki
ptwiki = pywikibot.Site('pt', 'wikipedia')
# and then to wikidata
ptwiki_repo = ptwiki.data_repository()
```

Initial definitions

Import modules

Connect!

Documentation!

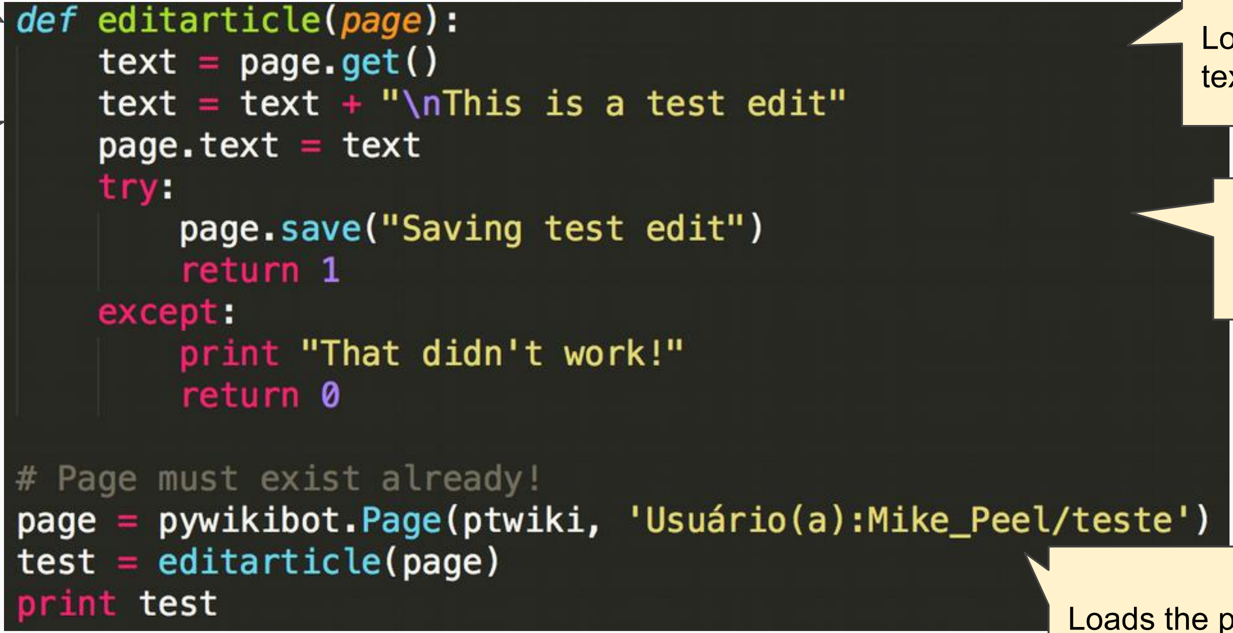
Python2: use utf-8

Doing a test edit to the Portuguese Wikipedia

Defines the function

Making the edit

Running the function

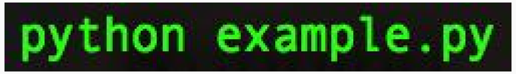


```python
def editarticle(page):
    text = page.get()
    text = text + "\nThis is a test edit"
    page.text = text
    try:
        page.save("Saving test edit")
        return 1
    except:
        print "That didn't work!"
        return 0

# Page must exist already!
page = pywikibot.Page(ptwiki, 'Usuário(a):Mike_Peel/teste')
test = editarticle(page)
print test
```

Loads the page text

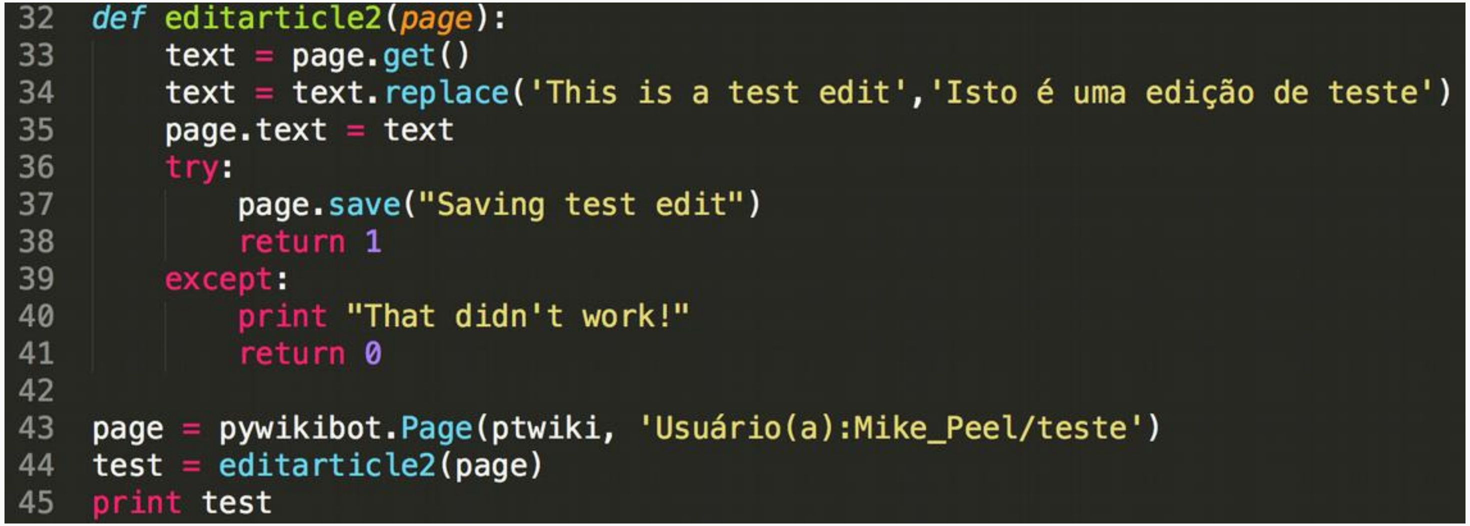
Saving the edit

Loads the page



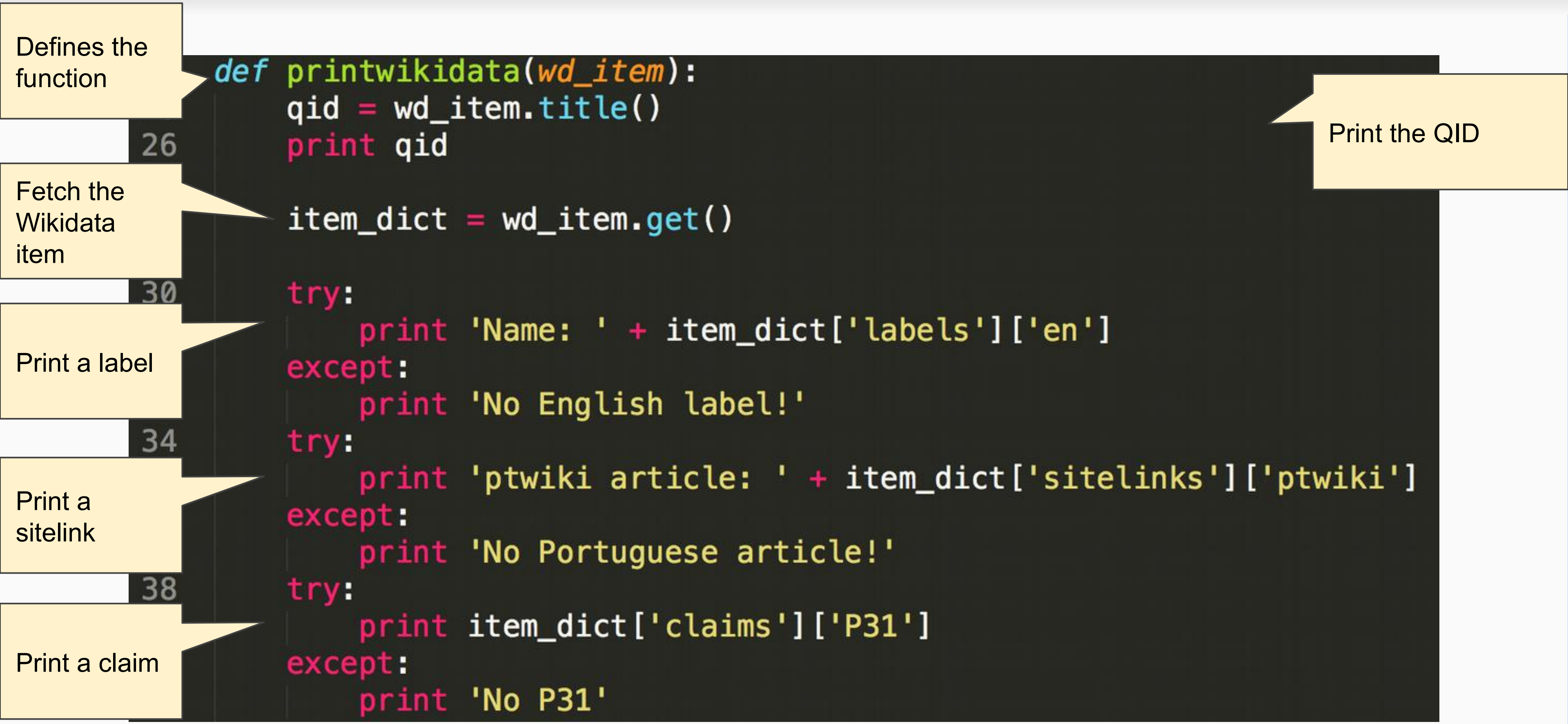
```
python example.py
```

Replacing text in the Portuguese Wikipedia



```python
def editarticle2(page):
    text = page.get()
    text = text.replace('This is a test edit','Isto é uma edição de teste')
    page.text = text
    try:
        page.save("Saving test edit")
        return 1
    except:
        print "That didn't work!"
        return 0

page = pywikibot.Page(ptwiki, 'Usuário(a):Mike_Peel/teste')
test = editarticle2(page)
print test
```

Printing the QID, label, sitelink and a claim from Wikidata



Defines the function

Print the QID

Fetch the Wikidata item

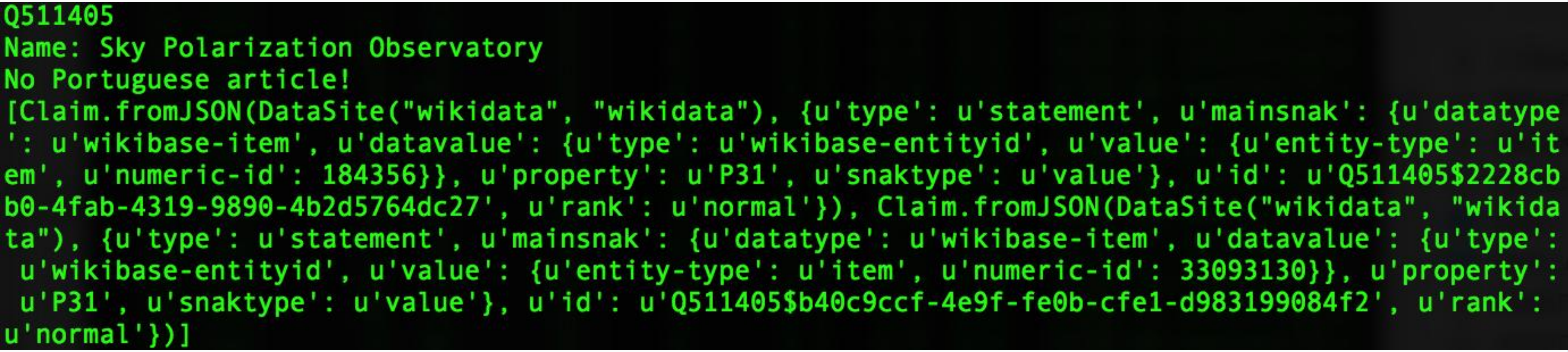
Print a label

Print a sitelink

Print a claim

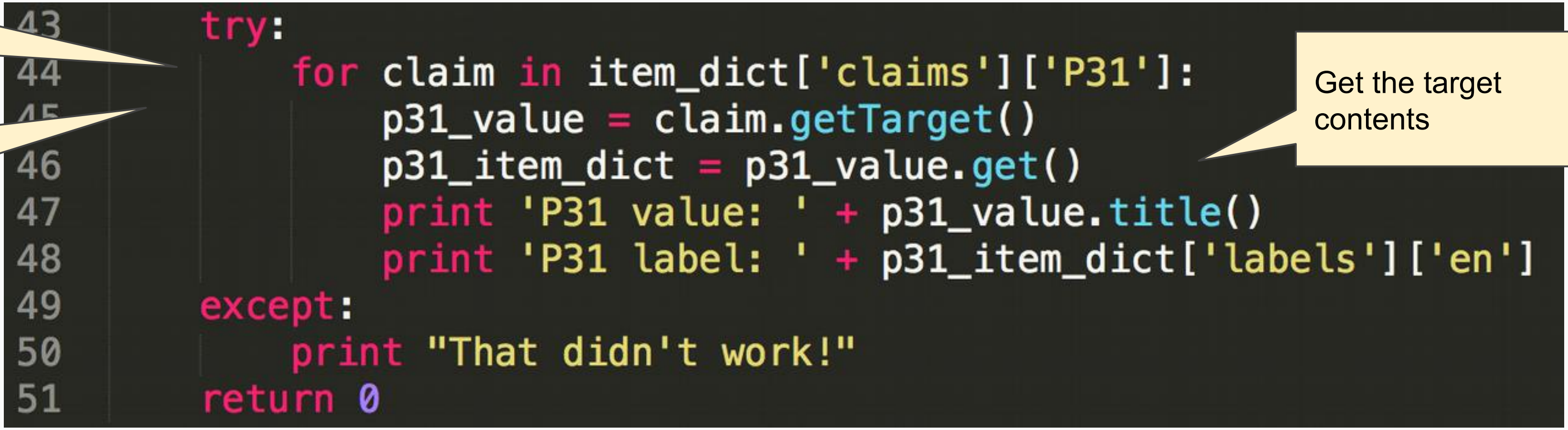
```
def printwikidata(wd_item):
    qid = wd_item.title()
    print qid

    item_dict = wd_item.get()

    try:
        print 'Name: ' + item_dict['labels']['en']
    except:
        print 'No English label!'
    try:
        print 'ptwiki article: ' + item_dict['sitelinks']['ptwiki']
    except:
        print 'No Portuguese article!'
    try:
        print item_dict['claims']['P31']
    except:
        print 'No P31'
```

The output from the previous slide



```
Q511405
Name: Sky Polarization Observatory
No Portuguese article!
[Claim.fromJSON(DataSite("wikidata", "wikidata"), {u'type': u'statement', u'mainsnak': {u'datatype
': u'wikibase-item', u'datavalue': {u'type': u'wikibase-entityid', u'value': {u'entity-type': u'it
em', u'numeric-id': 184356}}, u'property': u'P31', u'snaktype': u'value'}, u'id': u'Q511405$2228cb
b0-4fab-4319-9890-4b2d5764dc27', u'rank': u'normal'}), Claim.fromJSON(DataSite("wikidata", "wikida
ta"), {u'type': u'statement', u'mainsnak': {u'datatype': u'wikibase-item', u'datavalue': {u'type':
 u'wikibase-entityid', u'value': {u'entity-type': u'item', u'numeric-id': 33093130}}, u'property':
 u'P31', u'snaktype': u'value'}, u'id': u'Q511405$b40c9ccf-4e9f-fe0b-cfe1-d983199084f2', u'rank':
u'normal'})]
```

Printing the property values correctly



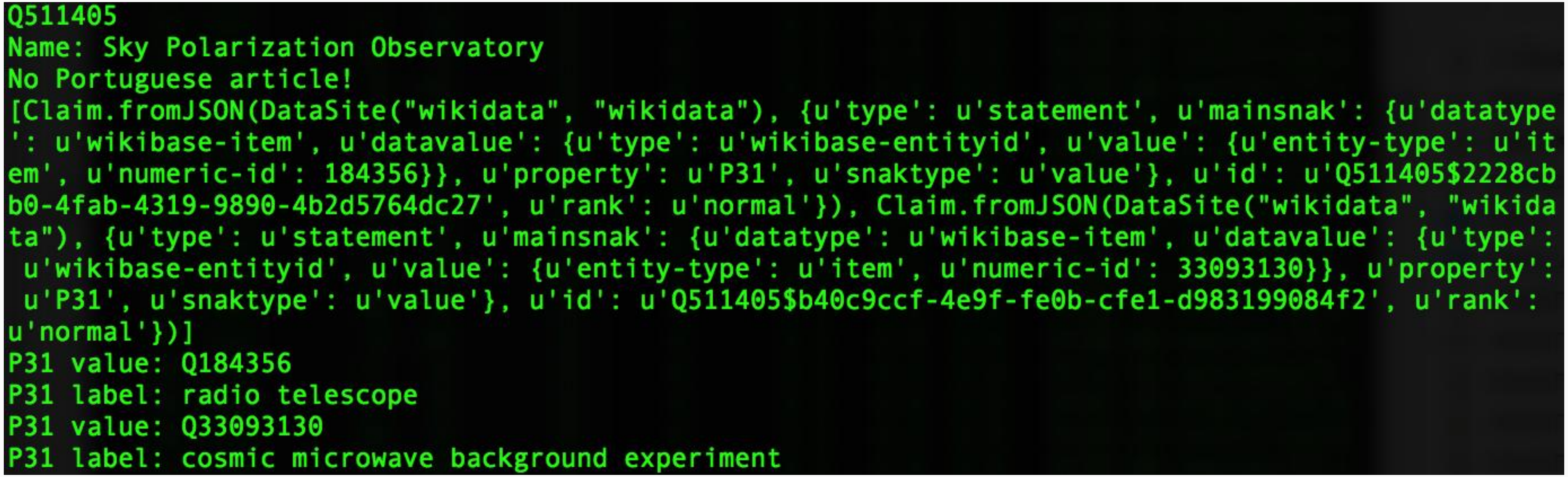
```
try:
    for claim in item_dict['claims']['P31']:
        p31_value = claim.getTarget()
        p31_item_dict = p31_value.get()
        print 'P31 value: ' + p31_value.title()
        print 'P31 label: ' + p31_item_dict['labels']['en']
except:
    print "That didn't work!"
return 0
```

Loop over claims

Get the claim target

Get the target contents

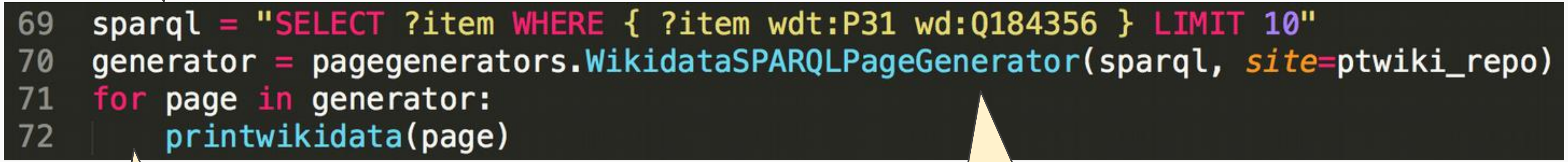
The output from the previous slides



```
Q511405
Name: Sky Polarization Observatory
No Portuguese article!
[Claim.fromJSON(DataSite("wikidata", "wikidata"), {u'type': u'statement', u'mainsnak': {u'datatype
': u'wikibase-item', u'datavalue': {u'type': u'wikibase-entityid', u'value': {u'entity-type': u'it
em', u'numeric-id': 184356}}, u'property': u'P31', u'snaktype': u'value'}, u'id': u'Q511405$2228cb
b0-4fab-4319-9890-4b2d5764dc27', u'rank': u'normal'}), Claim.fromJSON(DataSite("wikidata", "wikida
ta"), {u'type': u'statement', u'mainsnak': {u'datatype': u'wikibase-item', u'datavalue': {u'type':
 u'wikibase-entityid', u'value': {u'entity-type': u'item', u'numeric-id': 33093130}}, u'property':
 u'P31', u'snaktype': u'value'}, u'id': u'Q511405$b40c9ccf-4e9f-fe0b-cfe1-d983199084f2', u'rank':
u'normal'})]
P31 value: Q184356
P31 label: radio telescope
P31 value: Q33093130
P31 label: cosmic microwave background experiment
```

How to generate lists of pages to edit - using a sparql query

Sparql query

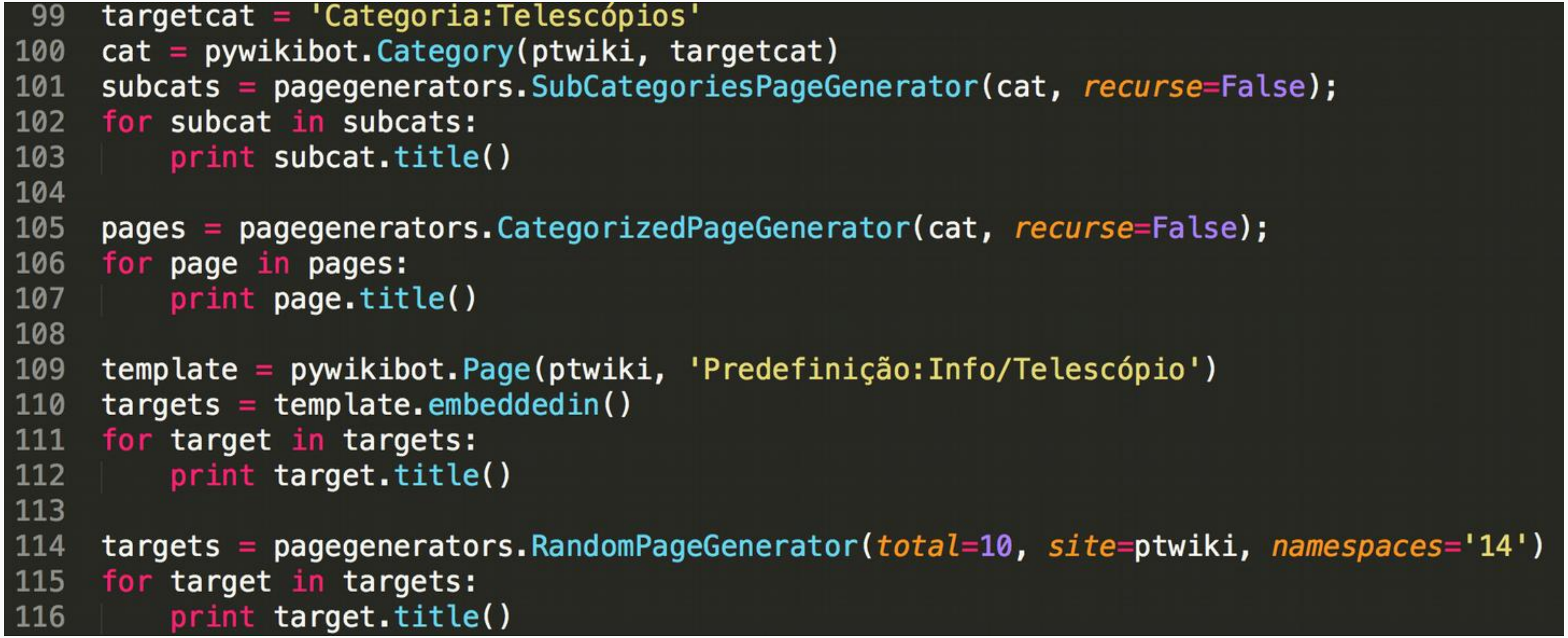


```
sparql = "SELECT ?item WHERE { ?item wdt:P31 wd:Q184356 } LIMIT 10"
generator = pagegenerators.WikidataSPARQLPageGenerator(sparql, site=ptwiki_repo)
for page in generator:
    printwikidata(page)
```

Loop over results

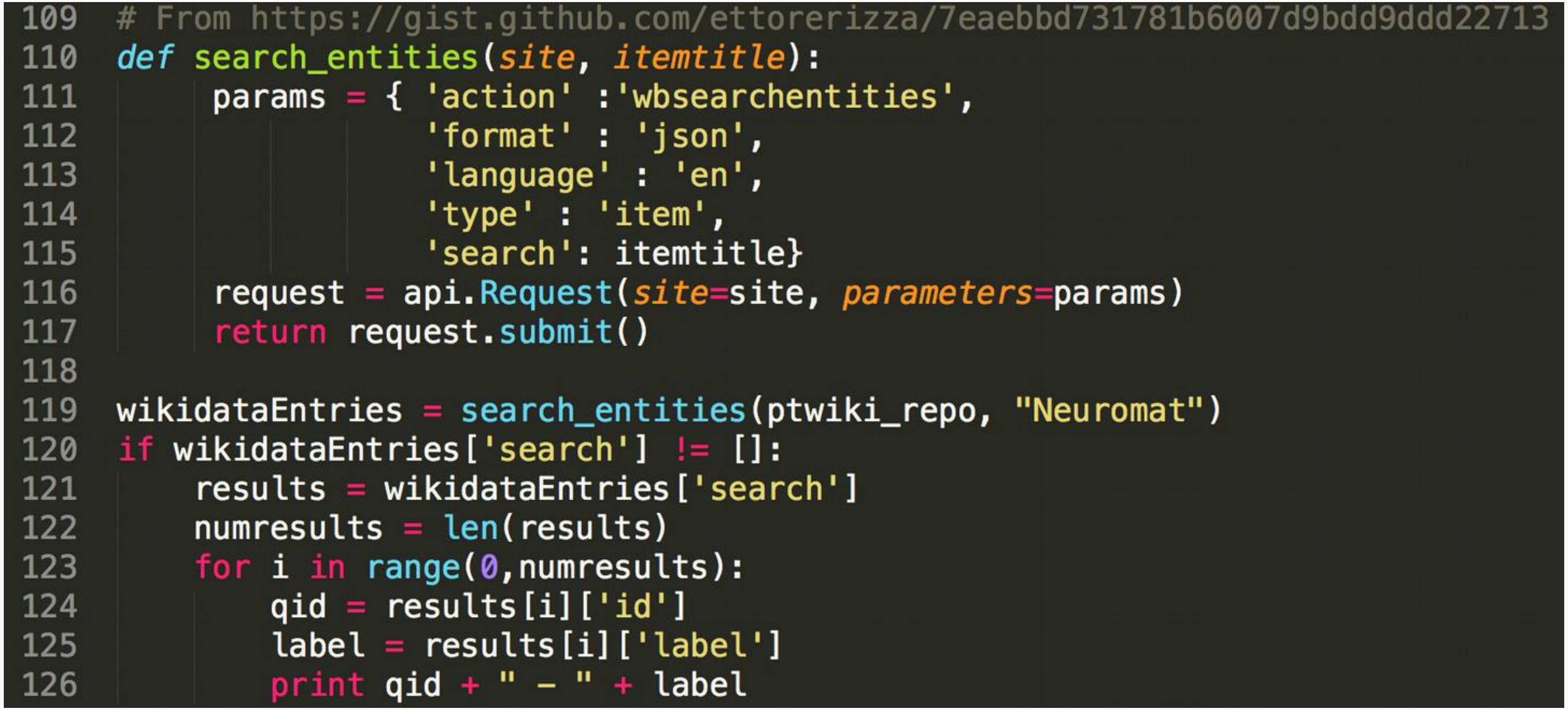
Run the query

How to generate lists of pages to edit - using categories and template uses



```
targetcat = 'Categoria:Telescópios'
cat = pywikibot.Category(ptwiki, targetcat)
subcats = pagegenerators.SubCategoriesPageGenerator(cat, recurse=False);
for subcat in subcats:
    print subcat.title()

pages = pagegenerators.CategorizedPageGenerator(cat, recurse=False);
for page in pages:
    print page.title()

template = pywikibot.Page(ptwiki, 'Predefinição:Info/Telescópio')
targets = template.embeddedin()
for target in targets:
    print target.title()

targets = pagegenerators.RandomPageGenerator(total=10, site=ptwiki, namespaces='14')
for target in targets:
    print target.title()
```

Search for existing Wikidata items



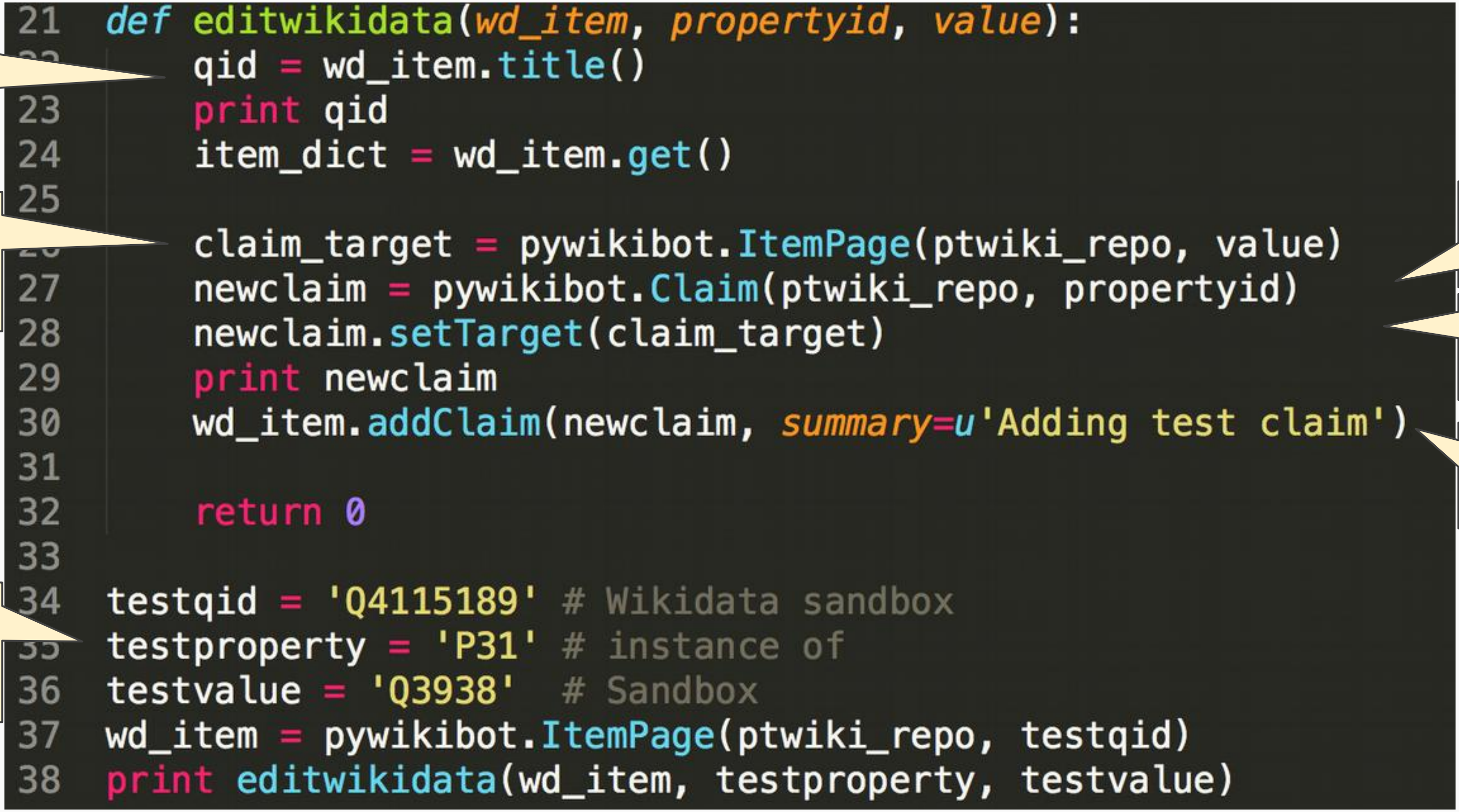
```
# From https://gist.github.com/ettorerizza/7eaebbd731781b6007d9bdd9ddd22713
def search_entities(site, itemtitle):
     params = { 'action' :'wbsearchentities',
                'format' : 'json',
                'language' : 'en',
                'type' : 'item',
                'search': itemtitle}
     request = api.Request(site=site, parameters=params)
     return request.submit()

wikidataEntries = search_entities(ptwiki_repo, "Neuromat")
if wikidataEntries['search'] != []:
    results = wikidataEntries['search']
    numresults = len(results)
    for i in range(0,numresults):
        qid = results[i]['id']
        label = results[i]['label']
        print qid + " - " + label
```

Doing a test edit to Wikidata - adding ‘instance of’ = ‘sandbox’

Get the item

Get the target item

QID and property ID to edit



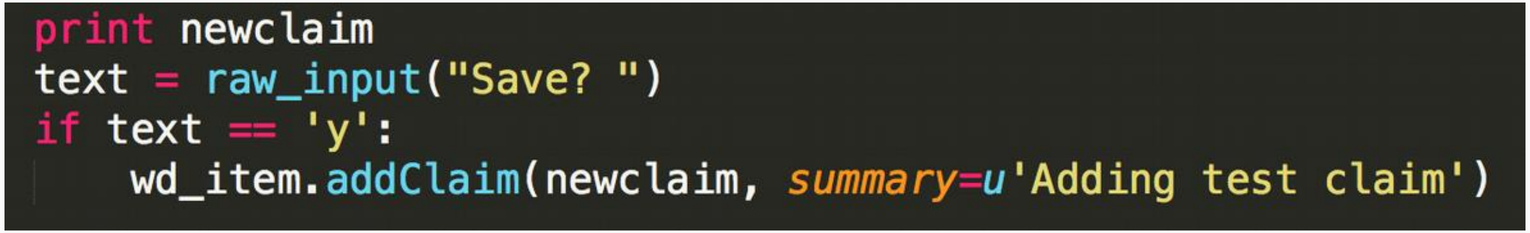
```python
def editwikidata(wd_item, propertyid, value):
    qid = wd_item.title()
    print qid
    item_dict = wd_item.get()

    claim_target = pywikibot.ItemPage(ptwiki_repo, value)
    newclaim = pywikibot.Claim(ptwiki_repo, propertyid)
    newclaim.setTarget(claim_target)
    print newclaim
    wd_item.addClaim(newclaim, summary=u'Adding test claim')

    return 0

testqid = 'Q4115189' # Wikidata sandbox
testproperty = 'P31' # instance of
testvalue = 'Q3938'  # Sandbox
wd_item = pywikibot.ItemPage(ptwiki_repo, testqid)
print editwikidata(wd_item, testproperty, testvalue)
```

Create a claim

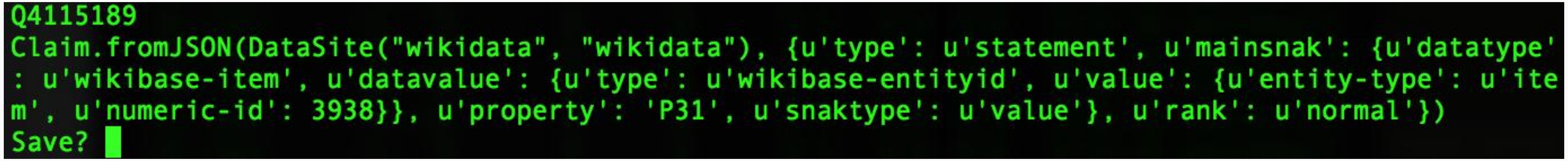
Set claim target

Save the claim

Manually checking the edit before saving it

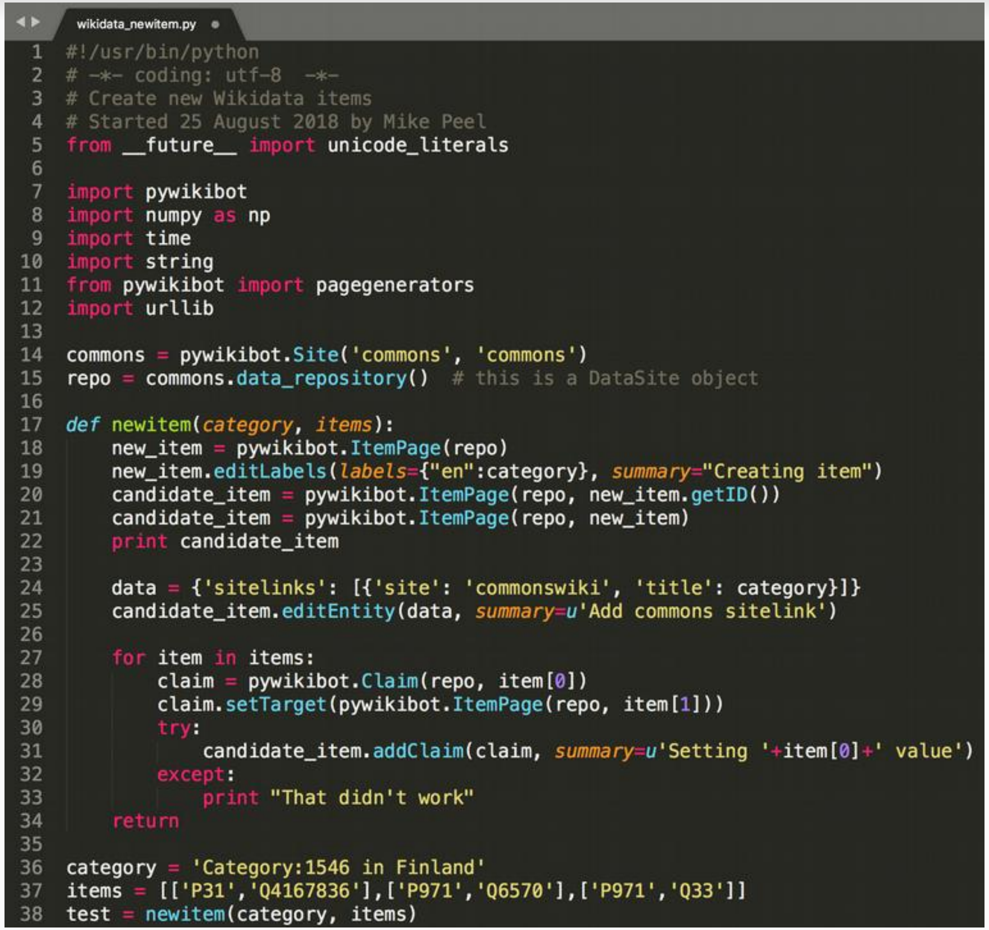


```
print newclaim
text = raw_input("Save? ")
if text == 'y':
    wd_item.addClaim(newclaim, summary=u'Adding test claim')
```



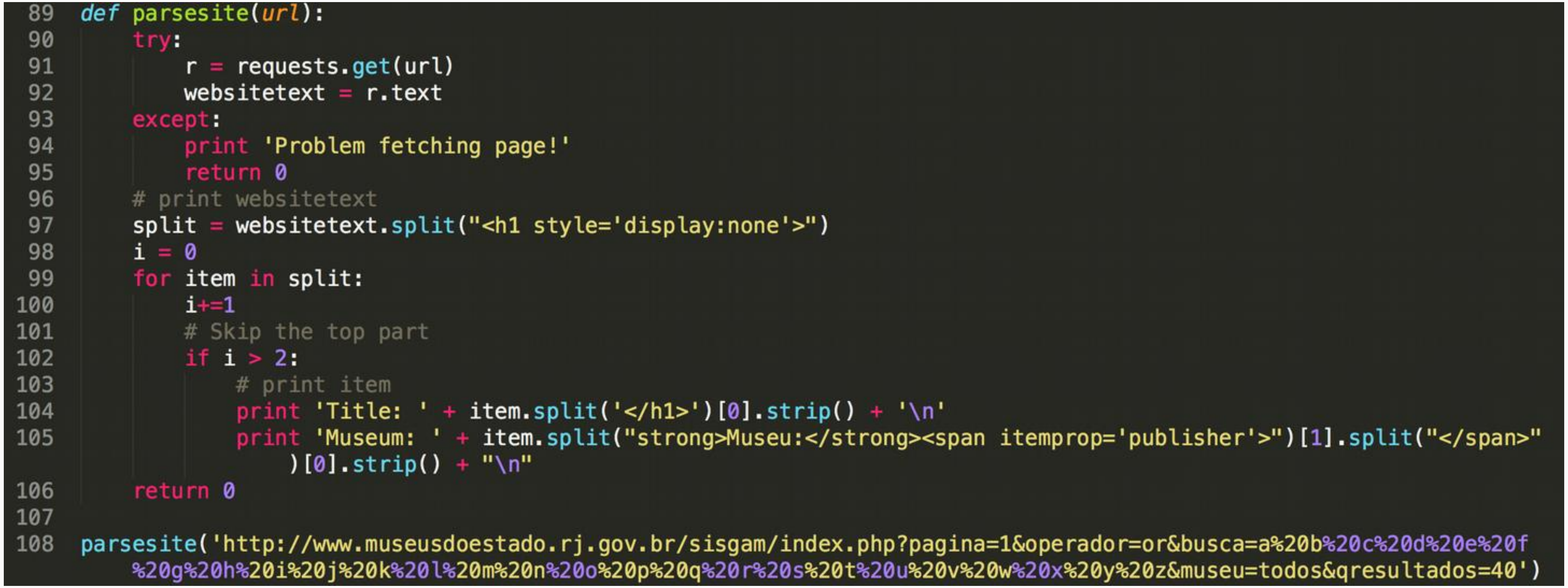
```
Q4115189
Claim.fromJSON(DataSite("wikidata", "wikidata"), {u'type': u'statement', u'mainsnak': {u'datatype'
: u'wikibase-item', u'datavalue': {u'type': u'wikibase-entityid', u'value': {u'entity-type': u'ite
m', u'numeric-id': 3938}}, u'property': 'P31', u'snaktype': u'value'}, u'rank': u'normal'})
Save? 
```

Creating a new Wikidata entry



```python
#!/usr/bin/python
# -*- coding: utf-8  -*-
# Create new Wikidata items
# Started 25 August 2018 by Mike Peel
from __future__ import unicode_literals

import pywikibot
import numpy as np
import time
import string
from pywikibot import pagegenerators
import urllib

commons = pywikibot.Site('commons', 'commons')
repo = commons.data_repository()  # this is a DataSite object

def newitem(category, items):
    new_item = pywikibot.ItemPage(repo)
    new_item.editLabels(labels={"en":category}, summary="Creating item")
    candidate_item = pywikibot.ItemPage(repo, new_item.getID())
    candidate_item = pywikibot.ItemPage(repo, new_item)
    print candidate_item

    data = {'sitelinks': [{'site': 'commonswiki', 'title': category}]}
    candidate_item.editEntity(data, summary=u'Add commons sitelink')

    for item in items:
        claim = pywikibot.Claim(repo, item[0])
        claim.setTarget(pywikibot.ItemPage(repo, item[1]))
        try:
            candidate_item.addClaim(claim, summary=u'Setting '+item[0]+' value')
        except:
            print "That didn't work"
    return

category = 'Category:1546 in Finland'
items = [['P31','Q4167836'],['P971','Q6570'],['P971','Q33']]
test = newitem(category, items)
```

Scrape a website



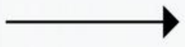
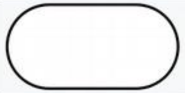
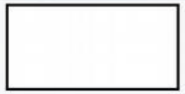
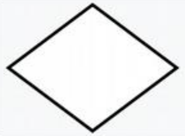
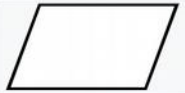
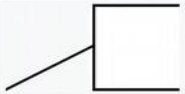
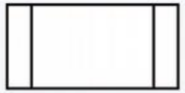
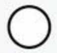
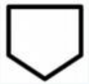
```python
def parsesite(url):
    try:
        r = requests.get(url)
        websitetext = r.text
    except:
        print 'Problem fetching page!'
        return 0
    # print websitetext
    split = websitetext.split("<h1 style='display:none'>")
    i = 0
    for item in split:
        i+=1
        # Skip the top part
        if i > 2:
            # print item
            print 'Title: ' + item.split('</h1>')[0].strip() + '\n'
            print 'Museum: ' + item.split("strong>Museu:</strong><span itemprop='publisher'>")[1].split("</span>")[0].strip() + "\n"
    return 0

parsesite('http://www.museusdoestado.rj.gov.br/sisgam/index.php?pagina=1&operador=or&busca=a%20b%20c%20d%20e%20f%20g%20h%20i%20j%20k%20l%20m%20n%20o%20p%20q%20r%20s%20t%20u%20v%20w%20x%20y%20z&museu=todos&qresultados=40')
```

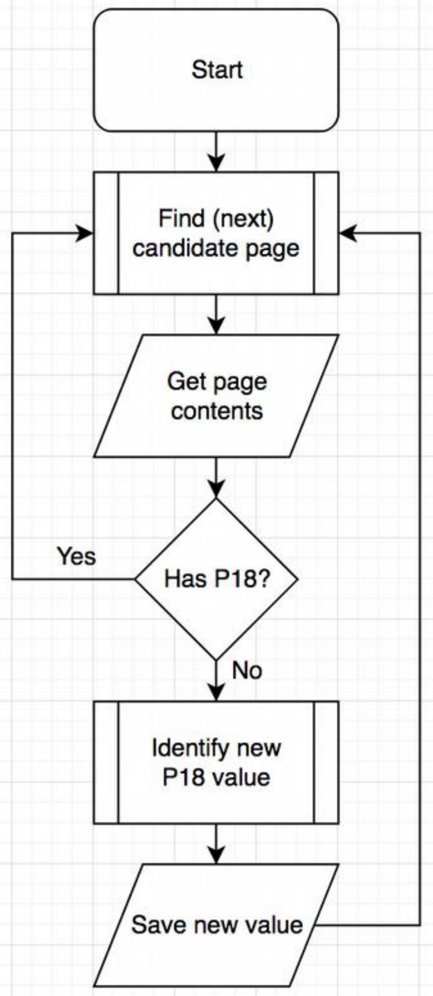
Flow charts

Write out the code logic in
a series of steps

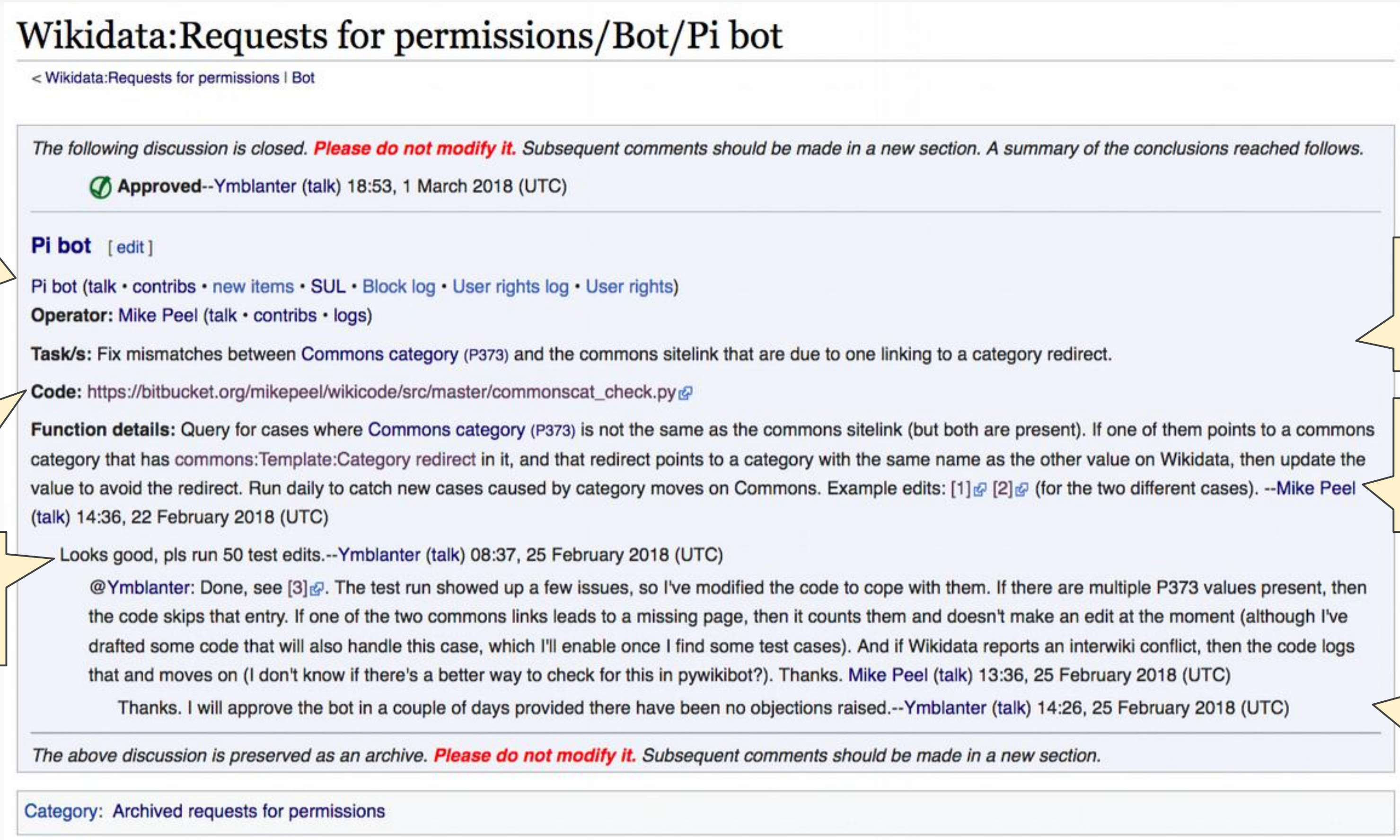
Identify main questions/if statements

Create online at www.draw.io

ANSI/ISO Shape	Name
	Flowline (Arrowhead)[15]
	Terminal[14]
	Process[15]
	Decision[15]
	Input/Output[15]
	Annotation[14] (Comment)[15]
	Predefined Process[14]
	On-page Connector[14]
	Off-page Connector[14]



Submitting a bot request



Wikidata:Requests for permissions/Bot/Pi bot

< Wikidata:Requests for permissions | Bot

The following discussion is closed. ***Please do not modify it.*** *Subsequent comments should be made in a new section. A summary of the conclusions reached follows.*

Approved--Ymblanter (talk) 18:53, 1 March 2018 (UTC)

Pi bot [edit]

Pi bot (talk • contribs • new items • SUL • Block log • User rights log • User rights)

Operator: Mike Peel (talk • contribs • logs)

Task/s: Fix mismatches between Commons category (P373) and the commons sitelink that are due to one linking to a category redirect.

Code: https://bitbucket.org/mikepeel/wikicode/src/master/commonscat_check.py

Function details: Query for cases where Commons category (P373) is not the same as the commons sitelink (but both are present). If one of them points to a commons category that has commons:Template:Category redirect in it, and that redirect points to a category with the same name as the other value on Wikidata, then update the value to avoid the redirect. Run daily to catch new cases caused by category moves on Commons. Example edits: [1] [2] (for the two different cases). --Mike Peel (talk) 14:36, 22 February 2018 (UTC)

Looks good, pls run 50 test edits.--Ymblanter (talk) 08:37, 25 February 2018 (UTC)

@Ymblanter: Done, see [3]. The test run showed up a few issues, so I've modified the code to cope with them. If there are multiple P373 values present, then the code skips that entry. If one of the two commons links leads to a missing page, then it counts them and doesn't make an edit at the moment (although I've drafted some code that will also handle this case, which I'll enable once I find some test cases). And if Wikidata reports an interwiki conflict, then the code logs that and moves on (I don't know if there's a better way to check for this in pywikibot?). Thanks. Mike Peel (talk) 13:36, 25 February 2018 (UTC)

Thanks. I will approve the bot in a couple of days provided there have been no objections raised.--Ymblanter (talk) 14:26, 25 February 2018 (UTC)

The above discussion is preserved as an archive. ***Please do not modify it.*** *Subsequent comments should be made in a new section.*

Category: Archived requests for permissions

Separate bot account

Code online somewhere

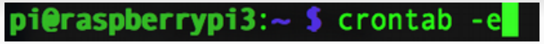
Do a test run

Short task description

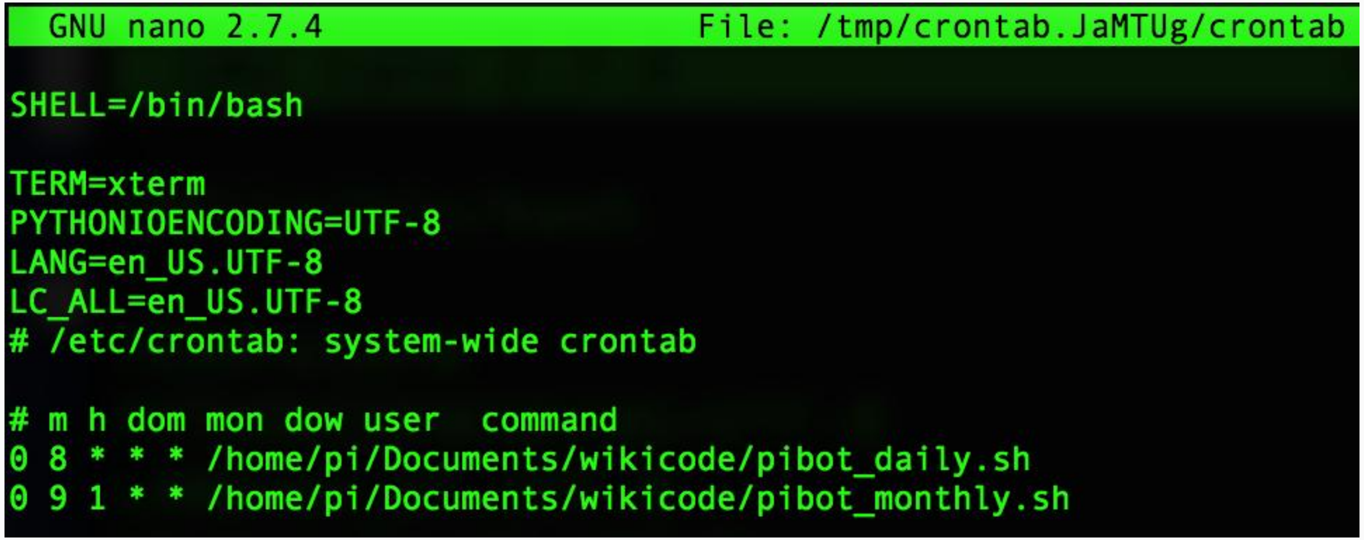
Full details

Approval

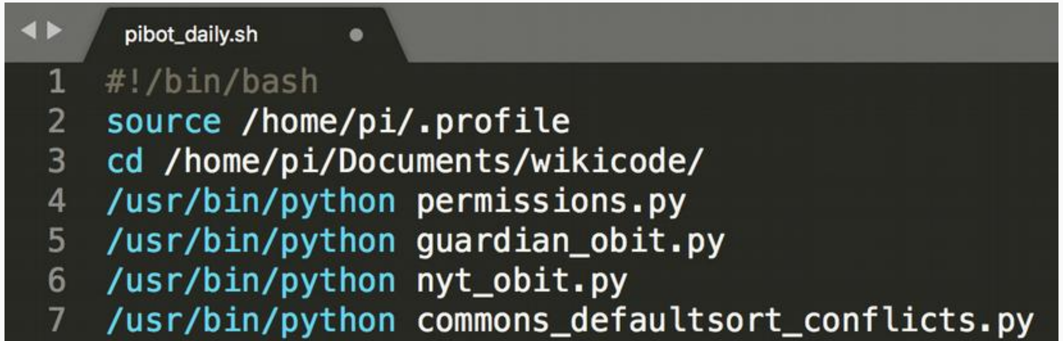
Cron jobs to automatically run the bot on a set schedule



```
pi@raspberrypi3:~ $ crontab -e
```



```
 GNU nano 2.7.4                    File: /tmp/crontab.JaMTUg/crontab

SHELL=/bin/bash

TERM=xterm
PYTHONIOENCODING=UTF-8
LANG=en_US.UTF-8
LC_ALL=en_US.UTF-8
# /etc/crontab: system-wide crontab

# m h dom mon dow user  command
0 8 * * * /home/pi/Documents/wikicode/pibot_daily.sh
0 9 1 * * /home/pi/Documents/wikicode/pibot_monthly.sh
```



pibot_daily.sh

```
#!/bin/bash
source /home/pi/.profile
cd /home/pi/Documents/wikicode/
/usr/bin/python permissions.py
/usr/bin/python guardian_obit.py
/usr/bin/python nyt_obit.py
/usr/bin/python commons_defaultsort_conflicts.py
```

Afternoon tasks

Identify routine tasks that could be automated, and writing out their logic chart

- Find Wikidata entry with no image but with a Commons category, give a list of suggestions and ask user to pick one before saving it
- ...
- Create by hand, or use: https://www.draw.io/

Writing web scrapers for different sites that add info into Wikidata

- http://www.museusdoestado.rj.gov.br/sisgam/
- ...
- Write code, or match contents on page with Wikidata properties

Useful links

- https://bitbucket.org/mikepeel/wikicode - pi bot scripts
- https://bitbucket.org/mikepeel/wikicode/src/master/example.py

- https://doc.wikimedia.org/pywikibot/master/api_ref/pywikibot.html

- https://www.google.com/ - good place to look for example code ;-)
- https://stackoverflow.com/ - where you'll actually find example code