

# **Working with APIs**

**Python basics**

Kunal Khurana

2023-10-13

# Table of contents

Learning outcomes . . . . .	2
Resources . . . . .	2
Pull out values from keys in repo_dict . . . . .	5
Summarizing top repositories . . . . .	6
Visualizing repositories using Plotly . . . . .	11
The Hacker News API . . . . .	16
Analysing all the top articles from Hackernews . . . . .	18

## Learning outcomes

- with web applicaiton programming interphase request specific information from a website to generate a visualization
- write programs that gather data they need and create a visualiztion
- use Github's API to explore the most starred projects on GitHub
- Use Requests package to issue and process results.
- Use plotly to generate and customize the appearance of charts

## Resources

- [Plotly guide](#)
- [Configure plotly visualizations](#)
- [API documentation](#)
- [Hacker news API](#)

```
import requests

# make an API and store the response

url = 'https://api.github.com/search/repositories?q=language:python&sort=stars'
headers = {'Accept': 'application/vnd.github.v3+json'} #uses specific version
r = requests.get(url, headers = headers) #using requests to make a call to API
print(f"Status code: {r.status_code}")
```

```

#store API response in a variable

response_dict = r.json() #using json method to convert it into a python dictionary

# process results

print(response_dict.keys())

```

Status code: 200

dict\_keys(['total\_count', 'incomplete\_results', 'items'])

```

response_dict = r.json()

print (f"Total repositories: {response_dict['total_count']}") #prints total count

# explore information about repos
repo_dicts = response_dict ['items'] #storing the list in repo_dicts
print(f"Repositories returned: {len(repo_dicts)}")

# examine the first repository
repo_dict = repo_dicts[0]
print(f"\nKeys: {len(repo_dict)}")
for key in sorted(repo_dict.keys()):
    print(key)

```

Total repositories: 9344484

Repositories returned: 30

Keys: 80

allow\_forking  
archive\_url  
archived  
assignees\_url  
blobs\_url  
branches\_url  
clone\_url  
collaborators\_url  
comments\_url  
commits\_url  
compare\_url

contents\_url  
contributors\_url  
created\_at  
default\_branch  
deployments\_url  
description  
disabled  
downloads\_url  
events\_url  
fork  
forks  
forks\_count  
forks\_url  
full\_name  
git\_commits\_url  
git\_refs\_url  
git\_tags\_url  
git\_url  
has\_discussions  
has\_downloads  
has\_issues  
has\_pages  
has\_projects  
has\_wiki  
homepage  
hooks\_url  
html\_url  
id  
is\_template  
issue\_comment\_url  
issue\_events\_url  
issues\_url  
keys\_url  
labels\_url  
language  
languages\_url  
license  
merges\_url  
milestones\_url  
mirror\_url  
name  
node\_id  
notifications\_url

```
open_issues
open_issues_count
owner
private
pulls_url
pushed_at
releases_url
score
size
ssh_url
stargazers_count
stargazers_url
statuses_url
subscribers_url
subscription_url
svn_url
tags_url
teams_url
topics
trees_url
updated_at
url
visibility
watchers
watchers_count
web_commit_signoff_required
```

### **Pull out values from keys in repo\_dict**

```
# explore information about repos
repo_dicts = response_dict ['items']

# examine first
repo_dict = repo_dicts[0]

print ("\n Selected information about first repository:")
print(f"Name: {repo_dict['name']}")
print(f"Owner: {repo_dict['owner']['login']}")
print(f"Stars: {repo_dict['stargazers_count']}")
print(f"Repository: {repo_dict['html_url']}")
print(f"Created: {repo_dict['created_at']}")
```

```
print(f"Updated: {repo_dict['updated_at']}")
print(f"Description: {repo_dict['description']}")
```

Selected information about first repository:  
 Name: Python-100-Days  
 Owner: jackfrued  
 Stars: 142124  
 Repository: <https://github.com/jackfrued/Python-100-Days>  
 Created: 2018-03-01T16:05:52Z  
 Updated: 2023-11-13T16:14:16Z  
 Description: Python - 100

```
# examine second
repo_dict = repo_dicts[1]

print ("\n Selected information about second repository:")
print(f"Name: {repo_dict['name']}")
print(f"Owner: {repo_dict['owner']['login']}")
print(f"Stars: {repo_dict['stargazers_count']}")
print (f"Repository: {repo_dict['html_url']}")
print(f"Created: {repo_dict['created_at']}")
print(f"Updated: {repo_dict['updated_at']}")
print(f"Description: {repo_dict['description']}")
```

Selected information about second repository:  
 Name: ColossalAI  
 Owner: hpcaitech  
 Stars: 35236  
 Repository: <https://github.com/hpcaitech/ColossalAI>  
 Created: 2021-10-28T16:19:44Z  
 Updated: 2023-11-13T20:52:01Z  
 Description: Making large AI models cheaper, faster and more accessible

## Summarizing top repositories

```
#prints total count first
response_dict = r.json()

print (f"Total repositories: {response_dict['total_count']}")
```

```

# explore information first
repo_dicts = response_dict['items']
print(f"\nRepositories returned: {len(repo_dicts)}")

print("\nSelected information about each repository:")
for repo_dict in repo_dicts:
    print(f"Name: {repo_dict['name']}")
    print(f"Owner: {repo_dict['owner']['login']}")
    print(f"Stars: {repo_dict['stargazers_count']}")
    print(f"Repository: {repo_dict['html_url']}")
    print(f>Description: {repo_dict['description']}")

```

Total repositories: 9344484

Repositories returned: 30

Selected information about each repository:

Name: Python-100-Days

Owner: jackfrued

Stars: 142124

Repository: <https://github.com/jackfrued/Python-100-Days>

Description: Python - 100

Name: ColossalAI

Owner: hpcaitech

Stars: 35236

Repository: <https://github.com/hpcaitech/ColossalAI>

Description: Making large AI models cheaper, faster and more accessible

Name: DragGAN

Owner: XingangPan

Stars: 33710

Repository: <https://github.com/XingangPan/DragGAN>

Description: Official Code for DragGAN (SIGGRAPH 2023)

Name: open-interpreter

Owner: KillianLucas

Stars: 33035

Repository: <https://github.com/KillianLucas/open-interpreter>

Description: OpenAI's Code Interpreter in your terminal, running locally

Name: XX-Net

Owner: XX-net

Stars: 32300

Repository: <https://github.com/XX-net/XX-Net>

Description: A proxy tool to bypass GFW.  
 Name: MockingBird  
 Owner: babysor  
 Stars: 31783  
 Repository: <https://github.com/babysor/MockingBird>  
 Description: AI : 5                      Clone a voice in 5 seconds to generate arbitrary speech in real  
 Name: HanLP  
 Owner: hankcs  
 Stars: 30751  
 Repository: <https://github.com/hankcs/HanLP>  
 Description: Natural Language Processing for the next decade. Tokenization, Part-of-Speech T  
 Name: ray  
 Owner: ray-project  
 Stars: 28605  
 Repository: <https://github.com/ray-project/ray>  
 Description: Ray is a unified framework for scaling AI and Python applications. Ray consists  
 Name: ItChat  
 Owner: littlecodersh  
 Stars: 24369  
 Repository: <https://github.com/littlecodersh/ItChat>  
 Description: A complete and graceful API for Wechat.  
 Name: hosts  
 Owner: StevenBlack  
 Stars: 24161  
 Repository: <https://github.com/StevenBlack/hosts>  
 Description: Consolidating and extending hosts files from several well-curated sources. Opt  
 Name: dash  
 Owner: plotly  
 Stars: 19624  
 Repository: <https://github.com/plotly/dash>  
 Description: Data Apps & Dashboards for Python. No JavaScript Required.  
 Name: chatgpt-on-wechat  
 Owner: zhayujie  
 Stars: 17214  
 Repository: <https://github.com/zhayujie/chatgpt-on-wechat>  
 Description: Wechat robot based on ChatGPT, which using OpenAI api and itchat library.  
 Name: Hitomi-Downloader  
 Owner: KurtBestor  
 Stars: 17151  
 Repository: <https://github.com/KurtBestor/Hitomi-Downloader>  
 Description: :cake: Desktop utility to download images/videos/music/text from various websites  
 Name: recommenders  
 Owner: recommenders-team



Stars: 16663  
 Repository: <https://github.com/recommenders-team/recommenders>  
 Description: Best Practices on Recommendation Systems  
 Name: loguru  
 Owner: Delgan  
 Stars: 16585  
 Repository: <https://github.com/Delgan/loguru>  
 Description: Python logging made (stupidly) simple  
 Name: awesome-oss-alternatives  
 Owner: RunaCapital  
 Stars: 14225  
 Repository: <https://github.com/RunaCapital/awesome-oss-alternatives>  
 Description: Awesome list of open-source startup alternatives to well-known SaaS products  
 Name: learn\_python3\_spider  
 Owner: wistbean  
 Stars: 13936  
 Repository: [https://github.com/wistbean/learn\\_python3\\_spider](https://github.com/wistbean/learn_python3_spider)  
 Description: python 0 1 python APP fiddler mitmproxy requests BeautifulSoup  
 Name: mlc-llm  
 Owner: mlc-ai  
 Stars: 13821  
 Repository: <https://github.com/mlc-ai/mlc-llm>  
 Description: Enable everyone to develop, optimize and deploy AI models natively on everyone's  
 Name: mackup  
 Owner: lra  
 Stars: 13780  
 Repository: <https://github.com/lra/mackup>  
 Description: Keep your application settings in sync (OS X/Linux)  
 Name: ChuanhuChatGPT  
 Owner: GaiZhenbiao  
 Stars: 13206  
 Repository: <https://github.com/GaiZhenbiao/ChuanhuChatGPT>  
 Description: GUI for ChatGPT API and many LLMs. Supports agents, file-based QA, GPT finetuning  
 Name: searx  
 Owner: searx  
 Stars: 13193  
 Repository: <https://github.com/searx/searx>  
 Description: Privacy-respecting metasearch engine  
 Name: PySimpleGUI  
 Owner: PySimpleGUI  
 Stars: 12208  
 Repository: <https://github.com/PySimpleGUI/PySimpleGUI>  
 Description: Launched in 2018. It's 2023 and PySimpleGUI is actively developed & supported.

Name: redis-py  
 Owner: redis  
 Stars: 11878  
 Repository: <https://github.com/redis/redis-py>  
 Description: Redis Python Client

Name: pelican  
 Owner: getpelican  
 Stars: 11872  
 Repository: <https://github.com/getpelican/pelican>  
 Description: Static site generator that supports Markdown and reST syntax. Powered by Python

Name: awesome-aws  
 Owner: donnemartin  
 Stars: 11799  
 Repository: <https://github.com/donnemartin/awesome-aws>  
 Description: A curated list of awesome Amazon Web Services (AWS) libraries, open source repos

Name: numba  
 Owner: numba  
 Stars: 9026  
 Repository: <https://github.com/numba/numba>  
 Description: NumPy aware dynamic Python compiler using LLVM

Name: kedro  
 Owner: kedro-org  
 Stars: 8981  
 Repository: <https://github.com/kedro-org/kedro>  
 Description: Kedro is a toolbox for production-ready data science. It uses software engineering

Name: OpenChatKit  
 Owner: togethercomputer  
 Stars: 8928  
 Repository: <https://github.com/togethercomputer/OpenChatKit>  
 Description: None

Name: Python  
 Owner: injetlee  
 Stars: 8825  
 Repository: <https://github.com/injetlee/Python>  
 Description: Python excel

Name: Reinforcement-learning-with-tensorflow  
 Owner: MorvanZhou  
 Stars: 8362  
 Repository: <https://github.com/MorvanZhou/Reinforcement-learning-with-tensorflow>  
 Description: Simple Reinforcement learning tutorials, Python AI

## Visualizing repositories using Plotly

```
import requests

from plotly.graph_objs import Bar
from plotly import offline

# make an API call and store the response
url = 'https://api.github.com/search/repositories?q=language:python&sort=stars'
headers = {'Accept': 'application/vnd.github.v3+json'} #uses specific version
r = requests.get(url, headers = headers) #using requests to make a call to API
print(f"Status code: {r.status_code}")

# process results
response_dict = r.json()
repo_dicts = response_dict['items']
repo_names, stars = [], [] #empty lists
for repo_dict in repo_dicts:
    repo_names.append(repo_dict['name'])
    stars.append(repo_dict['stargazers_count'])

# make visualization
data = [{
    'type': 'bar',
    'x': repo_names,
    'y': stars,
}]
my_layout = {
    'title': 'Most-starred python projects in Github',
    'xaxis': {'title': 'Repository'},
    'yaxis': {'title': 'Stars'},
}

fig = {'data': data, 'layout': my_layout}
offline.plot(fig, filename= 'python_repos.html')
```

Status code: 200

'python\_repos.html'

## Refining plotly charts

```
# make visualization
## modifying data
data_1 = [{
    'type': 'bar',
    'x': repo_names,
    'y': stars,
    'marker': {
        'color' : 'rgb(255,0,0)',
        'line' : {'width' : 1.5, 'color' : 'rgb(255,0,1)'}
    },
    'opacity' : 0.6,
}]
my_layout = {
    'title': 'Most-starred python projects in Github',
    'xaxis': {'title': 'Repository'},
    'yaxis': {'title' : 'Stars'},
}

fig = {'data': data_1, 'layout': my_layout}
offline.plot(fig, filename= 'python_repos_1.html')
```

'python\_repos\_1.html'

```
# make visualization
data_1 = [{
    'type': 'bar',
    'x': repo_names,
    'y': stars,
    'marker': {
        'color' : 'rgb(255,0,0)',
        'line' : {'width' : 1.5, 'color' : 'rgb(255,0,1)'}
    },
    'opacity' : 0.6,
}]
## modifying layout
my_layout_1 = {
    'title': 'Most-starred python projects in Github',
    'titlefont': {'size': 28},
    'xaxis': {
```

```

        'title': 'Repository',
        'titlefont' : {'size': 24},
        'tickfont' : {'size': 14},
    },
    'yaxis': {
        'title' : 'Stars',
        'titlefont' : {'size' : 24},
        'tickfont' : {'size' : 14},
    },
}

fig = {'data': data_1, 'layout': my_layout_1}
offline.plot(fig, filename= 'python_repos_2.html')

```

'python\_repos\_2.html'

## Adding Custom Tooltips

```

# process results
response_dict = r.json()
repo_dicts = response_dict['items']
repo_names, stars, labels = [], [], [] #empty lists
for repo_dict in repo_dicts:
    repo_names.append(repo_dict['name'])
    stars.append(repo_dict['stargazers_count'])

    owner = repo_dict['owner']['login']
    description = repo_dict['description']
    label = f"{owner}<br />{description}"
    labels.append(label)

# make visualization
data = [{
    'type': 'bar',
    'x': repo_names,
    'y': stars,
    'hovertext': labels,
    'marker': {
        'color' : 'rgb(250,0,0)',
        'line' : {'width' : 1.5, 'color' : 'rgb(255,0,1)'}
    }
}

```

```

    },
    'opacity' : 0.6,
}]
my_layout = {
    'title': 'Most-starred python projects in Github',
    'xaxis': {'title': 'Repository'},
    'yaxis': {'title' : 'Stars'},
}

fig = {'data': data, 'layout': my_layout}
offline.plot(fig, filename= 'python_repos.html')

```

'python\_repos.html'

## Adding clickable links

```

# process results
response_dict = r.json()
repo_dicts = response_dict['items']
repo_links, stars, labels = [], [], [] #empty lists
for repo_dict in repo_dicts:
    repo_name = repo_dict['name']
    repo_names.append(repo_dict['name'])
    repo_url = repo_dict['html_url']
    repo_link = f"<a href='{repo_url}'>{repo_name}</a>"
    repo_links.append(repo_link)
    stars.append(repo_dict['stargazers_count'])

    owner = repo_dict['owner']['login']
    description = repo_dict['description']
    label = f"{owner}<br />{description}"
    labels.append(label)

# make visualization
data = [{
    'type': 'bar',
    'x': repo_names,
    'y': stars,
    'hovertext': labels,
    'marker': {

```

```

        'color' : 'rgb(250,0,0)',
        'line' : {'width' : 1.5, 'color' : 'rgb(255,0,1)'}
    },
    'opacity' : 0.6,
}]
my_layout = {
    'title': 'Most-starred python projects in Github',
    'xaxis': {'title': 'Repository'},
    'yaxis': {'title' : 'Stars'},
}

fig = {'data': data, 'layout': my_layout}
offline.plot(fig, filename= 'python_repos.html')

```

'python\_repos.html'

```

# improved version
import plotly.graph_objs as go
from plotly.offline import plot as offline_plot

# Extract data from the JSON response
response_dict = r.json()
repo_dicts = response_dict['items']

# Initialize empty lists
repo_names, repo_links, stars, labels = [], [], [], []

# Process each repository in the response
for repo_dict in repo_dicts:
    repo_name = repo_dict['name']
    repo_url = repo_dict['html_url']
    repo_link = f"<a href='{repo_url}' target='_blank'>{repo_name}</a>"

    # Append data to lists
    repo_names.append(repo_name)
    repo_links.append(repo_link)
    stars.append(repo_dict['stargazers_count'])

    owner = repo_dict['owner']['login']
    description = repo_dict['description']
    label = f"{owner}<br />{description}"

```

```

        labels.append(label)

# Create visualization data
data = [{
    'type': 'bar',
    'x': repo_links, # Use repo_links for clickable links in the chart
    'y': stars,
    'hovertext': labels,
    'marker': {
        'color': 'rgb(250, 0, 0)',
        'line': {'width': 1.5, 'color': 'rgb(255, 0, 1)'}
    },
    'opacity': 0.6,
}]

# Configure layout
my_layout = {
    'title': 'Most-starred Python projects on GitHub',
    'xaxis': {'title': 'Repository'},
    'yaxis': {'title': 'Stars'},
}

# Create figure
fig = go.Figure(data=data, layout=my_layout)

# Save the interactive chart to an HTML file
offline_plot(fig, filename='python_repos.html')

```

'python\_repos.html'

## The Hacker News API

- contains articles about programming and technology (<http://news.ycombinator.com/>)

```

import requests
import json

# make an API call, and store the response

url = 'https://hacker-news.firebaseio.com/v0/item/19155826.json'

```



```

r = requests.get(url)
print (f"Status code: {r.status_code}")

# Explore data structure
#filename = "E:\\machine learning projects\\readable_hn_data.json"
#with open(filename, encoding = 'utf-8') as f:
#    all_eq_data = json.load(f)

# opening the readable file that we just created

readable_file = 'E:\\machine learning projects\\readable_hn_data.json'
with open(readable_file, 'r', encoding = 'utf-8') as f:
    content = f.read()
    print(content)

```

Status code: 200

```

{
  "by": "jimktrains2",
  "descendants": 221,
  "id": 19155826,
  "kids": [
    19156572,
    19158857,
    19156773,
    19157251,
    19156415,
    19159820,
    19157154,
    19156385,
    19156489,
    19158522,
    19156755,
    19156974,
    19158319,
    19157034,
    19156935,
    19158935,
    19157531,
    19158638,
    19156466,
    19156758,
    19156565,

```

```

        19156498,
        19156335,
        19156041,
        19156704,
        19159047,
        19159127,
        19156217,
        19156375,
        19157945
    ],
    "score": 728,
    "time": 1550085414,
    "title": "Nasa\u2019s Mars Rover Opportunity Concludes a 15-Year Mission",
    "type": "story",
    "url": "https://www.nytimes.com/2019/02/13/science/mars-opportunity-rover-dead.html"
}

```

## Analysing all the top articles from Hackernews

```

from operator import itemgetter

import requests

# make an API and store the response
url = 'https://hacker-news.firebaseio.com/v0/topstories.json'
r = requests.get(url)
print(f"Status code: {r.status_code}")

#process information about each submission
submission_ids = r.json()
submission_dicts = []

for submission_id in submission_ids[:30]:
    #make a seperate API call for each submission
    url = f"https://hacker-news.firebaseio.com/v0/item/{submission_id}.json"
    r = requests.get(url)
    print(f"id: {submission_id}/tstatus: {r.status_code}")
    response_dict = r.json()

    #build a dictionary for each article
    submission_dict = {

```

```

        'title' : response_dict['title'],
        'hn_link' : f"http://news.ycombinator.com/item?id={submission_id}",
        'comments': response_dict['descendants'],
    }
    submission_dicts.append(submission_dict)

submission_dicts = sorted(submission_dicts, key=itemgetter('comments'), reverse= True)

for submission_dict in submission_dicts:
    print(f"\nTitle: {submission_dict['title']}")
    print(f"Discussion link: {submission_dict['hn_link']}")
    print(f"Comments: {submission_dict['comments']}")

```

Status code: 200  
id: 38290145/tstatus: 200

Title: The real realtime preemption end game  
Discussion link: <http://news.ycombinator.com/item?id=38290145>  
Comments: 163  
id: 38292553/tstatus: 200

Title: Migrating to OpenTelemetry  
Discussion link: <http://news.ycombinator.com/item?id=38292553>  
Comments: 27

Title: Migrating to OpenTelemetry  
Discussion link: <http://news.ycombinator.com/item?id=38292553>  
Comments: 27  
id: 38295179/tstatus: 200

Title: Show HN: Tiny LLMs - Browser-based private AI models for a wide array of tasks  
Discussion link: <http://news.ycombinator.com/item?id=38295179>  
Comments: 0

Title: Show HN: Tiny LLMs - Browser-based private AI models for a wide array of tasks  
Discussion link: <http://news.ycombinator.com/item?id=38295179>  
Comments: 0

Title: Show HN: Tiny LLMs - Browser-based private AI models for a wide array of tasks  
Discussion link: <http://news.ycombinator.com/item?id=38295179>

Comments: 0

id: 38290613/tstatus: 200

Title: From email to phone number, a new OSINT approach (2019)

Discussion link: <http://news.ycombinator.com/item?id=38290613>

Comments: 76

Title: From email to phone number, a new OSINT approach (2019)

Discussion link: <http://news.ycombinator.com/item?id=38290613>

Comments: 76

Title: From email to phone number, a new OSINT approach (2019)

Discussion link: <http://news.ycombinator.com/item?id=38290613>

Comments: 76

Title: From email to phone number, a new OSINT approach (2019)

Discussion link: <http://news.ycombinator.com/item?id=38290613>

Comments: 76

id: 38291139/tstatus: 200

Title: Emu Video and Emu Edit, our latest generative AI research milestones

Discussion link: <http://news.ycombinator.com/item?id=38291139>

Comments: 15

Title: Emu Video and Emu Edit, our latest generative AI research milestones

Discussion link: <http://news.ycombinator.com/item?id=38291139>

Comments: 15

Title: Emu Video and Emu Edit, our latest generative AI research milestones

Discussion link: <http://news.ycombinator.com/item?id=38291139>

Comments: 15

Title: Emu Video and Emu Edit, our latest generative AI research milestones

Discussion link: <http://news.ycombinator.com/item?id=38291139>

Comments: 15

Title: Emu Video and Emu Edit, our latest generative AI research milestones

Discussion link: <http://news.ycombinator.com/item?id=38291139>

Comments: 15

id: 38291199/tstatus: 200

Title: Zimbra 0-day used to steal email data from government organizations

Discussion link: <http://news.ycombinator.com/item?id=38291199>

Comments: 13

Title: Zimbra 0-day used to steal email data from government organizations

Discussion link: <http://news.ycombinator.com/item?id=38291199>

Comments: 13

Title: Zimbra 0-day used to steal email data from government organizations

Discussion link: <http://news.ycombinator.com/item?id=38291199>

Comments: 13

Title: Zimbra 0-day used to steal email data from government organizations

Discussion link: <http://news.ycombinator.com/item?id=38291199>

Comments: 13

Title: Zimbra 0-day used to steal email data from government organizations

Discussion link: <http://news.ycombinator.com/item?id=38291199>

Comments: 13

Title: Zimbra 0-day used to steal email data from government organizations

Discussion link: <http://news.ycombinator.com/item?id=38291199>

Comments: 13

id: 38295638/tstatus: 200

KeyError: 'descendants'

```
## improved code
```

```
from operator import itemgetter
```

```
import requests
```

```
# Make an API call to get the top story IDs
```

```
url_top_stories = 'https://hacker-news.firebaseio.com/v0/topstories.json'
```

```
response_top_stories = requests.get(url_top_stories)
```

```
print(f"Status code: {response_top_stories.status_code}")
```

```
# Process information about each submission
```

```
submission_ids = response_top_stories.json()
```

```
submission_dicts = []
```

```
# Make a separate API call for each submission and store relevant information
```

```
for submission_id in submission_ids[:30]:
```

```
    url_submission = f"https://hacker-news.firebaseio.com/v0/item/{submission_id}.json"
```

```

response_submission = requests.get(url_submission)
print(f"id: {submission_id}\tstatus: {response_submission.status_code}")

# Check if the API call was successful
if response_submission.status_code == 200:
    submission_dict = {
        'title': response_submission.json().get('title', 'N/A'),
        'hn_link': f"http://news.ycombinator.com/item?id={submission_id}",
        'comments': response_submission.json().get('descendants', 0),
    }
    submission_dicts.append(submission_dict)

# Sort the submission dictionaries based on the number of comments in descending order
submission_dicts = sorted(submission_dicts, key=itemgetter('comments'), reverse=True)

# Print information about each submission
for submission_dict in submission_dicts:
    print(f"\nTitle: {submission_dict['title']}")
    print(f"Discussion link: {submission_dict['hn_link']}")
    print(f"Comments: {submission_dict['comments']}")

```

Status code: 200

```

id: 38290145      status: 200
id: 38295179      status: 200
id: 38292553      status: 200
id: 38290613      status: 200
id: 38291139      status: 200
id: 38291199      status: 200
id: 38295638      status: 200
id: 38291015      status: 200
id: 38276418      status: 200
id: 38294723      status: 200
id: 38295524      status: 200
id: 38294569      status: 200
id: 38292102      status: 200
id: 38294203      status: 200
id: 38289327      status: 200
id: 38291735      status: 200
id: 38291880      status: 200
id: 38291427      status: 200
id: 38294623      status: 200
id: 38293817      status: 200

```

id: 38276727      status: 200  
id: 38287257      status: 200  
id: 38275698      status: 200  
id: 38269866      status: 200  
id: 38295819      status: 200  
id: 38288980      status: 200  
id: 38288130      status: 200  
id: 38287299      status: 200  
id: 38291399      status: 200  
id: 38288743      status: 200

Title: Privacy is priceless, but Signal is expensive  
Discussion link: <http://news.ycombinator.com/item?id=38291427>  
Comments: 557

Title: I think I need to go lie down  
Discussion link: <http://news.ycombinator.com/item?id=38288130>  
Comments: 391

Title: A failed AI girlfriend product, and my lessons  
Discussion link: <http://news.ycombinator.com/item?id=38287299>  
Comments: 198

Title: Smart drugs reduce quality of effort, and slow decision-making  
Discussion link: <http://news.ycombinator.com/item?id=38287257>  
Comments: 173

Title: The real realtime preemption end game  
Discussion link: <http://news.ycombinator.com/item?id=38290145>  
Comments: 163

Title: Operating on a minimal two-core Postgres instance: Query optimization insights  
Discussion link: <http://news.ycombinator.com/item?id=38276727>  
Comments: 128

Title: Sweden Gov Announces 'Massive Expansion' of Nuclear Energy  
Discussion link: <http://news.ycombinator.com/item?id=38291015>  
Comments: 122

Title: Moving from AWS to Bare-Metal saved us 230k\$ /yr  
Discussion link: <http://news.ycombinator.com/item?id=38294569>  
Comments: 80

Title: From email to phone number, a new OSINT approach (2019)  
Discussion link: <http://news.ycombinator.com/item?id=38290613>  
Comments: 76

Title: Why thinking hard makes us feel tired  
Discussion link: <http://news.ycombinator.com/item?id=38294723>  
Comments: 59

Title: Ransomware Group Files SEC Complaint over Victim's Failure Disclose Data Breach  
Discussion link: <http://news.ycombinator.com/item?id=38291399>  
Comments: 45

Title: Hackers know everything is securities fraud  
Discussion link: <http://news.ycombinator.com/item?id=38293817>  
Comments: 40

Title: Frutiger Aero  
Discussion link: <http://news.ycombinator.com/item?id=38276418>  
Comments: 38

Title: Printed robots with bones, ligaments, and tendons  
Discussion link: <http://news.ycombinator.com/item?id=38288980>  
Comments: 38

Title: You don't need a CRDT to build a collaborative experience  
Discussion link: <http://news.ycombinator.com/item?id=38289327>  
Comments: 35

Title: std::source\_location Is Broken  
Discussion link: <http://news.ycombinator.com/item?id=38292102>  
Comments: 28

Title: Migrating to OpenTelemetry  
Discussion link: <http://news.ycombinator.com/item?id=38292553>  
Comments: 27

Title: Hello World on the GPU (2019)  
Discussion link: <http://news.ycombinator.com/item?id=38275698>  
Comments: 22

Title: Emu Video and Emu Edit, our latest generative AI research milestones  
Discussion link: <http://news.ycombinator.com/item?id=38291139>  
Comments: 15



Title: Federated finetuning of Whisper on Raspberry Pi 5  
Discussion link: <http://news.ycombinator.com/item?id=38294203>  
Comments: 15

Title: Zimbra 0-day used to steal email data from government organizations  
Discussion link: <http://news.ycombinator.com/item?id=38291199>  
Comments: 13

Title: AI-Exploits: Repo of multiple unauthenticated RCEs in AI tools  
Discussion link: <http://news.ycombinator.com/item?id=38291880>  
Comments: 10

Title: Serverless development experience for embedded computer vision  
Discussion link: <http://news.ycombinator.com/item?id=38288743>  
Comments: 8

Title: The CWEB System of Structured Documentation  
Discussion link: <http://news.ycombinator.com/item?id=38291735>  
Comments: 2

Title: OCapN, Interoperable Capabilities over the Network  
Discussion link: <http://news.ycombinator.com/item?id=38295524>  
Comments: 1

Title: A floating solar-powered device produces clean water and clean fuel  
Discussion link: <http://news.ycombinator.com/item?id=38269866>  
Comments: 1

Title: Microsoft support 'cracks' Windows for customer after activation fails  
Discussion link: <http://news.ycombinator.com/item?id=38295819>  
Comments: 1

Title: Show HN: Tiny LLMs - Browser-based private AI models for a wide array of tasks  
Discussion link: <http://news.ycombinator.com/item?id=38295179>  
Comments: 0

Title: In-person YC Startup Tech Talk and hiring mixer on 12/4 in SF  
Discussion link: <http://news.ycombinator.com/item?id=38295638>  
Comments: 0

Title: Planning for Unplanned Work in Linear  
Discussion link: <http://news.ycombinator.com/item?id=38294623>

Comments: 0