Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> """render a miniforge releases page"""

'render a miniforge releases page'

>>> import jinja2

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ModuleNotFoundError: No module named 'jinja2'

>>> from pathlib import Path

>>> import datetime

>>> import sys

>>> import requests\_cache

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ModuleNotFoundError: No module named 'requests\_cache'

>>>

>>> HERE = Path(\_\_file\_\_).parent

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name '\_\_file\_\_' is not defined

>>> BUILD = HERE.parent / "build"

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'HERE' is not defined

>>> DOCS = BUILD / "docs"

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'BUILD' is not defined

>>>

>>> if not DOCS.exists():

... DOCS.mkdir(parents=True)

...

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'DOCS' is not defined

>>> # TODO: handle pagination

>>> BASE\_URL = "https://api.github.com/repos/conda-forge/miniforge/releases?per\_page=100"

>>> ENV = jinja2.Environment(loader=jinja2.FileSystemLoader([HERE / "templates"]))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'jinja2' is not defined

>>>

>>>

>>> def get\_releases():

... """use the GitHub API to fetch release information"""

... s = requests\_cache.CachedSession(str(BUILD / "cache"))

... releases = s.get(BASE\_URL).json()

...

>>> new\_releases = []

File "<stdin>", line 1

new\_releases = []

IndentationError: unexpected indent

>>>

>>> for release in releases:

File "<stdin>", line 1

for release in releases:

IndentationError: unexpected indent

>>> if release["draft"] or release["prerelease"]:

File "<stdin>", line 1

if release["draft"] or release["prerelease"]:

IndentationError: unexpected indent

>>> continue

File "<stdin>", line 1

continue

IndentationError: unexpected indent

>>> new\_assets = []

File "<stdin>", line 1

new\_assets = []

IndentationError: unexpected indent

>>> for asset in release["assets"]:

File "<stdin>", line 1

for asset in release["assets"]:

IndentationError: unexpected indent

>>> name = asset["name"]

File "<stdin>", line 1

name = asset["name"]

IndentationError: unexpected indent

>>> if "sha256" in name:

File "<stdin>", line 1

if "sha256" in name:

IndentationError: unexpected indent

>>> continue

File "<stdin>", line 1

continue

IndentationError: unexpected indent

>>> if release["tag\_name"] not in name:

File "<stdin>", line 1

if release["tag\_name"] not in name:

IndentationError: unexpected indent

>>> continue

File "<stdin>", line 1

continue

IndentationError: unexpected indent

>>> if release["tag\_name"] in asset["name"]:

File "<stdin>", line 1

if release["tag\_name"] in asset["name"]:

IndentationError: unexpected indent

>>> asset["\_variant"], os\_plat = asset["name"].split(

File "<stdin>", line 1

asset["\_variant"], os\_plat = asset["name"].split(

IndentationError: unexpected indent

>>> f"""-{release["tag\_name"]}-"""

File "<stdin>", line 1

f"""-{release["tag\_name"]}-"""

IndentationError: unexpected indent

>>> )

File "<stdin>", line 1

)

IndentationError: unexpected indent

>>> asset["\_os"], asset["\_arch"] = os\_plat.split(".")[0].split("-")

File "<stdin>", line 1

asset["\_os"], asset["\_arch"] = os\_plat.split(".")[0].split("-")

IndentationError: unexpected indent

>>> else:

File "<stdin>", line 1

else:

IndentationError: unexpected indent

>>> raise ValueError(f"Couldn't variant for {name}")

File "<stdin>", line 1

raise ValueError(f"Couldn't variant for {name}")

IndentationError: unexpected indent

>>> asset["\_sha256"] = s.get(

File "<stdin>", line 1

asset["\_sha256"] = s.get(

IndentationError: unexpected indent

>>> f"""{asset["browser\_download\_url"]}.sha256"""

File "<stdin>", line 1

f"""{asset["browser\_download\_url"]}.sha256"""

IndentationError: unexpected indent

>>> ).text.split(" ")[0]

File "<stdin>", line 1

).text.split(" ")[0]

IndentationError: unexpected indent

>>> new\_assets += [asset]

File "<stdin>", line 1

new\_assets += [asset]

IndentationError: unexpected indent

>>> release["assets"] = new\_assets

File "<stdin>", line 1

release["assets"] = new\_assets

IndentationError: unexpected indent

>>> new\_releases += [release]

File "<stdin>", line 1

new\_releases += [release]

IndentationError: unexpected indent

>>> releases = new\_releases

File "<stdin>", line 1

releases = new\_releases

IndentationError: unexpected indent

>>> return releases

File "<stdin>", line 1

return releases

IndentationError: unexpected indent

>>>

>>>

>>> def render(releases):

... """render the release page HTML"""

... context = dict(

... title="Miniforge Releases", releases=releases, year=datetime.datetime.now().year

... )

... html = ENV.get\_template("all-releases.html").render(\*\*context)

...

>>> release\_html = DOCS / "all-releases" / "index.html"

File "<stdin>", line 1

release\_html = DOCS / "all-releases" / "index.html"

IndentationError: unexpected indent

>>>

>>> if not release\_html.parent.exists():

File "<stdin>", line 1

if not release\_html.parent.exists():

IndentationError: unexpected indent

>>> release\_html.parent.mkdir(parents=True)

File "<stdin>", line 1

release\_html.parent.mkdir(parents=True)

IndentationError: unexpected indent

>>>

>>> release\_html.write\_text(html, encoding="utf-8")

File "<stdin>", line 1

release\_html.write\_text(html, encoding="utf-8")

IndentationError: unexpected indent

>>>

>>>

>>> def main():

... """main entrypoint"""

... releases = get\_releases()

... render(releases)

... return 0

...

>>>

>>> if \_\_name\_\_ == "\_\_main\_\_":

... sys.exit(main())