Flask

web development,
one drop at a time

Using Flask to easily create simple dynamic web pages

Sarah Graves, East Asian observatory

Links!

- Flask documentation: http://flask.pocoo.org
- Flask snippets: http://flask.pocoo.org/snippets/
- Jinja2 templates: http://jinja.pocoo.org/
- Werkzeug: http://werkzeug.pocoo.org/
- Flask extensions: http://flask.pocoo.org/extensions/

What is Flask?

- a “microframework for Python based on Werkzeug, Jinja2 and good intentions”

- Primary maintainer: Armin Ronacher

- BSD licensed.

Uses of Flask at EAO

- Web interface to our data processing system (internal only): allows users to change the state of jobs to resolve errors and monitor progress

- Hedwig, our proposal and technical assent platform (written by Graham Bell). This has performed happily through more than 6 calls for proposals and TAC meetings now.

- SCUBA-2 Calibration database (public), which retrieves calibration values for a user's project from our database and returns the values and produces some plots of them.

- Pigwidgeon, a publication tracking web app for observatories.

Features of flask

- Built in server for testing and debugging. (doesn't scale well)
- Unicode based
- session/cookie support
- Fairly easy to deploy to a WSGI server for production.
- Well documented!
- Large number of extensions, including:
  - Login schemes: including specific logins and/or LDAP
  - sqlalchemy/DB connections.

The simplest Flask App

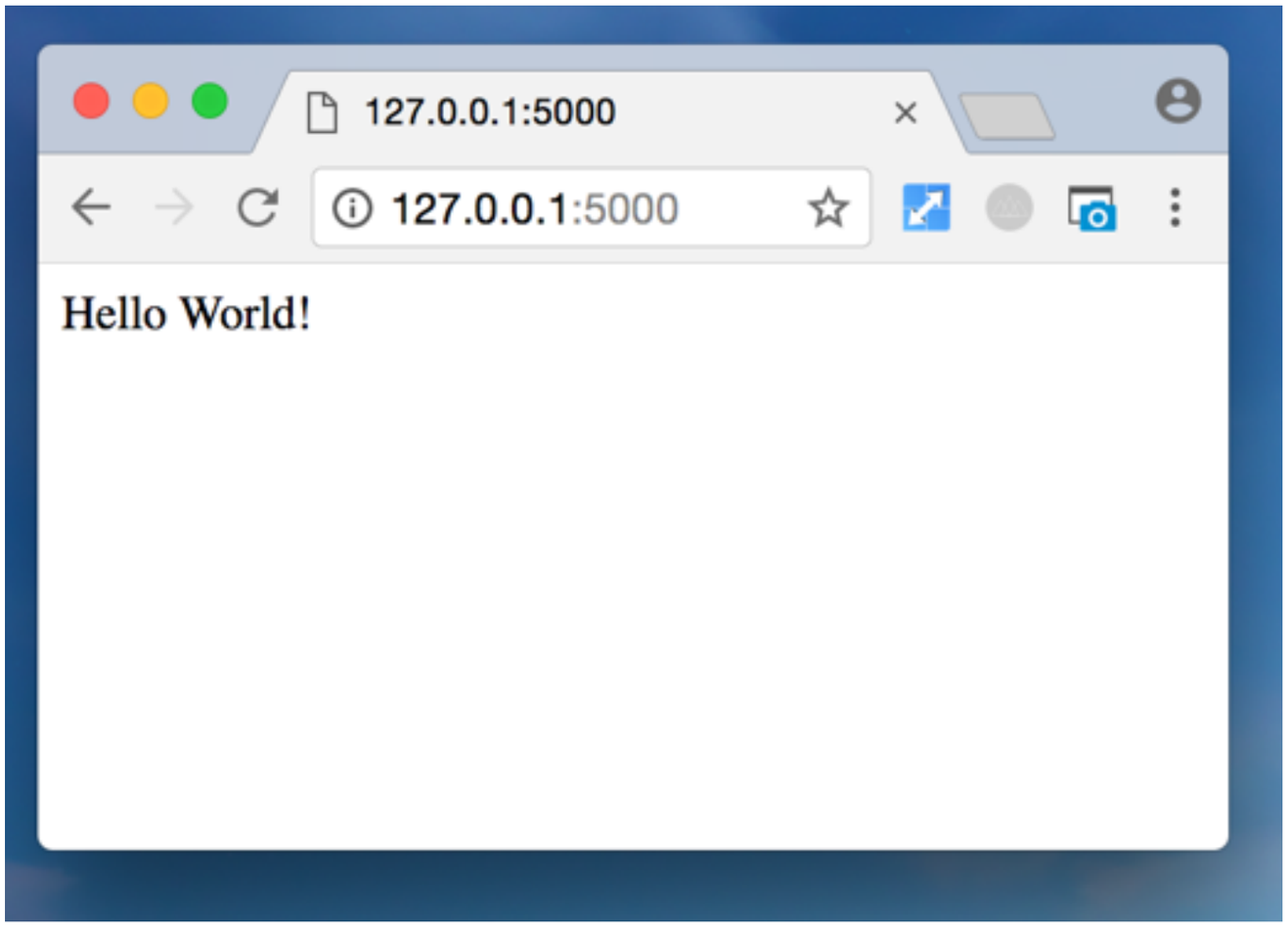
- Basic Hello world application from the documentation:

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"
```

```
$ pip install Flask
$ FLASK_APP=hello.py flask run
 * Running on http://localhost:5000/
```

The simplest Flask App

- This produces the web page:



A simple flask app

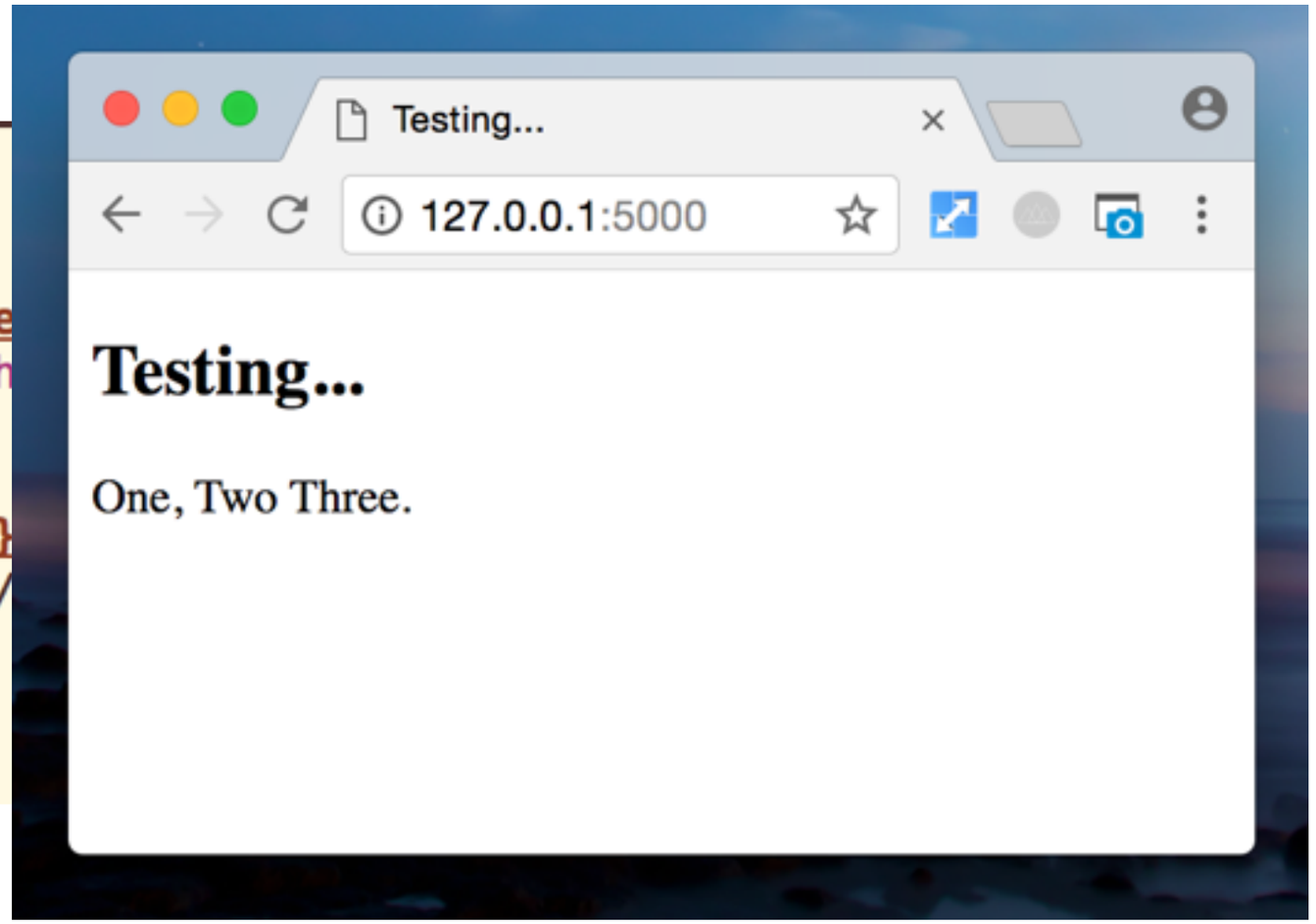
- What about a slightly more complicated one?
  - Templates!

```python
from flask import Flask, render_template
app = Flask(__name__)

@app.route("/")
def hello():
    pagetitle = 'Testing...'
    pageheading = 'Testing...'
    pagetext = 'One, Two Three.'
    return render_template("hello.html", pagetitle=pagetitle,
                           pageheading=pageheading,
                           pagetext=pagetext)
```

```html
<!DOCTYPE HTML>
<htm lang="en">
  <head>
    <title>{{pagetitle}}</title>
    <link rel="stylesheet" href="/static/style.css" type="text/css"/>
  </head>
  <body>
    <h2>{{pageheading}}</h2>
    <p> {{pagetext}}</p>
  </body>
</html>
```

```python
from flask import Flask, render_template
app = Flask(__name__)

@app.route("/")
def hello():
    pagetitle = 'Testing...'
    pageheading = 'Testing...'
    pagetext = 'One, Two Three.'
    return render_template("hello.html", pagetitle=pagetitle,
                           pageheading=pageheading,
                           pagetext=pagetext)
```

```
<!DOCTYPE HTML>
<htm lang="en">
  <head>
    <title>{{pagetitle
    <link rel="stylesh
  </head>
  <body>
    <h2>{{pageheading}
    <p> {{pagetext}}</
  </body>
</html>
```



A simple flask app

- What about a slightly more complicated one?
  - Templates!
  - Routes — easily change your pages URLs.

```
app = Flask(__name__)

@app.route("/")
def hello():
    pagetitle = 'Testing...'
    pageheading = 'Testing...'
    pagetext = 'One, Two Three.'
    return render_template("hello.html", pagetitle=pagetitle,
                                pageheading=pageheading,
                                pagetext=pagetext)
```

```
@app.route("/<obsid>/info")
def info(obsid):

    pagetext = "An information page summarising obsid={}".format(obsid)
    pagetitle = "{} summary".format(obsid)
    pageheading = pagetitle

    return render_template("hello.html", pagetitle=pagetitle,
                                pageheading=pageheading,
                                pagetext=pagetext,
                                )
```


```
itle,
def info(obsid):

    pagetext = "An information page summarising obsid={}".format(obsid)
    pagetitle = "{} summary".format(obsid)
    pageheading = pagetitle

    return render_template("hello.html", pagetitle=pagetitle,
                           pageheading=pageheading,
                           pagetext=pagetext,
                           )
```

```
<!DOCTYPE HTML>
<htm lang="en">
  <head>
    <title>{{pagetitle}}</title>
    <link rel="stylesheet" href="/static/style.css" type="text/css"/>
  </head>
  <body>
    <h2>{{pageheading}}</h2>
    <p> {{pagetext}}</p>
    <h2>Hello page</h2>
    <a href="{{url_for('hello')}}">Click here to see hello page</a>
  </body>
</html>
```

Debugging!

- Because sometime, somewhere, the 503 Internal Server Error will strike you…

- Debug mode for development runs the internal Flask server with automatic reloading on changes to the source code.

- If errors occur, dumps you into an interactive screen with the traceback and an interactive python terminal so you can find out what went wrong.

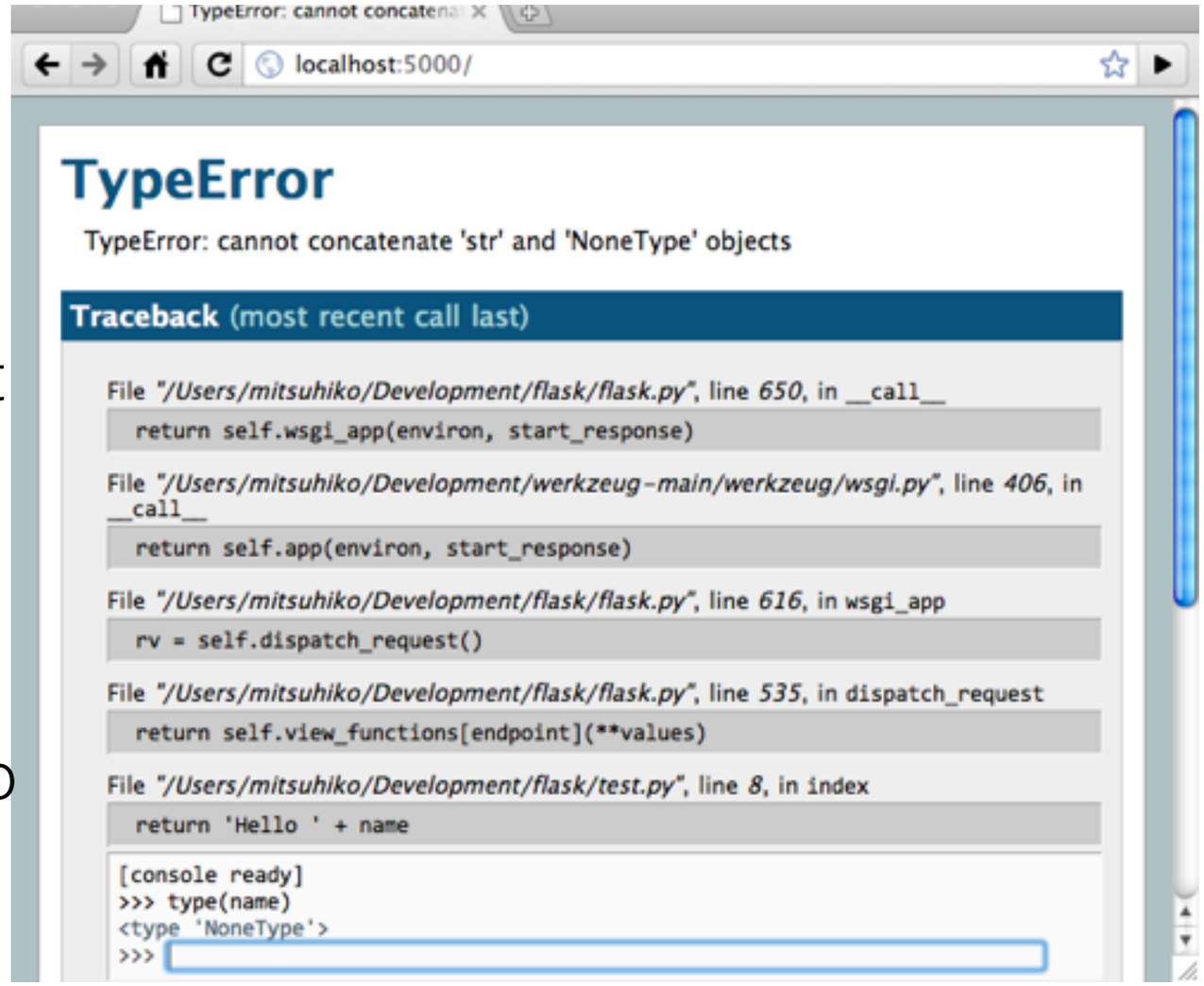


Image from Flask documentation

Jinja 2 templates

- Very full featured and powerful.
- Basic idea: write html, then put variables names in double curly braces:

```
<p> The python variable is {{variablename}}</p>
```

- More advanced features:
  - loops and ifs:

```
{% for x in object %}
<insert text here>
{% endfor %}
```

  - macros: write once, use again and again. Call like a function.
  - filters and tests:

Images

- Don’t need to write images to files on disk: can use sendfile method to create image when page is loaded.

- Create image with e.g. matplotlib

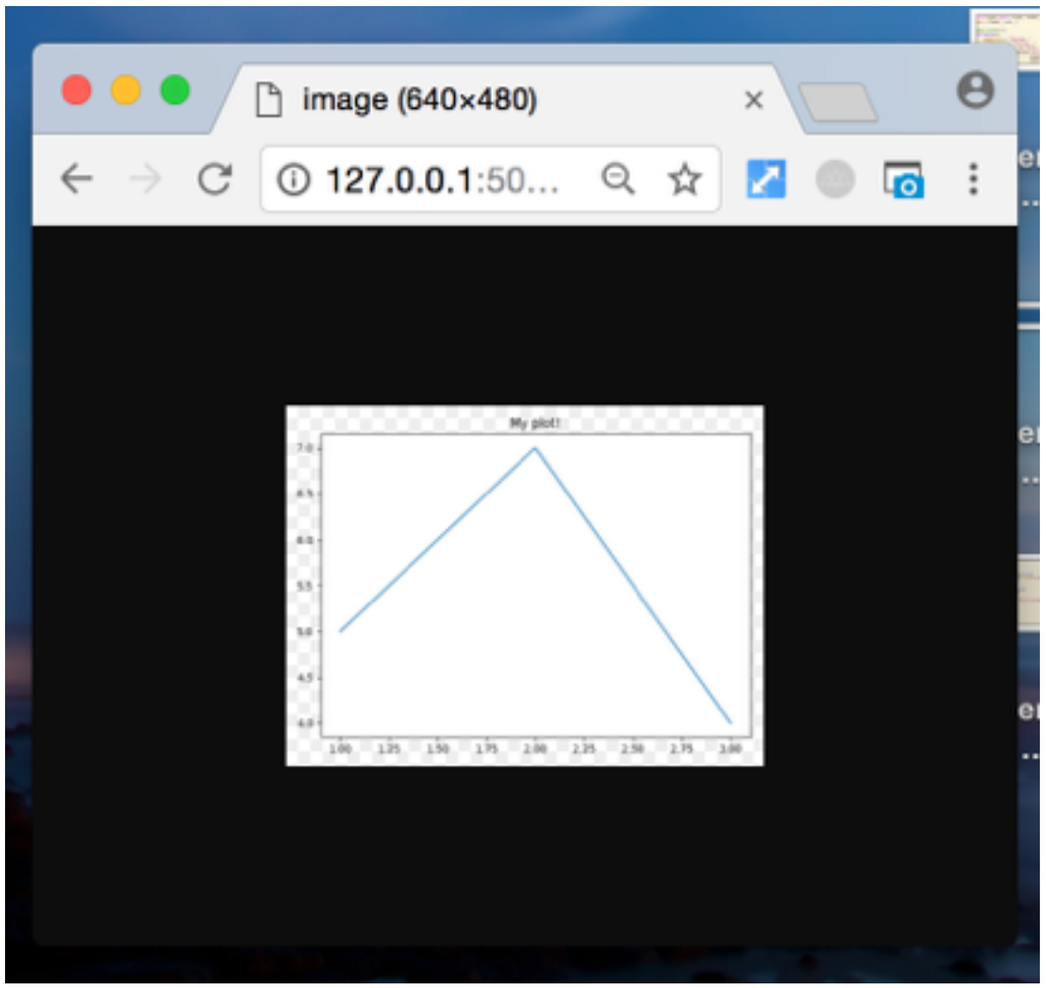
- Write that image to a text stream

- Create a route that returns a send_file object of that stream.

```
from flask import send_file
from matplotlib.backends.backend_agg import FigureCanvasAgg as F
from matplotlib.figure import Figure

@app.route("/image")
def create_image():
    fig = Figure()
    # Transparent background
    fig.patch.set_visible(False)
    ax = fig.add_subplot(111)
    ax.plot([1,2,3], [5,7,4.0])
    ax.set_title('My plot!')
    #Ensure all of image is in the plot
    fig.tight_layout()

    # Create image as stream.
    canvas = FigureCanvas(fig)
    img = BytesIO()
    canvas.print_png(img)
    img.seek(0)

    return send_file(img, mimetype='image/png')
```

Images

- Don’t need to write images to files on disk: can use sendfile method to create image when page is loaded.
  - Create image with e.g. matplotlib
  - Write that image to a text stream
  - Create a route that returns a send_file object of that stream.



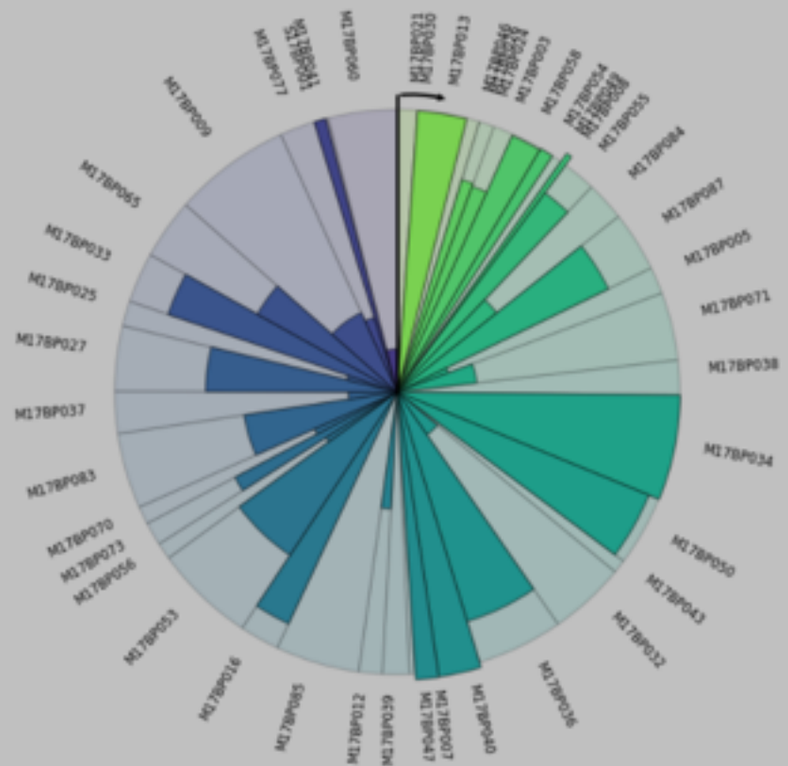
Summary of projects: semester=17B and queue=PI

Project constraints

Semester: 17B
Queue: PI
Pattern: Project pattern
FoP: OMP ID, Requires Sem.
Project IDs: one per line
Excluded Projects: one per line

Show details ☐
Show observing blocks ☑
Exclude completed projects ☐
Show MSB availability ☑
Submit

42.6% complete (444.4 / 1042.9 hrs)



Project list (priority order)

M17BP021 TETARENKOA	0.0%	Constraining Jet Formation and Evolution with Transient X-ray Binaries
M17BP030 BOWERG	0.0%	Probing Magnetized Accretion Flow around Sgr A* on 1-40 pc scales and on 1-10,000 Schwarzschild Radii with JCMT Polarimetry
M17BP013 CHAPMANS	99.5%	Completing the S2-Web survey
✗M17BP046 BASTIENP	0.0%	Testing Alignment of Carbonaceous Grains
M17BP026 KOCHP	0.0%	Magnetic Field and Rotational Motion in Proto-circumstellar Disk Formation
M17BP024 FUMAGAL-LIM	61.7%	Extended Quasar Nebulae as Nurseries of Massive Clusters
M17BP003 GEARW	60.0%	SCUBA2 Imaging of M33: Triangulum Galaxy
M17BP058 KWONW	98.9%	Magnetic Field Morphologies around Class 0 Young Stellar Objects
M17BP054 YOOH	100.0%	High sensitivity observations of variable source EC53
✗M17BP049 YANGB	0.0%	Large Particles in 3200 Phaethon: A Unique Opportunity
M17BP008 WHITEJ	107.4%	Measuring Emission from Stellar Atmospheres in Submillimeter/millimeter Wavelengths
M17BP055 ROSOTTIG	77.0%	Proto-planetary disc masses at the end of their lifetime