Intro:

-Created the python environment + project main file.
-pip install pyzmq (for messaging) and jupyter client
-import them + sys + JSON
- make a dummy json file for testing, will create a better one down the line.
- i created a **read_connection_file** function that takes a connection file path, and if the file is available and valid parses it and returns it.

Kernel Class:

- Started by making the initialize function which takes the connection file, and creates the sockets we use later for message sending/receiving
- Sockets:
  - Shell
  - IOPub
  - Stdin
  - Control
  - Heartbeat

I create the sockets using the custom function **setup_sockets**, this function creates the sockets then uses the connection info to assign each socket its port and binds it then returns the sockets so init can use it.

- Now i created **handle_heartbeat** function, the purpose of this function is just to recieve heartbeat pings and to send them back(using the hb_socket), just ensuring the kernel is responsive, for this purpose im gonnna run it on a seperate thread so its responsive and fast.

- Created 2 functions **hande_control_message** and **handle_shell_message**, each recieve thier messages on thier respective channels and break it down into parts.

TODO: add handling according to message type + at the start check if message is valid using connection key.

Main Loop:

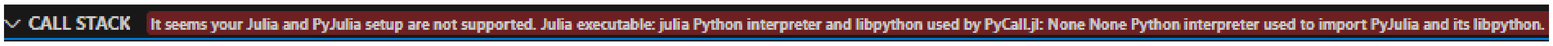
-Creates and starts the heartbeat thread
use **zmq.Poller** to monitor the sockets , currently only control and shell. for now it checks if any of the sockets got a message and if so send its to the correct message handler according to the socket.

-Next up i created **validate_message** function to validate message signatures and make sure the messages have been not tampered with by comparing the calculated signature and current one.

After that added the function to my previous **handle_control_message** and **handle_shell_message** functions.

-after seeing both functions, i decided to refactor them into a single function called **handle_message** since theyre doing the same thing just on different sockets. this should simplify the code and cut some of it.
Now i can start handling the different message types:
To start i make the **handle_kernel_info_request** function which should return my kernel info to the frontend. and deals with kernel info requests.
but to do that i need a function thats able to send responses back to the frontend. this means ill have to make a send_response function that does just that, takes the response, the socket its supposed to send on, and sends it.
while creating it, i also realized i needed to make a **sign_message** function, to sign the messages and create a signature before we send them so they're valid and secure,
- To make sure everything i did so far works its time to implement code execution , the general idea is using % magic commands, the user can pick either **Python** or **Julia** , %python , %julia. and then the functions should handle executing the code for now,

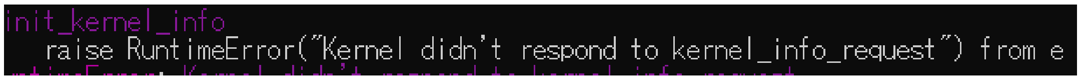
after identifying the language we send it to the correct execution engine . for now ill make it super basic/ barebones with not alot of functionality just to be able to run it and make sure everything is working correctly.
-To be able to use julia i have to install **Pyjulia ,** so install jill first then pyjulia then import it.
after installing and importing , ill have initialize it in my kernel init function aswell as adding the execution count variable which will be used soon.
after finishing my **handle_execute** function i need to debug, i already started but getting issues because i forgot to install julia on pc, doing that then will continue debugging.



```
CALL STACK  It seems your Julia and PyJulia setup are not supported. Julia executable: julia Python interpreter and libpython used by PyCall.jl: None None Python interpreter used to import PyJulia and its libpython.
```

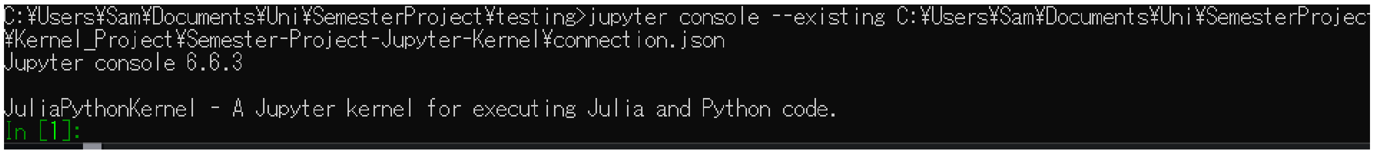
tried creating new environment and reinstalling everything.
uninstalled jill and used juliacall instead, seemes to fix the issues. changed my code accordingly.
-currently debugging alot of errors, one was a typo , another seems to be type mismatch etc...
-after looking at documentation again i realized that zmq messages have an extra parameter that i forgot to include, i have to redo some stuff in the message functions.
Notes for me : ** Currently to run it im using Running it on VScode then using jupyter console --existing C:\Users\Sam\Documents\Uni\SemesterProject\Kernel_Project\Semester-Project-Jupyter-Kernel\connection.json to run it.

-Kernel is currently failing because its expecting a response to kernel info request and not getting any.
ill create a function **handle_kernel_info_request** to handle it.



```
init_kernel_info
    raise RuntimeError("Kernel didn't respond to kernel_info_request") from e
```

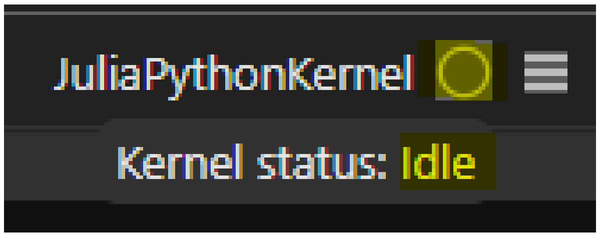
Kernel Loaded after hrs of debugging, progress!!!.



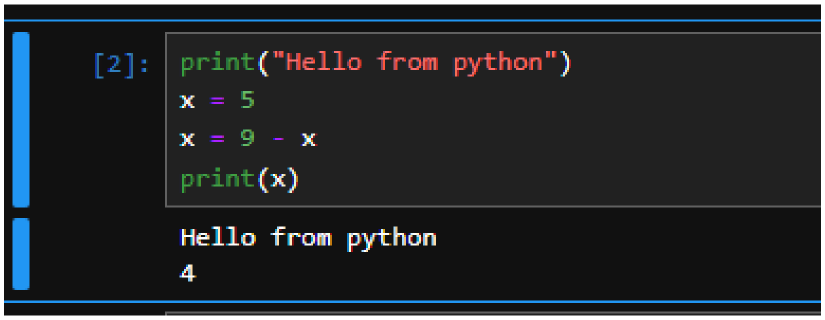
```
C:¥Users¥Sam¥Documents¥Uni¥SemesterProject¥testing>jupyter console --existing C:¥Users¥Sam¥Documents¥Uni¥SemesterProjec
¥Kernel_Project¥Semester-Project-Jupyter-Kernel¥connection.json
Jupyter console 6.6.3

JuliaPythonKernel - A Jupyter kernel for executing Julia and Python code.
In [1]:
```

i'll make a requirments.txt file now to make running the code easier. ill add to it more stuff as the project develops.
Now to work on code exectuion, i need to figure out what messages need to be sent /recieved during the whole proecess .
- going back to handle_execute_request, ill redo the function since its not working as intended , at the start we seperate the code, then increment the execution count variable, then we prepare our response message to iopub, we'll basically broadcast that the code is being exectued right now to the frontend.
after that we check the first line of the code, using it we can decide what language the code should be executed in, %julia for julia and %python for python, i set the default to python since its more useful.
Before going to code execute i make variables where ill capture the errors/prints/outputs.
Then we finally get to execution, use the appropriate engine to execute the code, then send output/error output to the IOpub channel as a response, and finally send a execute_reply request to the shell to tell it either that the code is working fine, or that theres an error.
- For now ill be making a kernel.JSON and trying to get my code to work in a vscode notebook to debug it later, so far its not working but ill keep trying different stuff.
- Changed the validate signature function.
- adding Busy - Idle status after kernel info request message. for easier use later on ill just make send_iopub_status function to handle the busy busy idle responses.
- ill make a handle_extra_messages function to handle non-important message requests and thier replies (usually empty)

-To get it to work on jupyter notebook it took alot of more work, since its relatively more complex than just getting it to work on jupyter console, but im happy that after all the debugging and new functions/function changing , the kernel now loads on jupyter notebook !, after connecting and getting the kernel info and other msg requests , it goes into idle mode waiting for actions .



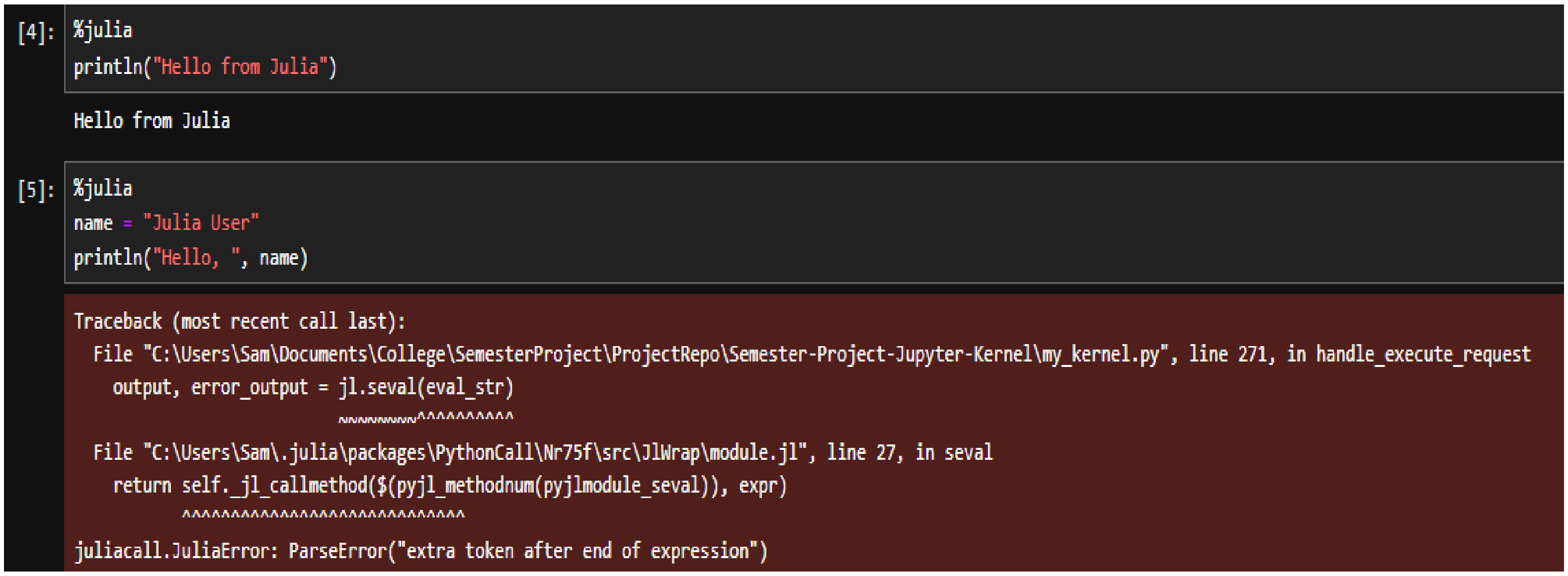
-Next up checking the simple code execution of the kernel.



```
[2]: print("Hello from python")
     x = 5
     x = 9 - x
     print(x)

     Hello from python
     4
```

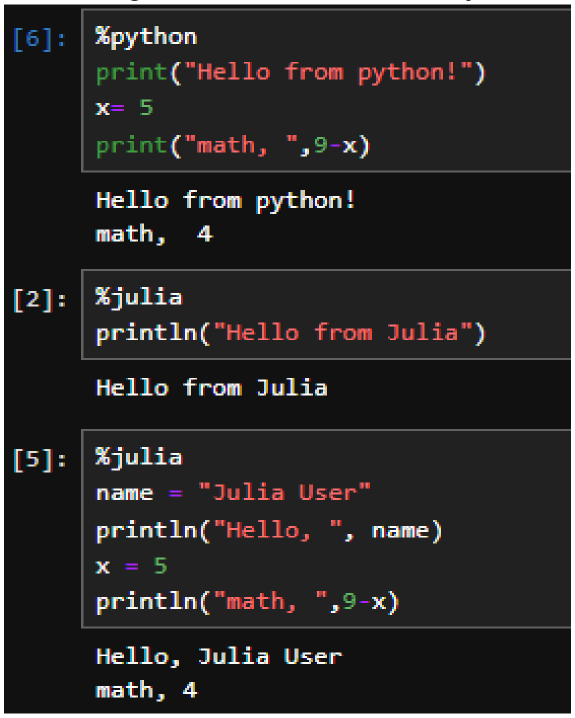
So python execution seems to work, julia didnt for some reason (i tried println("hello from julia")) so for now ill try to debug and fix that.
after debugging for a bit i figured the issue is me not capturing the julia output stream, so ill be working on figuring that out next session.

Im having alot of trouble getting the julia code to run correctly and return the stdout/stderr, after trying different approaches, the code worked for a single println but then gets an error for mulitple lines.



```
[4]: %julia
     println("Hello from Julia")

     Hello from Julia

[5]: %julia
     name = "Julia User"
     println("Hello, ", name)

     Traceback (most recent call last):
       File "C:\Users\Sam\Documents\College\SemesterProject\ProjectRepo\Semester-Project-Jupyter-Kernel\my_kernel.py", line 271, in handle_execute_request
         output, error_output = jl.seval(eval_str)
                                ^^^^^^^^^^^^^^^^^^
       File "C:\Users\Sam\.julia\packages\PythonCall\Nr75f\src\JlWrap\module.jl", line 27, in seval
         return self._jl_callmethod($(pyjl_methodnum(pyjlmodule_seval)), expr)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
     juliacall.JuliaError: ParseError("extra token after end of expression")
```

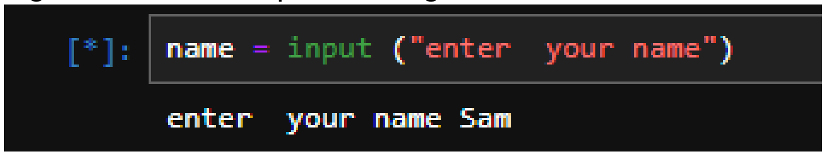
I fixed it after trying mulitple methods, the thing that finally worked for multiline code was wrapping the julia code with "begin" and "end" then use json.dumps on it, then finally running it with capture eval.



```
[6]: %python
     print("Hello from python!")
     x= 5
     print("math, ",9-x)

     Hello from python!
     math,  4

[2]: %julia
     println("Hello from Julia")

     Hello from Julia

[5]: %julia
     name = "Julia User"
     println("Hello, ", name)
     x = 5
     println("math, ",9-x)

     Hello, Julia User
     math, 4
```

So far so good, got basic different codes running in different cells, with different languages!.

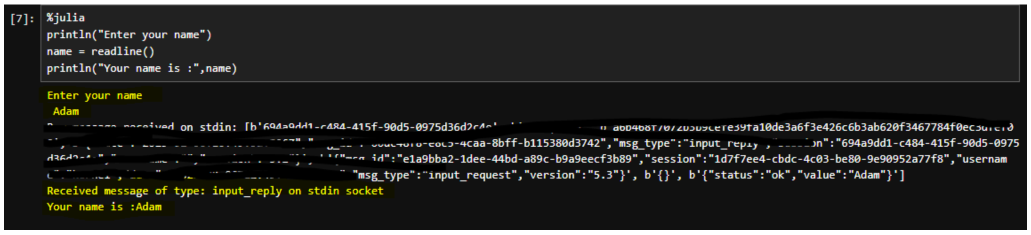
Now i need to work on taking in input from julia and python correctly, starting i made send_input_request, and handle_input reply functions. and also register stdin for polling
Now ill work on my execute code function to implement taking in input. ill start with python since it should be simpler to do .
i got some basic input working but for some reason the kernel gets stuck after reciveing the input



```
[*]: name = input ("enter  your name")

     enter  your name Sam
```

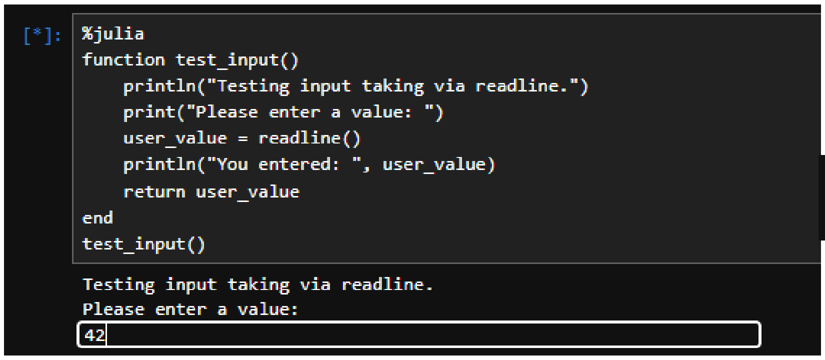
the problem was it seems that i needed to pool to get the reply so it doesnt get stuck, changed my code and now its working! will work on julia input (which should be harder to implement) next session.

- For julia input capturing i created a julia_input_callback function simillar to the python one, and added functions in julia language to handle the code capturing collabing with python , after trying multiple approaches and having problems/blocks (notebook got stuck when asking for julia input) and fixing it, julia can take input but now a new problem arised , it seems that taking input is blocking iocapture from working proprerly.
The problem is IOcapture takes the stdout all at once while i need it to stream it partially sometimes (for example println("Enter Name:") then name = readln(). this wont work because iocapture will return the output at the end in oneshot. so i need to figure a way to do partial streaming in chunks instead of all at once.
I tried a new streaming method, currently its not working.
i managed to get the new reimplementaion to work , hopefully this will be better than the IOcapture approach, so basically overrode python's stdout/stderr for a bit then restored them.
now for input taking ill make a simillar function to python's with a pooler to handle julia's input, as for the input capturing im thinking of using a simillar method to my stdout taking. after trying a bunch of different methods i found something that worked, it streams the prints, and handles inputs correctly

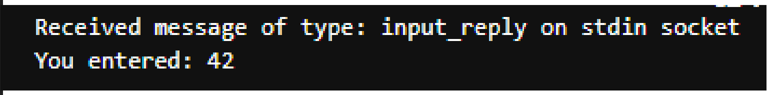


```
[7]: %julia
     println("Enter your name")
     name = readline()
     println("Your name is :",name)

     Enter your name
      Adam
     [redacted]ived on stdin: [b'694a9dd1-c484-415f-90d5-0975d36d2c4[redacted]a6f3e426c6b3ab620f3467784f0ec3[redacted]
     [redacted]-4caa-8bff-b115380d3742","msg_type":"input_reply",[redacted]on":"694a9dd1-c484-415f-90d5-0975
     d36d[redacted]id":"e1a9bba2-1dee-44bd-a89c-b9a9eecf3b89","session":"1d7f7ee4-cbdc-4c03-be80-9e90952a77f8","usernam
     [redacted]"msg_type":"input_request","version":"5.3"}', b'{}', b'{"status":"ok","value":"Adam"}']
     Received message of type: input_reply on stdin socket
     Your name is :Adam
```

So far so good, now ill try to do some small changes and then try a more complex input/printing to make sure its working fine, for example a function



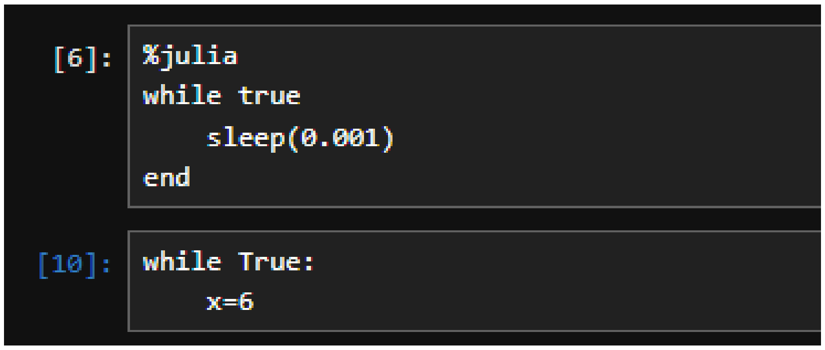
```
[*]: %julia
     function test_input()
         println("Testing input taking via readline.")
         print("Please enter a value: ")
         user_value = readline()
         println("You entered: ", user_value)
         return user_value
     end
     test_input()

     Testing input taking via readline.
     Please enter a value:
     42
```



```
Received message of type: input_reply on stdin socket
You entered: 42
```

Its working great!
Now that most of the code execution functionality im gonna go back and look at the code and try to improve some of the code before i continue.
- changed the execution env for python .
now i'll implement some other type of message handlings.
- for shutdown request ill make a handler function for the message itself + a shutdown function to actually implement the shutdown .
now ill work on code interupttion
i added a flag and a function to hande it, ill test it now with a while true loop or something. the problem is , which i was avoding so far for a while is that the code execution is blocking , so i wont recieve the interupt request till the code is actually finished execution, so i think now ill just do something i should have done from the start and actually run the code execution in seperate thread. i didnt want to deal with threads too much other than the simple heartbeat but for code exeuction it seems like i have to.
Now that i moved code exeution to a seperate thread, i dont need to pool anymore for input, since input will come from the main thread and its not blocked anymore.
so ill change do_input and julia input callback to not pool anymore and input relply to unblock.
i managed to get everything to work in the new seperate thread exeuction and removing the pooling and all previous prints/input taking seem to work, except it didnt fix my current issue of interupts. i need to figure out a way to check peridoically during execution for interupts.
the interrupt handling is very challenging especially on the julia side which im trying to deal with first since the python side should be easier, out of all the methods i tried so far , nothing seems to be working.
the problem seems to be that python doesnt have a native way to kill threads, so i have to find a way to kill the thread but also in a safe as possible way, the thread would be in the middle of execution .
After researching ways, it really seems there is no way to get over this safetly, i could use an unsafe method but this could casue complications later on. so i think my approach was wrong . i will try to change code execution from being in a seperate thread to being in a seperate process, this way i can just kill the process with no issues after getting an interrupt.

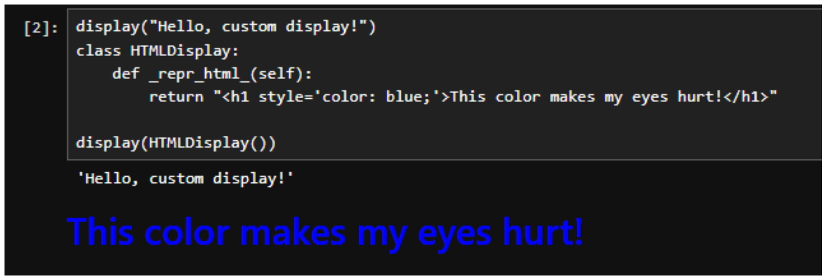
Ok so execute code implementation again...using processes now, imported from multiprocessing , Process and pipe, for communication, then i defined 2 functions that are very simillar , python worker and julia worker they do the stuff execute code function used to do before but now its in a diff process , one for python and one for julia, it takes and sends messages to the main process using the pipes.
in Kernel init we init the pipes and processes.
Added restart worker which bascially restarts the excecution process when we get an interrupt according to the given language.
Also made a thread to process input requests and results using the newly defined wait for result function (waiting for results func) , added interrupt handling where if theres an interrupt we just close the process and restart it. also changed the kernel json file to tell the kernel to use interrupt messages instead of signals.
So now i have a working interrupt + julia and python get thier own process with pipes to communicate with main thread and a helper thread for input etc...



```
[6]:  %julia
      while true
          sleep(0.001)
      end

[10]: while True:
          x=6
```

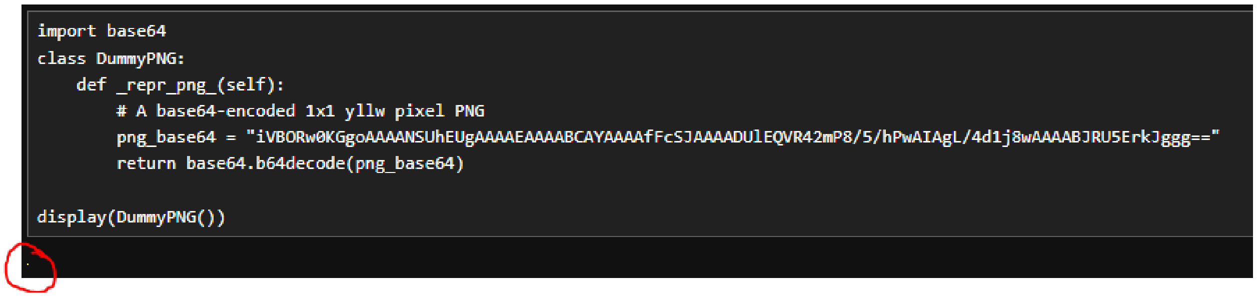
Both of these got interrupted succesfully and the other cells worked after the restart.
-P.S since the interrupt restarts the process, running new code again will take a bit longer since it'll init some stuff , but only the first code run in that language, the rest will run fast.

- Next up... i think ill add the ability to display images in the notebook like graphs etc for both python and julia. ill research the message scheme for these and try to implement it.
- Added send display data function for that purpose.
- inside python worker, i made a function custom_display using builtins to override pythons display , capture the displays according to thier type, (basically the same approach as i did with inputs earlier.+ added a handling elif in my wait_for result thread to handle the display data and send response.
i did the python stuff first because its less complicated :



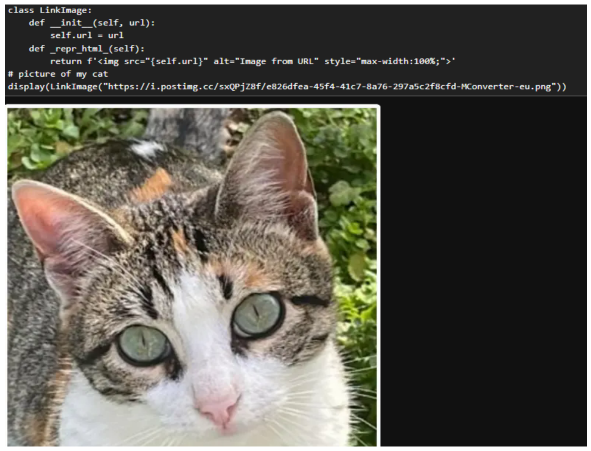
```
[2]: display("Hello, custom display!")
     class HTMLDisplay:
         def _repr_html_(self):
             return "<h1 style='color: blue;'>This color makes my eyes hurt!</h1>"

     display(HTMLDisplay())

     'Hello, custom display!'
```

This color makes my eyes hurt!



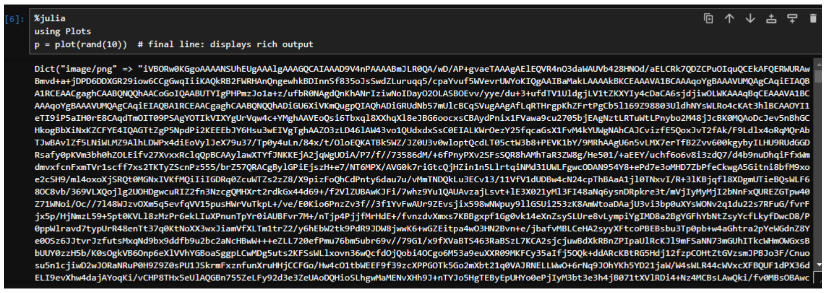
```
import base64
class DummyPNG:
    def _repr_png_(self):
        # A base64-encoded 1x1 yllw pixel PNG
        png_base64 = "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAADUlEQVR42mP8/5/hPwAIAgL/4d1j8wAAAABJRU5ErkJggg=="
        return base64.b64decode(png_base64)

display(DummyPNG())
```

html image,iused a picture of mycat (very cute yes)



```
class LinkImage:
    def __init__(self, url):
        self.url = url
    def _repr_html_(self):
        return f'<img src="{self.url}" alt="Image from URL" style="max-width:100%;">'
# picture of my cat
display(LinkImage("https://i.postimg.cc/sxQPjZ8f/e826dfea-45f4-41c7-8a76-297a5c2f8cfd-MConverter-eu.png"))
```

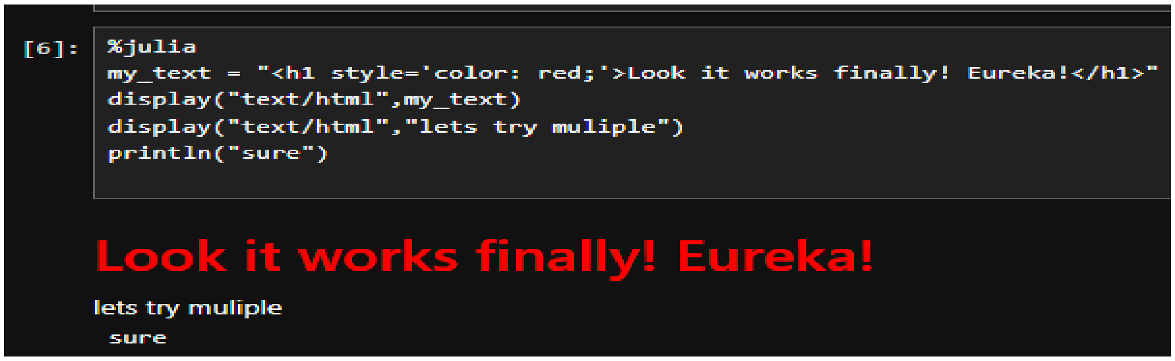
Now since its working ill try adding more types of formats to display, like jpeg and svg.
- this method will rely on a _repr function, from what i saw, most popular libs have one so it shouldnt be an issue, i changed my code again to ovewrite displayhook instead of display.
-matplob compatibility looks like a challenge so ill do the julia part for now and maybe get back to it later.
as for julia i tried using the same approach by making a funcing in my MyStreaming to do the same as the python function, currently im getting issues and instead of a plot, its giving back a string stream representation of it, so now im trying to figure out why and fixing it. (also edited printcallback),



```
[6]: %julia
     using Plots
     p = plot(rand(10))  # final line: displays rich output
```

```
Dict("image/png" => "iVBORw0KGgoAAAANSUhEUgAAAlgAAAGQCAIAAAD9V4nPAAAABmJLR0QA/wD/AP+gvaeTAAAgAElEQVR4nO3daWAUVb428HNOd/aELCRk7QDZCPuOIquQCEkAFQERWURAw
Bmvd+a+jDPD6DDXGR29iow6CCgGwqIiiKAQkRB2FWRHAnQngewhkBDInnSf835oJsSwdZLuruqq5/cpaYvuf5WVevrUWYoKIQgAAIBaMakLAAAAkBKCEAAAVA1BCAAAqoYgBAAAVUMQAgCAqiEIAQB
A1RCEAACgaghCAABQNQQhAACoGoIQAABUTYIgPHPmzJo1a+z/ufbR0NAgdQnKhANrIziwNoIDayO2OLASBOEvv/yye/du+3+ufdTV1UldgjLV1tZKXYIy4cDaCA6sjdjiwOLWKAAAAqBqCEAAAAVA1BC
AAAqoYgBAAAVUMQAgCAqiEIAQBA1RCEAACgaghCAABQNQQhADiGU6XiVKmQugpQIAQhADiGRUdNb57mUlcBCqSVugAAgAfLqRTHrgpKhZFrtPgCb5l169Z98803UldhNYsWLRo4cKAt3hlBCOAYI1
eTI9iP5aIH0rE8CAqdTmOIT09PSAgYOTIkVIXYgUrVqw4c+YMghAAVEoQsi6Tbxql8XXhqXl8eJBG6oocxsCBAydPnix1FVawa9cu2705bjEAgNztLRTuWtLPnybo2M48jJcBK0MQAoDcJev5nBhGC
HkogBbXiNxKZCFYE4IQAGTtZgP5NpdPi2KEEEbJY6Hsu3wEIVgTghAAZO3zLD46lAW43vo1QUdxdxSsC0EIALKWrOezY25fqcaGsX1FvM4kYUWgNAhCAJCvizfE5QoxJvT2fAk/F9Ldlx4oRqMQrAb
TJwBAvlZf5LNiWLMZ9AlhLDWPx4diEoVylJeX79u37/Tp0y4uLn/84x/t/OloEQKATBk5WZ/JZ0U3v0wloptQcdLT05ctW3b8+PEVK1bY/9MRhAAgU6n5vLMX7erTfB2Zvv600kgybyILHU9RUdGGD
Rsafy0pKVm3bh0hZOLEifv27XvxxRclqQpBCAAylawXTYfJNKKEjA2jqWgUOiA/P7/f//73586dM/+6fPnyPXv2SFsSQR8hAMhTaR3ZW8g/He501/+aEEY/uchf6o6v8i3zdQ7/d4b9nuDhqiFfxWm
dmvxfcnFxmTVr1scff7xs2TKTyZScnPz555/brZ57QRACgBylGPiEjszH+e7/NT6MPX/AVG0k7riGtcQjHZin1n5LlrtqiNMd31UWLFgwcODAN954Y8+ePd7e3oMHD7ZbPfeCkwgA5Gitni8bfM9xo
e2cSH9/ml4oxoXjSRQt0MGNxIVKfMQiIiIGDRq0ZcuWTZs2zZ8/X9pizFoQhCdPnty6dau7u/vMmTNDQkLu3ECv13/11VfV1dVV1dUDBw4cN24cpThBAaA1jl0TNxvI/R+3lKBjqfl8XDgmUTieBQsWLF6
8OC8vb/369+0LXQojlg2UOHDgwcuRIZ2fn3NzcgQMHXrt2rdkGx44/d69+/f2VlZUBAwKJFi/7whz9Yu1QAUAvzajLsvt+lE3X021yMl3FI48aNq6sDRpkre3t/mVjIyMyMjI2bNnNfxQUREZGTpw40
Z71WNoi/Oc//7148WJzvOXm5q15evfqVV15pusHw+Wpuy9llGSUi253zK8AmWtoaDAajU3DAajU3vi3bp0uXYsWONv2q1du22s7RFuG/fvrF
jx5p/HjNmzPzL59+5pt0KVL18zMzTsySLUre8vLympiYgIMD8a2BgYGFhh2BgYGFhYbNtZsyYcfLkyfDwcD8/P
0ppWlravd7typUrR48enTt37q0KtNoXKX3wxJiamVfXLTm1trZ2/y6hEbW2tk9PdR9JDW8jwwK6+wGZEitpa4wO3HN2Bvn+e/jbafvMBLCeHA2syyXFtcoPBEBsbu3Tp0pb+w4aGhtra2DnZ8Y
e0OSz6JJtvrJzfutsMxqNd9bx9ddfb9u2bc2aNcHBwW+++eZLL720efPmu76bm5ubr69v//79G1/x9fXVaBTS463RaBSzL7KCA2sjcjuwBdXkRBnZPIpaUlRcKJl9mFSaNN73mGUhITkcWHmOWGxsB
bUUY0zzH5b/K0sOgkVB6Onp6eXlVVhYGBoaSggpLCwcLwMDg5uts2KFSsWLlxovn36xovn36wQcfdOjQobi4OCgo6M53a9euXXR09Mkf3F/Cnuo
su5n1cjiwD2wJORaNRuP0H9Z9Z0sPU1JSkrmFxznfunXruHHjCCFGo/Hw4cO1tbWEEF9f39zcXPPGOTk5Go2mXbt21q0VAJRNELLWwO+6rNq9JOhYKh5YD21jaW/W4sWLR44cWVxcXFBQUF1dPX36d
ELI9evXhw4dajAYoqKi/vCHP8THx5eUlAQGBn755ZeLFy92d3e3ZeUAoDQHioSLhgwMaMENvXHh9J+nTYJo5HgTEByEpUHYo0ePj/jB071tXVlRDi4+Nz4MCBsLAwQki/fv0MBsOBAwc
```

i reformatted both the python and julia display functions, and aswell the send_display function so hopefully that'll help and not cause more bugs. not working, ill try changing Dict String to Dict Any in julia code.
it works you'll have to just use display() and its not inline the notebook.. i think its like matplot problem, so for now ill just check basic images png/jpeg/svg/html.
-This is frustrating, i tried alot of different functions/ code etc in julia but i cant seem to capture the output correctly so its sent to my send_display function and rendered on the notebook...
so i added julia code for a bunch of display overrides , one for 1 parameter one for 2 etc, and a global variable that'll store current display.
i tried to make sure the dict is converted while passed to python and it still didnt help , ill keep it because its better to check anyway.
Nevermind!! that was actually the issue ! i just needed to relax the constaints of the "if". For now HTML worked, ill try/work on the rest next session.

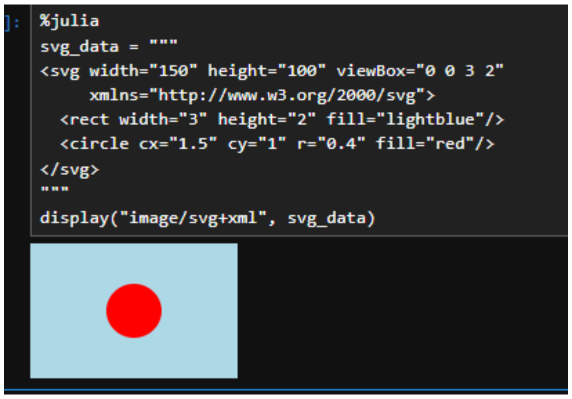


```
[6]: %julia
     my_text = "<h1 style='color: red;'>Look it works finally! Eureka!</h1>"
     display("text/html",my_text)
     display("text/html","lets try muliple")
     println("sure")
```

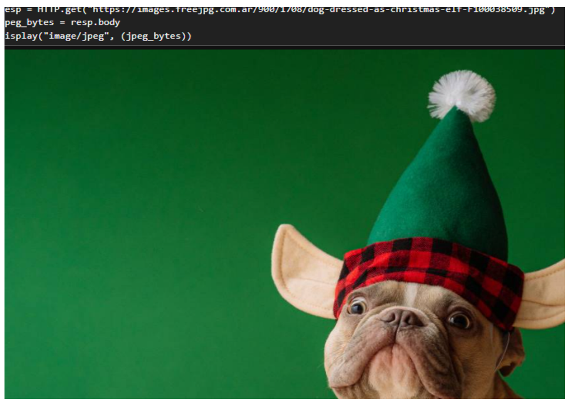
Look it works finally! Eureka!

lets try muliple
 sure

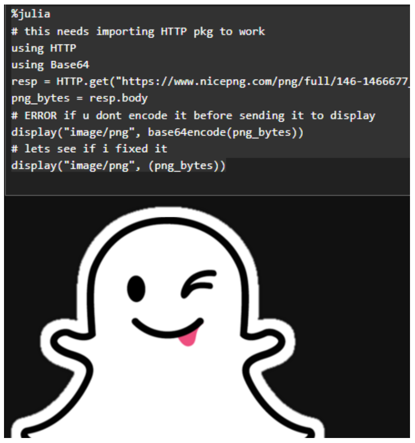
Okay now ill test other display types!
SVG worked, png didnt, will be figuring out why and fixing it. ok the problem is sending the raw png bytes will casue an error but b64 encoding them first works. fixed it by adding a check in my display function that automatically converts to b64 if the image already isnt!



```
%julia
svg_data = """
<svg width="150" height="100" viewBox="0 0 3 2"
     xmlns="http://www.w3.org/2000/svg">
  <rect width="3" height="2" fill="lightblue"/>
  <circle cx="1.5" cy="1" r="0.4" fill="red"/>
</svg>
"""
display("image/svg+xml", svg_data)
```

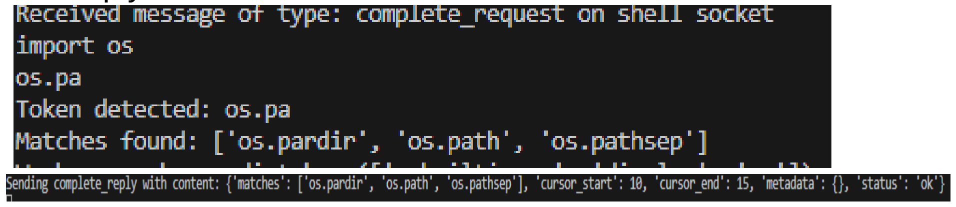


```
esp = HTTP.get("https://images.freejpg.com.ar/900/1708/dog-dressed-as-christmas-elf-F100038509.jpg")
peg_bytes = resp.body
isplay("image/jpeg", (jpeg_bytes))
```



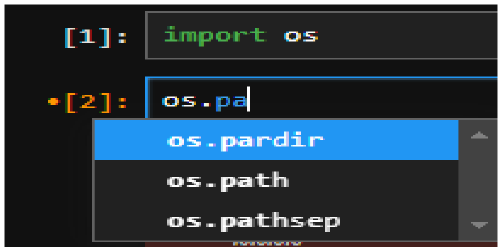
```
%julia
# this needs importing HTTP pkg to work
using HTTP
using Base64
resp = HTTP.get("https://www.nicepng.com/png/full/146-1466677_
png_bytes = resp.body
# ERROR if u dont encode it before sending it to display
display("image/png", base64encode(png_bytes))
# lets see if i fixed it
display("image/png", (png_bytes))
```

Next up ill implement python code completion,first ill import re and rlcompleter.
then ill make a handle_complete_request function itll use rl completer , get cursor position, get all possible matches for token by iteration then send a reply with cursor start,end and matches.
now that thats done i just need to add the handler to my handle_message function to tell it if we get a autocomplete request to route it to my function.
small problem, since this notebook has 2 languages ill have to check what language..
after doing all of this it didnt quite work, i think ill move the complete handling to the python worker process and try again.
Okay so i switched it and edited some stuff, made some progress, the tokens are detected correctly in my debugs but im not getting anything to help autocomplete in the notebook itself. so gotta find out why/.
the reply content looks correct aswell.



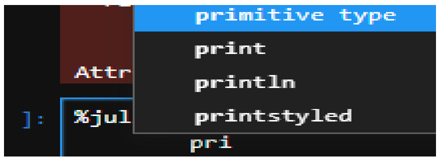
```
Received message of type: complete_request on shell socket
import os
os.pa
Token detected: os.pa
Matches found: ['os.pardir', 'os.path', 'os.pathsep']
Sending complete_reply with content: {'matches': ['os.pardir', 'os.path', 'os.pathsep'], 'cursor_start': 10, 'cursor_end': 15, 'metadata': {}, 'status': 'ok'}
```

i cant seem to pinpoint the issue, i tried some code changes that didnt help, the reply message looks correct so in theory i should be getting a autocomplete popup. but for some reason im not.
apparently the problem was not wrapping it in a busy idle ... took way too long to debug for something this simple, i thought busy idle was for time consuming stuff like execution and the documentation didnt mention using it for complete requests..oh well it works now.



```
[1]: import os

•[2]: os.pa
      os.pardir
      os.path
      os.pathsep
```

Lets see, now to do this in julia, after looking at some methods i can do the naive way by iterating over possible functions in Base etc, or just use REPLCompletions , i'll use the ReplCompletions since it seems more effiecient, ill make a julia function to handle completions, send it in the correct format to the python side, etc like the python way.
it worked but its giving too much text not just the autocomplete, lemme try to remove the extra text, its almost working! just small issue with im guessing cursor pos/start/end where stuff isnt getting replaced totally. got it to work!



```
Attr   primitive type
       print
       println
]: %jul printstyled
       pri
```