Why are you interested in this project and how do you see its implementation?

This project is interesting to me because I would like to participate in the research of such interesting phenomena. I use Jupiter notebooks pretty often and notice the bad code quality in them, and the difficulty to operate with connected, chained cells, so it will be a pleasure to try to fix it and make Notebook better myself, I am very excited about a real research project with soft I really use on daily basis. I am ready to learn new things quickly and get new experiences from mentors and people from the industry.

I satisfy the requirements because I had the Advanced Python course and two years of discrete mathematics graph theory. Also, I am finishing the Kotlin course right now and planning to go to Advanced Kotlin next semester, also I had a second year semester Kotlin project.

Some thoughts about implementing the project.

Firstly I think it will be a good idea to analyze what type of connection we are interested in. For example, creating and using the variables; defining and calling functions; importing modules and using fields from them, and so on.

Then we should learn how to differ if the detected case is important or should be skipped (same variables names in different scopes, usage of a variable without mutating its value, and more)

After that extract chains and make the execution group from them. Then build a plugin.

Ввод [1]:

```
import astpretty
import ast
import graphviz
```

Ввод [2]:

```
g = graphviz.Graph()
cnt = 0
var_colors = {}


def get_color():
    colors = ["Red", "Orange", "Green", "Blue", "Yellow", "Purple"]
    i = 0
    while True:
        yield colors[i]
        i += 1
gen = get_color()
```

Ввод [3]:

```
def getlabel(node):
    name = node.__class__.__name__
    label = name
    ret = [name]
    if isinstance(node, ast.Constant):
        label += ": " + str(node.value)
        ret.append(str(node.value))
    elif isinstance(node, ast.Name):
        label += ": " + str(node.id)

        ret.append(str(node.id))
    elif isinstance(node, ast.FunctionDef):
        label += ": " + str(node.name)
        ret.append(str(node.name))
    elif isinstance(node, ast.arg):
        label += ": " + str(node.arg)
        ret.append(str(node.arg))
    else:
        ret.append("unknown")
    ret.append(label)

    return ret
```

Ввод [4]:

```
class Visitor(object):

    def generic_visit(self, node):
        global gen
        global cnt, var_colors
        curnum = cnt
        col = "black"
        style = "unfilled"
        name, val, lab = getlabel(node)
        if lab != "unknown":
            if name == "Name":
                if val not in var_colors.keys():
                    var_colors[val] = next(gen)
                col = var_colors[val]
                style = "filled"

        g.node(str(curnum), label=lab, shape="square", color=col, style=style)

        cnt += 1
        for field, value in ast.iter_fields(node):
            if isinstance(value, list):
                for item in value:
                    if isinstance(item, ast.AST):
                        g.edge(str(curnum), str(self.generic_visit(item)))

            elif isinstance(value, ast.AST):
                g.edge(str(curnum), str(self.generic_visit(value)))
        return curnum
```

Ввод [5]:

```
def ast_pprint_pic(code):
    x = Visitor()
    x.generic_visit(ast.parse(code))
    g.render('artifacts/ast')
```

Ввод [7]:

```
source_code = open('test.py').read()
ast_pprint_pic(source_code)
```

```
nodes -- operations, variables, functions
edges -- ast edges
same colors -- same variables (may be connected between each other and probobly
should be excecuted together)
```

Ввод [23]:

```
from IPython.display import IFrame
IFrame("./artifacts/ast.pdf", width=900, height=900)
```

Out[23]: