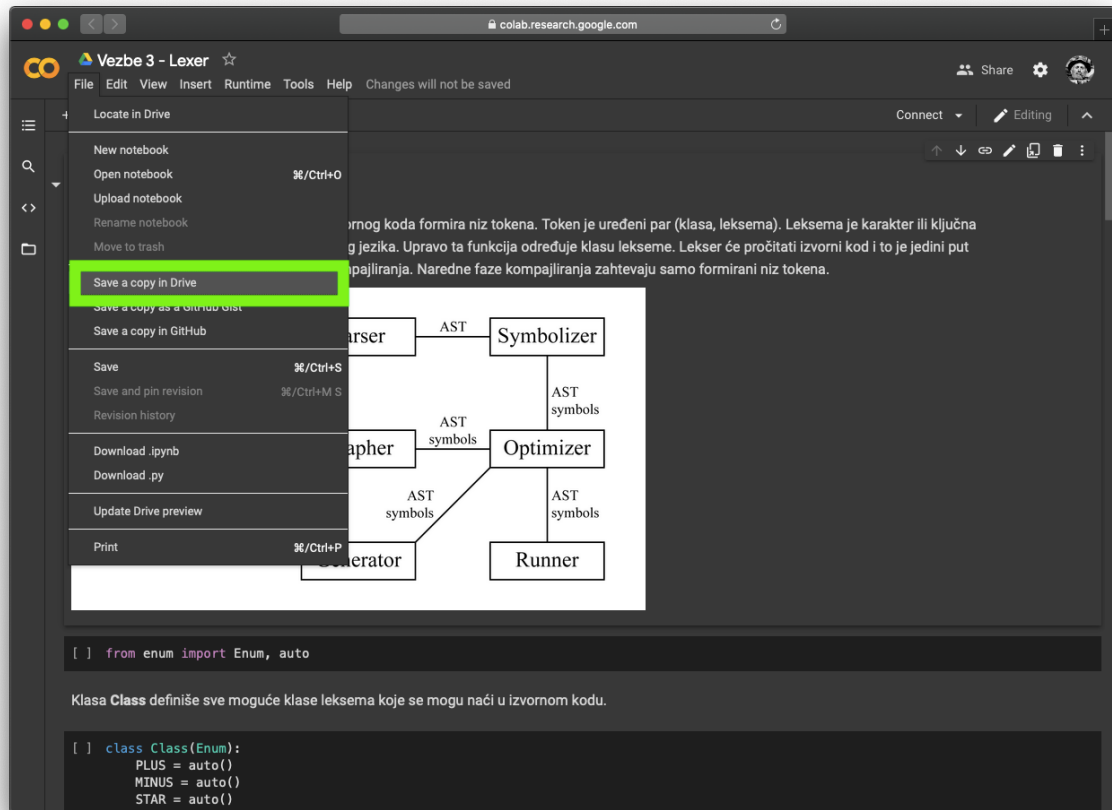


## 1. Kreiranje kopije Notebooka



## 2. Povezivanje na virtuelnu mašinu

The screenshot shows a Google Colab notebook interface. The title bar indicates the notebook is named "Copy of Vezbe 3 - Lexer". The main content area displays a diagram of the compilation process and some Python code.

**Diagram of the compilation process:**

```
graph LR; SC[source code] -- characters --> L[Lexer]; L -- tokens --> P[Parser]; P -- AST --> S[Symbolizer]; S -- AST symbols --> O[Optimizer]; O -- AST symbols --> G[Grapher]; O -- AST symbols --> R[Runner]; G -- AST symbols --> R
```

The diagram illustrates the flow of data during compilation. It starts with "source code" being processed by a "Lexer" (receiving "characters" and outputting "tokens"). The "tokens" are then processed by a "Parser" (outputting "AST"). The "AST" is then processed by a "Symbolizer" (outputting "AST symbols"). The "AST symbols" are then processed by an "Optimizer" (outputting "AST symbols"). The "Optimizer" then processes the "AST symbols" and outputs them to a "Grapher" and a "Runner". The "Grapher" also outputs "AST symbols" to the "Runner".

**Python code snippet:**

```
[ ] from enum import Enum, auto
```

Klasa `Class` definiše sve moguće klase leksema koje se mogu naći u izvornom kodu.

```
[ ] class Class(Enum):  
    PLUS = auto()  
    MINUS = auto()  
    STAR = auto()  
    ...
```

### 3. Pregled trenutnog direktorijuma

Copy of Vezbe 3 - Lexer

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk

Editing

Vežbe 3: Lexer

Lexer je deo kompajlera koji na osnovu izvornog koda formira niz tokena. Token je uređeni par (klasa, leksema). Leksema je karakter ili ključna reč koja ima funkciju u sintaksi programskog jezika. Upravo ta funkcija određuje klasu lekseme. Lekser će pročitati izvorni kod i to je jedini put kada će se to uraditi u čitavom procesu kompajliranja. Naredne faze kompajliranja zahtevaju samo formirani niz tokena.

```
graph LR; SC[source code] -- characters --> L[Lexer]; L -- tokens --> P[Parser]; P -- AST --> S[Symbolizer]; S -- AST symbols --> O[Optimizer]; G[Grapher] -- AST symbols --> O; O -- AST symbols --> R[Runner]; O -- AST symbols --> Gen[Generator];
```

The diagram illustrates the compilation process flow. It starts with 'source code' which is processed by the 'Lexer' (receiving 'characters' and outputting 'tokens'). The 'tokens' are then processed by the 'Parser' (outputting 'AST'). The 'AST' is then processed by the 'Symbolizer' (outputting 'AST symbols'). The 'AST symbols' are then processed by the 'Optimizer' (outputting 'AST symbols'). The 'Optimizer' also receives 'AST symbols' from the 'Grapher' and the 'Generator'. Finally, the 'Optimizer' outputs 'AST symbols' to the 'Runner'.

```
[ ] from enum import Enum, auto
```

Klasa `Class` definiše sve moguće klase leksema koje se mogu naći u izvornom kodu.

```
[ ] class Class(Enum):  
    PLUS = auto()  
    MINUS = auto()  
    STAR = auto()  
    DIVIDE = auto()  
    ...
```

## 4. Mountovanje Google diska na trenutni direktorijum

**Copy of Vezbe 3 - Lexer**

File Edit View Insert Runtime Tools Help All changes saved

RAM Disk

Editing

### Vezbe 3: Lexer

[Lexer](#) je deo kompajlera koji na osnovu izvornog koda formira niz tokena. Token je uređeni par (klasa, leksema). Leksema je karakter ili ključna reč koja ima funkciju u sintaksi programskog jezika. Upravo ta funkcija određuje klasu lekseme. Lexer će pročitati izvorni kod i to je jedini put kada će se to uraditi u čitavom procesu kompajliranja. Naredne faze kompajliranja zahtevaju samo formirani niz tokena.

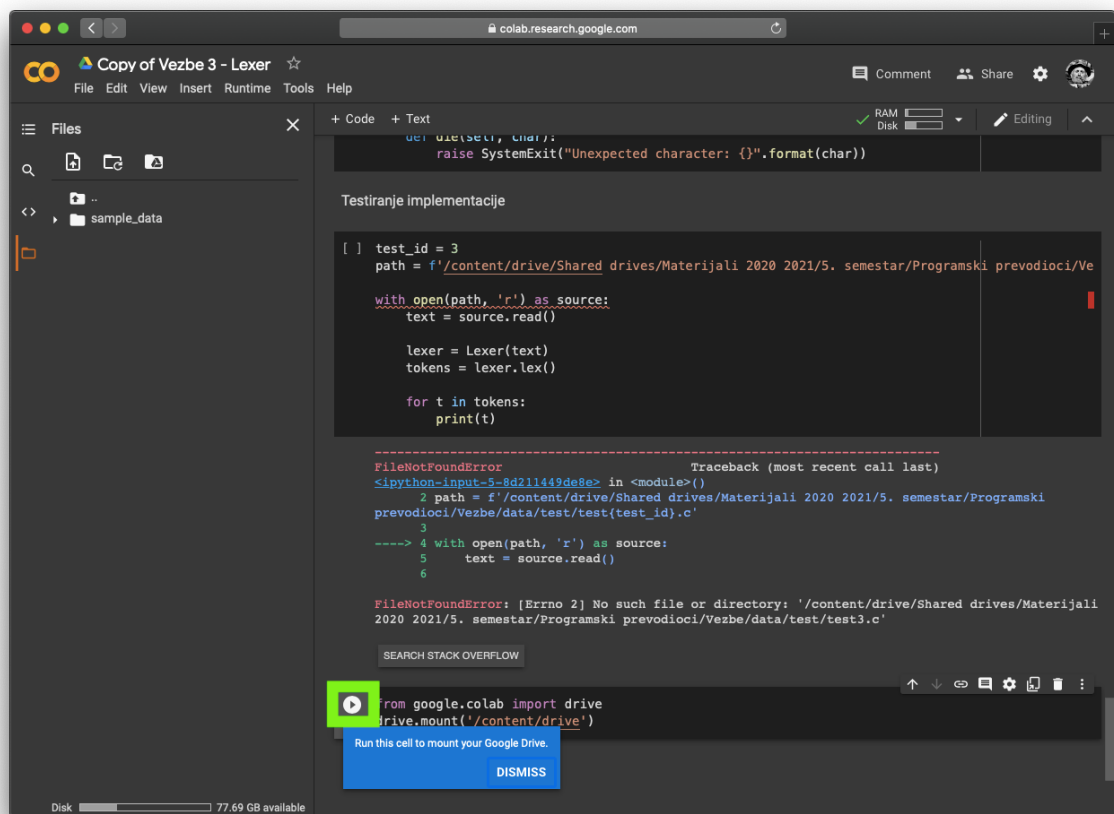
```
graph LR
    SC[source code] -- characters --> L[Lexer]
    L -- tokens --> P[Parser]
    P -- AST --> S[Symbolizer]
    S -- AST symbols --> O[Optimizer]
    G[Grapher] -- AST symbols --> O
    Gen[Generator] -- AST symbols --> O
    O -- AST symbols --> R[Runner]
```

```
[ ] from enum import Enum, auto
```

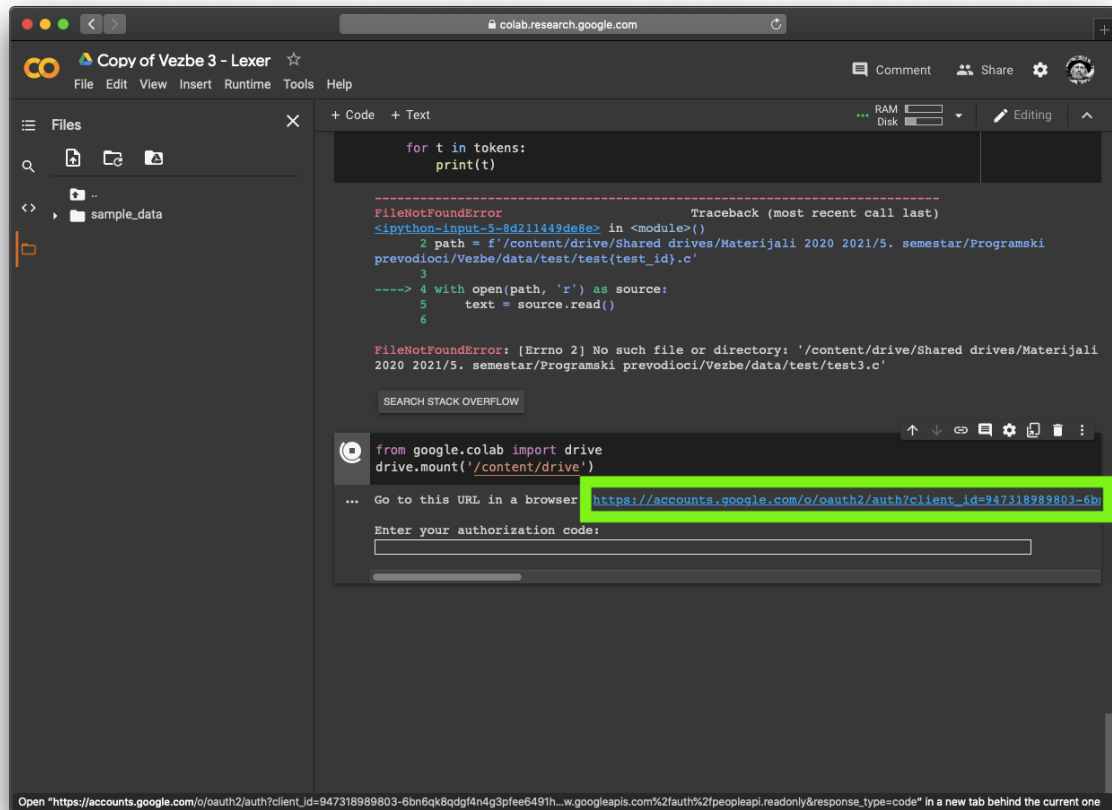
Klasa **Class** definiše sve moguće klase leksema koje se mogu naći u izvornom kodu.

```
[ ] class Class(Enum):
    PLUS = auto()
    MINUS = auto()
```

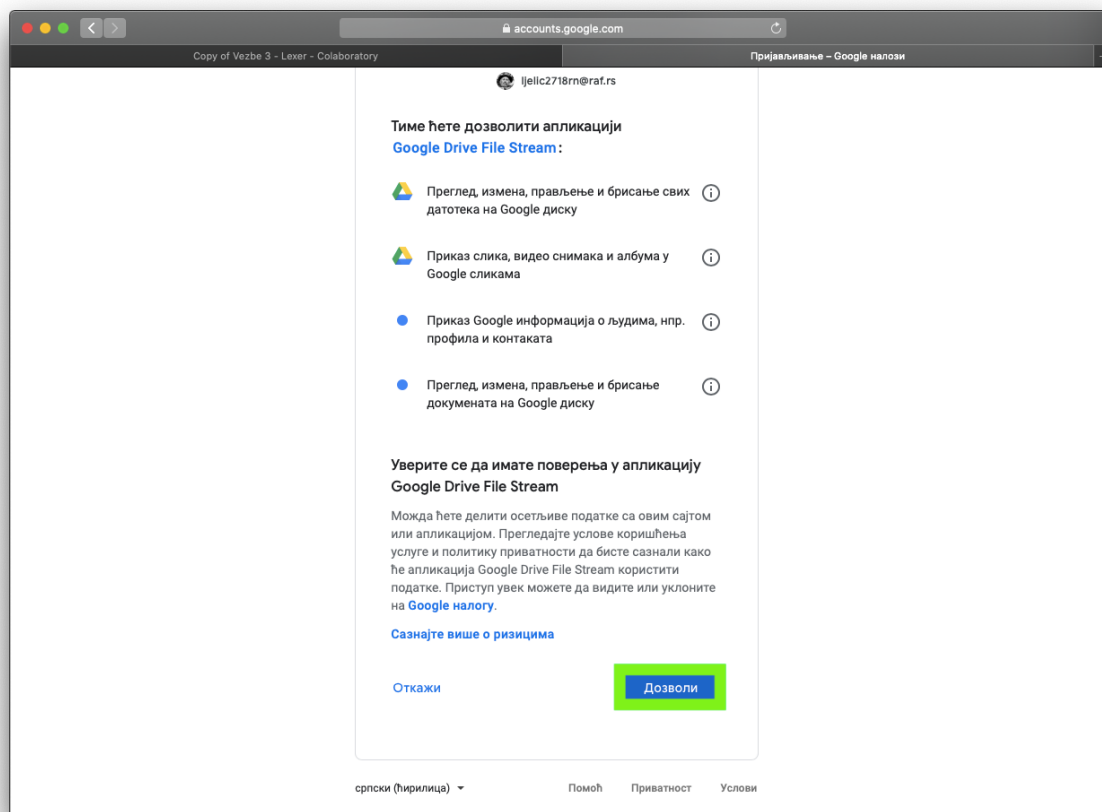
## 5. Pokretanje ćelije za autorizaciju



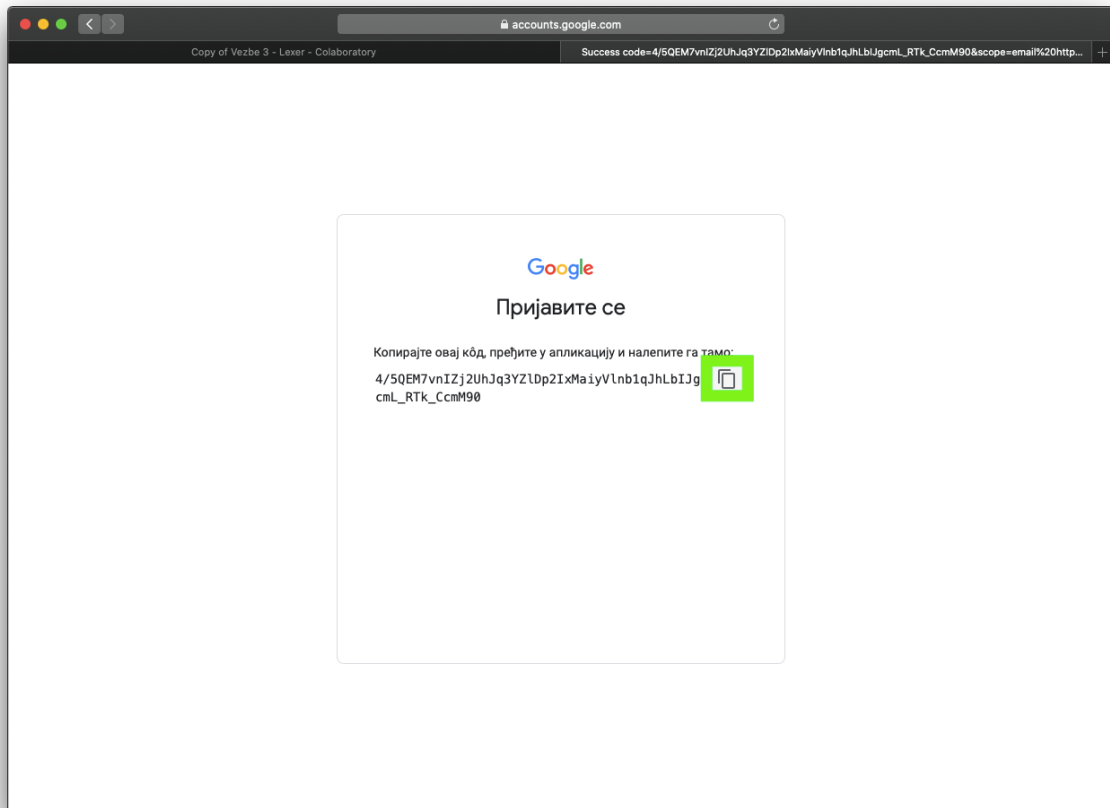
## 6. Klik na link za autorizaciju



## 7. Dodela dozvole za pristup Google disku

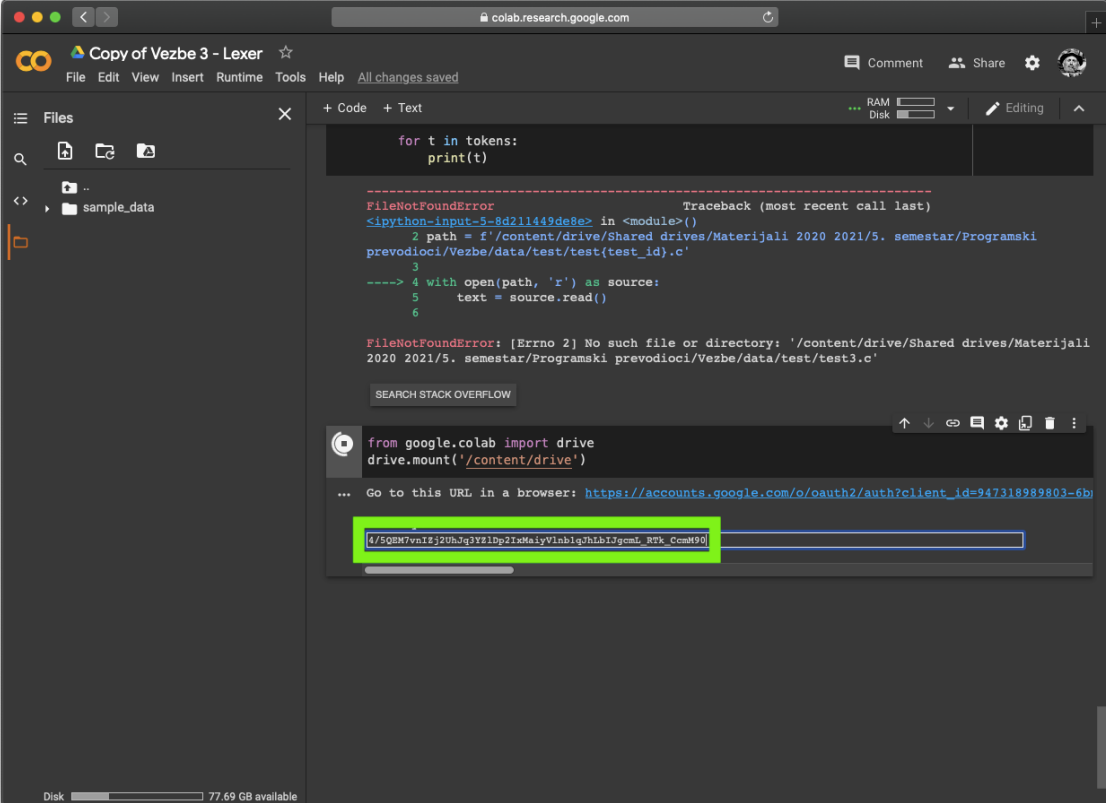


## 8. Kopiranje koda za autorizaciju





## 9. Unos koda za autorizaciju



The screenshot shows a Google Colab notebook titled "Copy of Vezbe 3 - Lexer". The left sidebar shows a file explorer with a folder named "sample\_data". The main code area contains a Python script that attempts to read a file from a Google Drive path. The script is as follows:

```
for t in tokens:
    print(t)

-----
FileNotFoundError                                Traceback (most recent call last)
<ipython-input-5-8d211449de8e> in <module>()
      2 path = f'/content/drive/Shared drives/Materijali 2020 2021/5. semestar/Programski
prevodioci/Vezbe/data/test/test(test_id).c'
      3
----> 4 with open(path, 'r') as source:
      5     text = source.read()
      6

FileNotFoundError: [Errno 2] No such file or directory: '/content/drive/Shared drives/Materijali
2020 2021/5. semestar/Programski prevodioci/Vezbe/data/test/test3.o'

SEARCH STACK OVERFLOW
```

Below the error, there is a code cell with the following code:

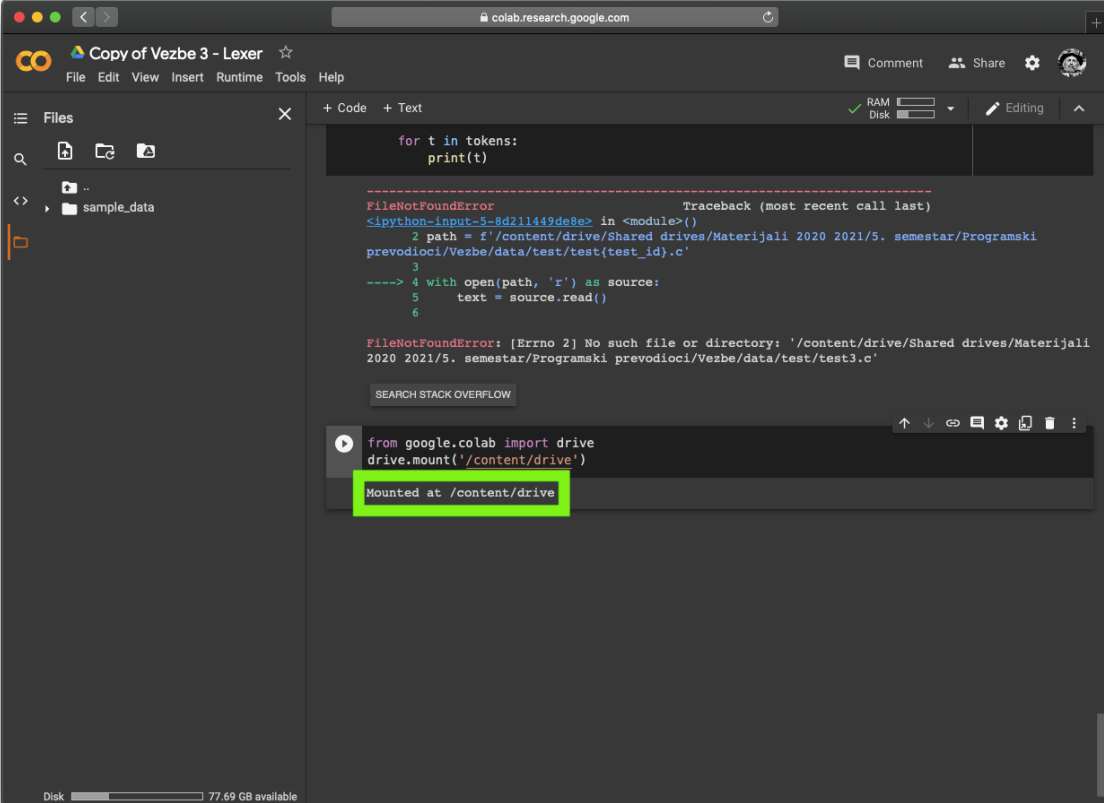
```
from google.colab import drive
drive.mount('/content/drive')
```

Below the code cell, there is a text box with the following text:

... Go to this URL in a browser: [https://accounts.google.com/o/oauth2/auth?client\\_id=947318989803-6b4/SQEM7vniIsj2Uhg3Yz1Dp2LxMaIyVlnblqHbLJgcmL\\_RTk\\_CcmM90](https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6b4/SQEM7vniIsj2Uhg3Yz1Dp2LxMaIyVlnblqHbLJgcmL_RTk_CcmM90)

The URL is highlighted in green. Below the URL, there is a text input field.

## 10. Čekanje na završetak mountovanja



The screenshot shows the Google Colab interface. The top bar indicates the notebook is titled "Copy of Vezbe 3 - Lexer". The left sidebar shows a file explorer with a folder named "sample\_data". The main code area contains a Python script that attempts to open a file from a Google Drive path. The script is as follows:

```
for t in tokens:
    print(t)

-----
FileNotFoundError                                Traceback (most recent call last)
<ipython-input-5-8d211449de8e> in <module>()
      2 path = f'/content/drive/Shared drives/Materijali 2020 2021/5. semestar/Programski
prevodioci/Vezbe/data/test/test(test_id).c'
      3
----> 4 with open(path, 'r') as source:
      5     text = source.read()
      6

FileNotFoundError: [Errno 2] No such file or directory: '/content/drive/Shared drives/Materijali
2020 2021/5. semestar/Programski prevodioci/Vezbe/data/test/test3.o'

SEARCH STACK OVERFLOW
```

Below the error, a new code cell is shown with the following code:

```
from google.colab import drive
drive.mount('/content/drive')
```

The output of this code cell is "Mounted at /content/drive", which is highlighted with a green box. The bottom status bar shows "Disk" and "77.69 GB available".

## 11. Pokretanje svih ćelija

The screenshot shows a Google Colab notebook interface. The 'Runtime' menu is open, and the 'Run all' option is highlighted with a green rectangle. The notebook content includes a diagram of a compiler's phases and some Python code.

**Diagram of Compiler Phases:**

```
graph LR
    SC[source code] --> L[Lexer]
    L --> S[Symbolizer]
    S --> O[Optimizer]
    O --> R[Runner]
    O --> G[Grapher]
    G --> Gen[Generator]
```

The diagram shows the flow of a compiler. It starts with 'source code' which is processed by the 'Lexer'. The output of the 'Lexer' is then processed by the 'Symbolizer'. The output of the 'Symbolizer' is then processed by the 'Optimizer'. The 'Optimizer' produces 'AST symbols' which are then processed by the 'Runner'. The 'Optimizer' also produces 'symbols' which are then processed by the 'Grapher'. The 'Grapher' produces 'AST symbols' which are then processed by the 'Generator'.

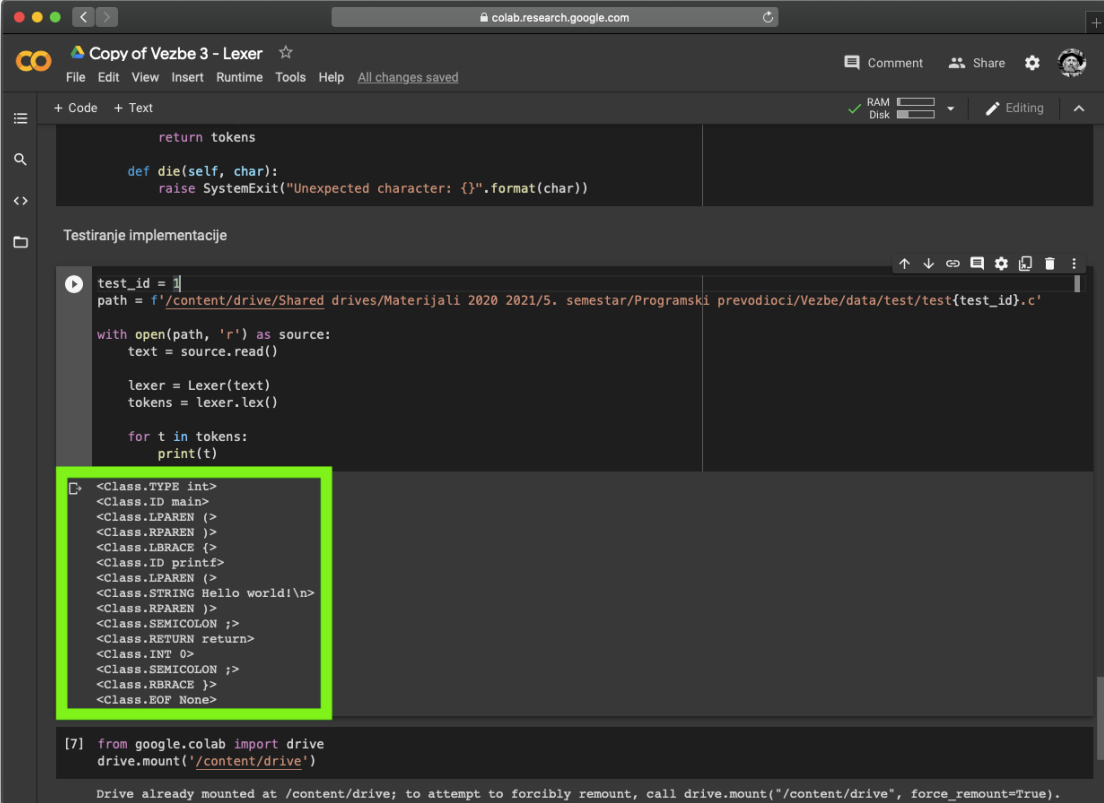
**Python Code:**

```
[ ] from enum import Enum, auto

Klasa Class definiše sve moguće klase leksema koje se mogu naći u izvornom kodu.

[ ] class Class(Enum):
    PLUS = auto()
    MINUS = auto()
    STAR = auto()
    DIVIDE = auto()
```

## 12. Uspešna leksička analiza C izvornog fajla



```
return tokens

def die(self, char):
    raise SystemExit("Unexpected character: {}".format(char))
```

Testiranje implementacije

```
test_id = 1
path = f'/content/drive/Shared drives/Materijali 2020 2021/5. semestar/Programski prevodioci/Vezbe/data/test/test{test_id}.c'

with open(path, 'r') as source:
    text = source.read()

    lexer = Lexer(text)
    tokens = lexer.lex()

    for t in tokens:
        print(t)
```

```
<Class.TYPE int>
<Class.ID main>
<Class.LPAREN (>
<Class.RPAREN )>
<Class.LBRACE {>
<Class.ID printf>
<Class.LPAREN (>
<Class.STRING Hello world!\n>
<Class.RPAREN )>
<Class.SEMICOLON ;>
<Class.RETURN return>
<Class.INT 0>
<Class.SEMICOLON ;>
<Class.RBRACE }>
<Class.EOF None>
```

```
[7]: from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True).