assignment-6

By: Alexander
UMSL

Coding Journey Summary:

- ⏱️ 1 hour 26 minutes 14 seconds of active typing
- 📅 1 day 9 hours 10 minutes 52 seconds total duration
- 🔢 4,124 characters | 94 lines
- 📈 Typing speed: 1 character(s) per second
- 🗄️ Process Data: Showing 277 changes made every 5+ seconds (of 557 changes total)
- 📊 Interactive version of this report is AVAILABLE HERE.

Final Code

If the code contains more than one file, they are concatenated and shown.

```
==+==
FILE: Main.py
==+==

from number_1 import number_1
from number_3 import number_3
from number_4 import number_4
from number_5 import number_5
from number_6 import number_6
from number_7 import number_7

BDAY = 22
D = 2 + (BDAY % 2)
M = 1 + (BDAY // 2)

# 1 and 2 are in number_1()
number_1()
number_3(D)
number_4()
number_5()
number_6()
number_7()

==+==
FILE: number_1.py
==+==

import re

def number_1():
    """
    For Qs 1 and 2
```

```
    """
    some_string = 'broseph ended Line of text as a string bromigo started sob'.split()
    print(f"reversed {token_reverse(some_string)}")
    print(f"Words beginning with b {letter_b(some_string)}")
    print(f"Words ending with ed {ending_ed(some_string)}")
    print(f"Words beginning with b by regex {letter_b_regex(some_string)}")
    print(f"Words ending with ed by regex {ending_ed_regex(some_string)}")

def token_reverse(token):
    return list(reversed(token))

def letter_b(token):
    return [word for word in token if word.startswith('b')]

def ending_ed(token):
    return [word for word in token if word.endswith('ed')]


def letter_b_regex(token):
    return [x for x in token if re.findall('^b', x)]

def ending_ed_regex(token):
    return [x for x in token if re.findall('ed\Z', x)]


==+==
FILE: number_3.py
==+==

from itertools import permutations

def number_3(D):
    user_input = input(f"Enter a {2 * D} letter word\n")
    perm_obj = permutations(user_input, len(user_input) - 1)
    words = [''.join(val) for val in perm_obj]
    print(words)

==+==
FILE: number_4.py
==+==

import re

def number_4():
    string_with_number = 'jk 34.45 -32 -32.5'
    # (?<!\S) is a negative look behind
    # [+-]? is checking for optional signs
    # (\d+\.\d+|\d+) is saying one or more digits follow by a decimal point and one or more digits
or one or more digits without .
    # (?!\S) is a negative look-ahead to check for non-whitespace
    matches = re.findall('(?<!\S)([+-]?(\d+\.\d+|\d+))(?!\S)', string_with_number)
    numbers = [match[0] for match in matches]
    print(numbers)

==+==
FILE: number_5.py
==+==
```

```
import re

def number_5():
    count_me_string = "i have 234 steps to take on 78 days and such!"
    digits = re.findall(r'\d', count_me_string)
    non_digits = re.findall(r'\D', count_me_string) # non-Digits
    whitespace = re.findall(r'\s', count_me_string)
    words = re.findall(r'\b\w+\b', count_me_string) # \b is setting the boundary
    print(f"number of digits {len(digits)}")
    print(f"number of non-digits: {len(non_digits)}")
    print(f"number of whitespace: {len(whitespace)}")
    print(f"number of words: {len(words)}")

==+==
FILE: number_6.py
==+==

import re

def number_6():
    quote = (
        "Because not I'm a Vanderbilt, suddenly white I'm trash? ",
        "grew I up in Bel Air, Warner. Across the street from Aaron Spelling. ",
        "think I most people would agree that's a lot better than some stinky old Vanderbilt."
    )
    first_quote = switch_search_term(quote[0], "I'm")
    print(first_quote)
    second_quote = switch_search_term(quote[1], "I")
    print(second_quote)
    third_quote = switch_search_term(quote[2], "I")
    print(third_quote)


def switch_search_term(text, search_term):
    # \b\w+\b creates a boundary of the word preceding our search term and uses two capture groups
    pattern = r"(\b\w+\b)\s+(" + re.escape(search_term) + r")"
    # sub is switching the capture groups from the pattern around in the string
    switch = re.sub(pattern, r"\2 \1", text)
    return switch

==+==
FILE: number_7.py
==+==

import re

def number_7():
    quote = "Aragorn: If by my life or death I can protect you I will. You have my sword. Legolas:
And you have my bow. Gimli: And my ax."

    # matches words based on the name using capital letters followed by lowercase and then colon
    # this will make each person's quote a single item in a list
    # since we know the weapons are the last thing each person says we can just use each item of the
list then grab the last word to construct our weapons list
    pattern = r'(?<!^)(?=\b[A-Z][a-z]+:)'
    list_of_quotes = re.split(pattern, quote)
    # each item in the list is a string so split it and grab the last word which is the weapons
```

```
    weapons = [x.strip().split()[-1] for x in list_of_quotes]
    print(weapons)
```

Final Execution Output

Output of the most recent code execution.

```
reversed ['sob', 'started', 'bromigo', 'string', 'a', 'as', 'text', 'of', 'Line', 'ended',
'broseph']
Words beginning with b ['broseph', 'bromigo']
Words ending with ed ['ended', 'started']
Words beginning with b by regex ['broseph', 'bromigo']
Words ending with ed by regex ['ended', 'started']
Enter a 4 letter word
['bat', 'bah', 'bta', 'bth', 'bha', 'bht', 'abt', 'abh', 'atb', 'ath', 'ahb', 'aht', 'tba', 'tbh',
'tab', 'tah', 'thb', 'tha', 'hba', 'hbt', 'hab', 'hat', 'htb', 'hta']
['34.45', '-32', '-32.5']
number of digits 5
number of non-digits: 40
number of whitespace: 10
number of words: 11
Because I'm not a Vanderbilt, suddenly I'm white trash?
I grew up in Bel Air, Warner. Across the street from Aaron Spelling.
I think most people would agree that's a lot better than some stinky old Vanderbilt.
['sword.', 'bow.', 'ax.']
```

Inputs to the Code

Most recent user-typed inputs to the program (STDIN).

```
bath
```

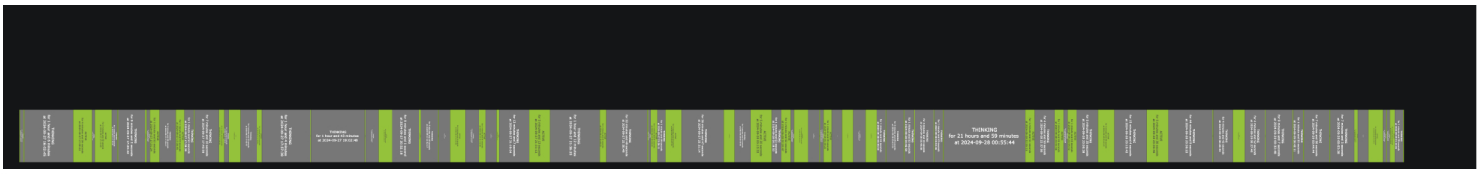
Most Common Error

The most common error that occured during the coding process; "null" (if any) indicates no errors.

```
STDERR:
Traceback (most recent call last):
  File "number_6.py", line 23, in <module>
    number_6()
  File "number_6.py", line 9, in number_6
    find_word(quote[0], "I'm")
  File "number_6.py", line 14, in find_word
    matches = re.split(pattern, text)
  File "/usr/local/python-3.8.1/lib/python3.8/re.py", line 229, in split
    return _compile(pattern, flags).split(string, maxsplit)
  File "/usr/local/python-3.8.1/lib/python3.8/re.py", line 302, in _compile
    p = sre_compile.compile(pattern, flags)
  File "/usr/local/python-3.8.1/lib/python3.8/sre_compile.py", line 764, in compile
    p = sre_parse.parse(p, flags)
  File "/usr/local/python-3.8.1/lib/python3.8/sre_parse.py", line 948, in parse
    p = _parse_sub(source, state, flags & SRE_FLAG_VERBOSE, 0)
  File "/usr/local/python-3.8.1/lib/python3.8/sre_parse.py", line 443, in _parse_sub
    itemsappend(_parse(source, state, verbose, nested + 1,
  File "/usr/local/python-3.8.1/lib/python3.8/sre_parse.py", line 834, in _parse
    p = _parse_sub(source, state, sub_verbose, nested + 1)
  File "/usr/local/python-3.8.1/lib/python3.8/sre_parse.py", line 443, in _parse_sub
    itemsappend(_parse(source, state, verbose, nested + 1,
  File "/usr/local/python-3.8.1/lib/python3.8/sre_parse.py", line 525, in _parse
    code = _escape(source, this, state)
  File "/usr/local/python-3.8.1/lib/python3.8/sre_parse.py", line 426, in _escape
    raise source.error("bad escape %s" % escape, len(escape))
re.error: bad escape \I at position 1
```

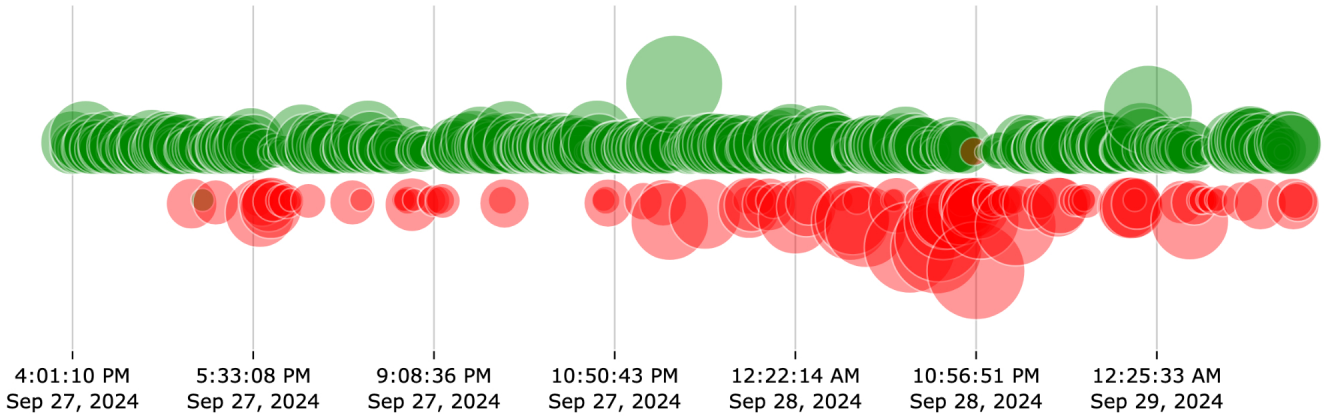
Active Sessions

Active typing sessions while working on the task, shown as green bars.



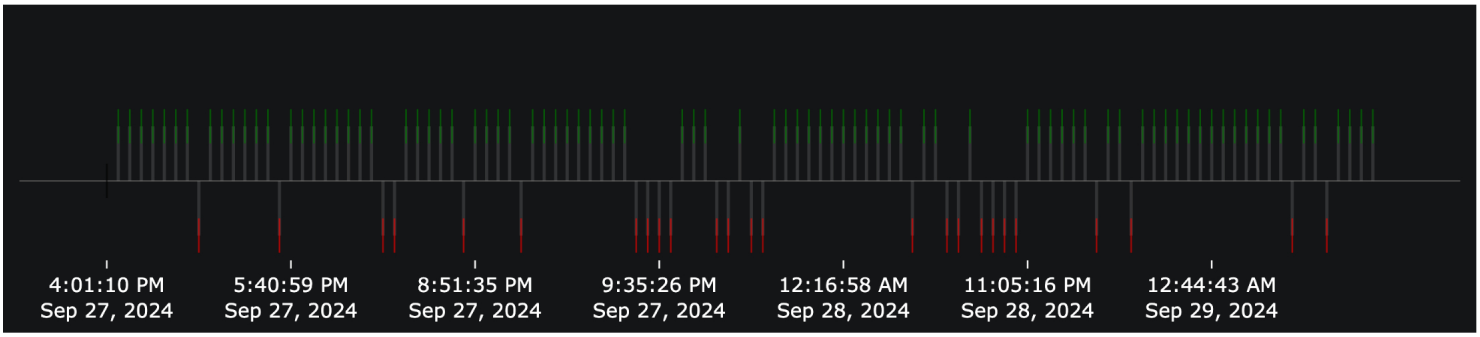
Characters Added and Removed Timeline

Green and red circles show text add and removed; size of the circle shows change in the number of characters.



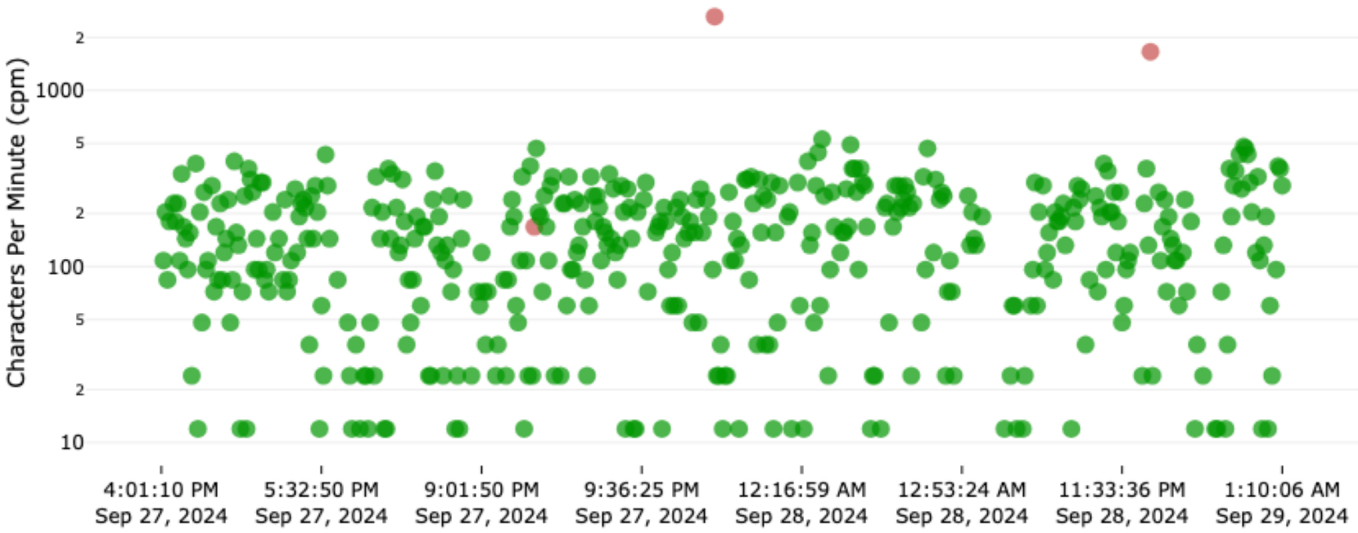
Code Execution Timeline

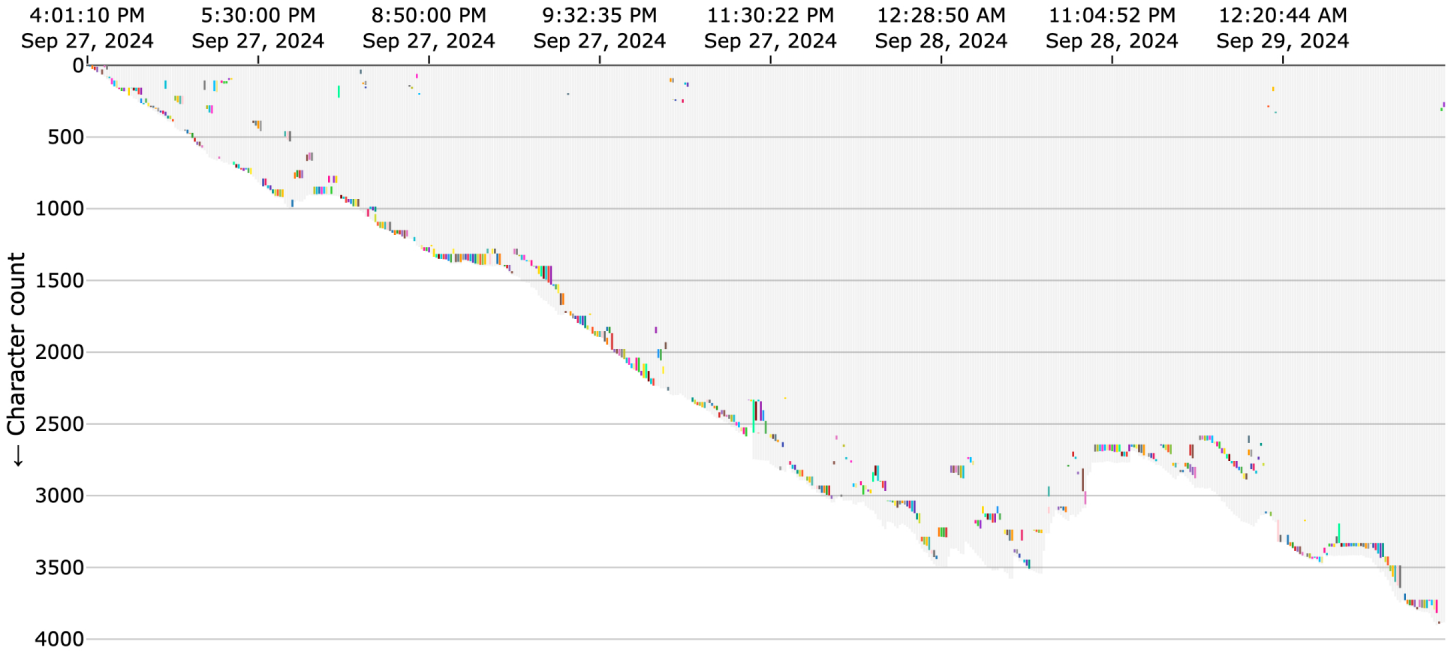
Green and red dots correspond to successful and failed executions.



Typing Speed Timeline

Typing speed can be negative when text is deleted.



Edited Line Locations

Each edited line block is assigned a color; highlighted bars show the line(s) edited at that revision.