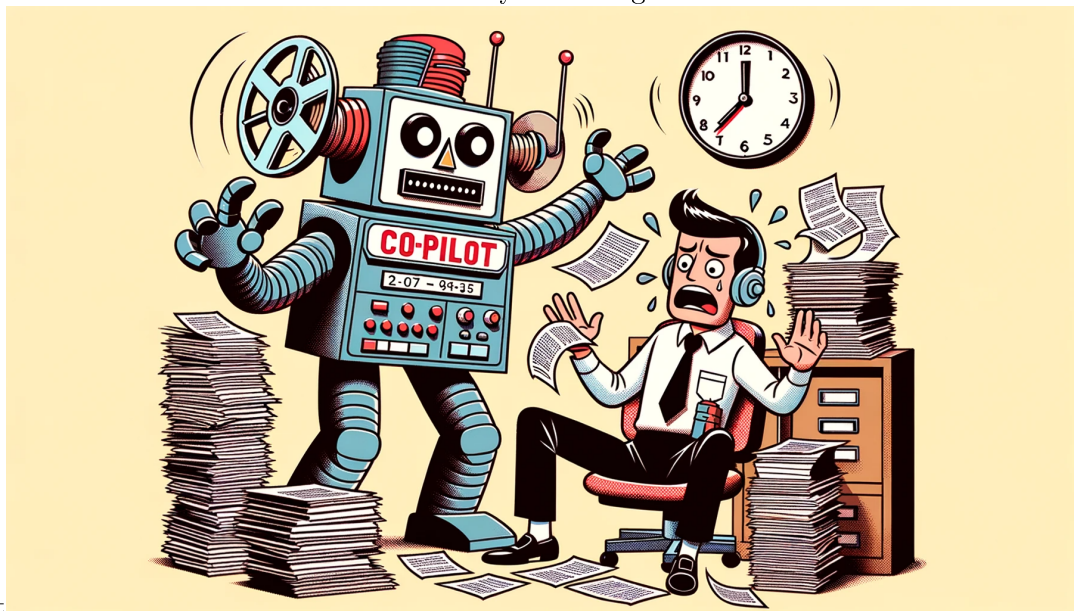


## News

- July 20: VS Code enhancements for Copilot
- Chatbot Bard integrated in replit.com (already has Ghostwriter)
- David Smith - Copilot for R (1hr 20min) (Lander analytics)

## What is GitHub Copilot?

+caption: Illustration in 60s cartoon style showing Copilot  
Illustration in 60s cartoon style showing



Copilot

AI English language coding assistant best for Python, JavaScript, Go. Live since 06/2021, most widely used professional AI developer tool (Yao, 06/2023).

Allows conversion of comments to executable code, autocomplete, and method/function creation. Especially useful for web development.

Based on OpenAI Codex (modified GPT-3) trained on source code in different programming languages. Licensed exclusively to Microsoft.

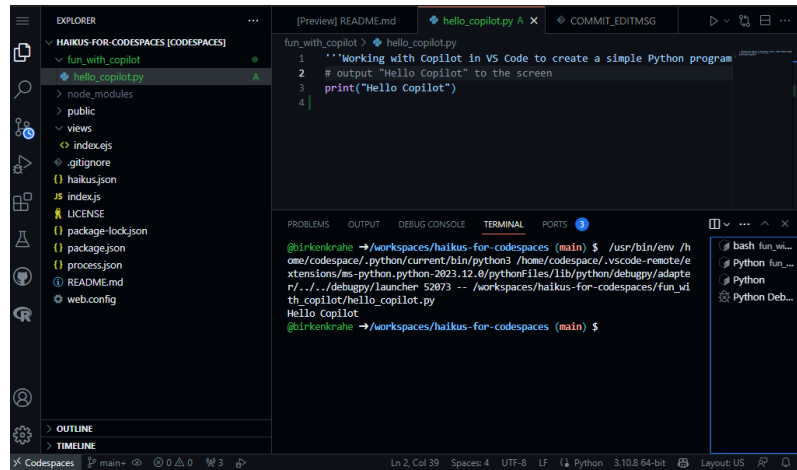
(See also: Wikipedia).

## Setup for Copilot with Github Codespaces

1. Register at [GitHub.com](https://github.com) using your Lyon email address!
2. Get a GitHub Copilot subscription as part of the student developer pack (free for students) - needed for Codespaces and Copilot:
  - Login to GitHub using your Lyon account.
  - Open this page and follow the instructions (you'll need to upload a photo of your student ID).
  - Once submitted, you'll be certified within 4 days.
3. Go to [github.com/github/haikus-for-codespaces](https://github.com/github/haikus-for-codespaces) and click on
  - Use this template
  - Open in a codespace
4. Open the VS Code extension market place (**CTRL + SHIFT + X**), search and install the following extensions:
  - Python and Pylint
  - GitHub CoPilot
5. To test the setup, create a new **File > New File > Python File**, enter the following line:

```
# output "Hello Copilot" to the screen
```

When you press **Enter**, Copilot should now generate this code: `print("Hello Copilot")`, which you can confirm with **TAB** and run with **CTRL + F5** (run without debugging).



6. You may be asked to install the 'Pylint' extension - do it. Pylint is a static code analyzer (grammar checker) for Python.
7. As an alternative, on your own computer, you can download the VS Code editor and load the same extensions there.
8. If you use Emacs, see here for using Copilot inside Emacs (to be preferred by all means over VSCode if you can manage Emacs). There are different packages and you have full control over the interface.

## mtcars example with Copilot

Tasks:

1. load necessary libraries
2. read the CSV file from the repo as a dataframe
3. display the dataframe
4. Sum a column across all records
5. Compute the average of all records for one column
6. Plot a column as a histogram
7. Customize the histogram
8. Save the file

9. Create a scatterplot of two features
10. Draw a colored trendline through the scatterplot

Here is the code after the session with Copilot:

```
# load the mtcars dataset
import pandas as pd
df = pd.read_csv('data/mtcars.csv')
# display the first 5 rows of the dataframe
print(df.head())
# get the sum of the mpg column
print(df['mpg'].sum())
# get the mean of the mpg column
print(df['mpg'].mean())
'''Plot the mpg column as a histogram'''
import matplotlib.pyplot as plt
df['mpg'].plot(kind='hist')
# customize the plot
plt.title('Histogram of mpg')
plt.xlabel('mpg')
plt.ylabel('Frequency')
# save the plot
plt.savefig('histogram.png')
'''plot mpg vs. wt as a scatter plot'''
df.plot(kind='scatter', x='mpg', y='wt')
# customize the plot
plt.title('Scatter plot of mpg vs. wt')
plt.xlabel('mpg')
plt.ylabel('wt')
# save the plot
plt.savefig('scatter.png')
# draw a red trend line using linear regression
import seaborn as sns
sns.regplot(x='mpg', y='wt', data=df, color='red')
# customize the plot
plt.title('Scatter plot of mpg vs. wt')
plt.xlabel('mpg')
plt.ylabel('wt')
# save the plot
plt.savefig('scatter.png')
```

```
# display the plot
plt.show()
```

## pima example with Copilot

The Pima dataset is more interesting because it has a bunch of missing values that are coded as 0s creating nonsensical patient data.

Task:

1. upload a CSV file from URL into dataframe
2. rename the columns and delete the old column names
3. visualize the missing data in the dataframe as a heatmap
4. customize and save the plot.

Code transcript from Copilot:

```
'''
Upload a CSV file from a URL into a dataframe. Name the dataframe "pima".
The URL is: https://raw.githubusercontent.com/plotly/datasets/master/diabetes.csv
Name columns as follows: 'pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', '
'''

import pandas as pd
# load the data
url = 'https://raw.githubusercontent.com/plotly/datasets/master/diabetes.csv'
pima = pd.read_csv(url, names=['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', '
# print the first 5 rows of the dataframe
print(pima.head(5))
# remove the first row of the dataframe
pima = pima.iloc[1:]
# print the first 5 rows of the dataframe again
print(pima.head(5))

'''

Visualize the missing data in the dataframe as a heatmap.
'''

import matplotlib.pyplot as plt
import seaborn as sns
# create a heatmap of the missing data using corr()
sns.heatmap(pima.corr(), annot=True)
```

```
# customize the plot
plt.title('Heatmap of missing data')
plt.xlabel('Columns')
plt.ylabel('Columns')
# save the plot
plt.savefig('heatmap.png')
```

## Extended example: NFL data

### With GitHub Copilot

#### Step 1: Aaron Rodgers passing yards 2019-2022

1. Download and extract the csv file from Kaggle to your PC: [kaggle.com/datasets/dtrade84/nfl-offensive-stats-2019-2022](https://www.kaggle.com/datasets/dtrade84/nfl-offensive-stats-2019-2022)
2. Create a directory `data` in your current Copilot Python working directory and upload the CSV file there (right click on the directory name and select `Upload ...`).
3. You may be asked to install the "Rainbow CSV" extension. This is not a political statement! It's only extended syntax highlighting.



```
data > nfl_offensive_stats.csv
1 game_id,player_id,position ,player,team,pass_cmp,pass_att,pass_yds,pas
2 201909050chi,RodgAa00,QB,Aaron Rodgers,GNB,18,30,203,1,0,5,37,47,91.4,
3 201909050chi,JoneAa00,RB,Aaron Jones,GNB,0,0,0,0,0,0,0,0,0,13,39,0,9,1
4 201909050chi,ValdMa00,WR,Marquez Valdes-Scantling,GNB,0,0,0,0,0,0,0,0,
5 201909050chi,AdamDa01,WR,Davante Adams,GNB,0,0,0,0,0,0,0,0,0,0,0,8
6 201909050chi,GrahJi00,TE,Jimmy Graham,GNB,0,0,0,0,0,0,0,0,0,0,0,5,
7 201909050chi,DaviTr03,WR,Trevon Davis,GNB,0,0,0,0,0,0,0,0,0,0,0,1,
8 201909050chi,TonyRo00,TE,Robert Tonyan,GNB,0,0,0,0,0,0,0,0,0,0,0,1
9 201909050chi,WillJa06,RB,Jamaal Williams,GNB,0,0,0,0,0,0,0,0,0,5,0,0,5
10 201909050chi,LewiMa00,TE,Mercedes Lewis,GNB,0,0,0,0,0,0,0,0,0,0,0,0,
11 201909050chi,TrubMi00,QB,Mitchell Trubisky,CHI,26,45,228,0,1,5,20,27,6
12 201909050chi,DaviMi01,RB,Mike Davis,CHI,0,0,0,0,0,0,0,0,0,5,19,0,8,7,6
13 201909050chi,MontDa01,RB,David Montgomery,CHI,0,0,0,0,0,0,0,0,0,6,18,0
14 201909050chi,RobiAl02,WR,Allen Robinson,CHI,0,0,0,0,0,0,0,0,0,0,0,0,
15 201909050chi,CohTa00,RB,Tarik Cohen,CHI,0,0,0,0,0,0,0,0,0,0,0,0,10,
```

4. Create a new Python file and name it `nfl_data.py`.
5. Write the prompt in a docstring:

```
'''
open the csv file called ./data/nfl_data.csv and
```

```
read in the csv data from the file.
'''
```

6. Entering TAB repeatedly generated this output, which generates an error because the `csv` library has not been imported:

```
'''
open the csv file called ./data/nfl_data.csv and read in
the csv data from the file
'''
def read_nfl_data():
    with open('./data/nfl_data.csv', 'r') as nfl_data:
        reader = csv.reader(nfl_data)
        nfl_data = list(reader)
    return nfl_data    # return the data as a list of lists
```

7. This version works but it suggests printing the entire file without being asked; also, it did not pick up that the file path needed to be complete (relative, including the directory, or absolute, including the whole path from `/workspaces`):

```
import csv    # import the csv module to read in the csv file

# open the csv file called nfl_offensive_stats.csv
with open('fun_with_copilot/nfl_offensive_stats.csv',\
    encoding="utf-8") as csv_file:
    # read in the csv data from the file
    csv_reader = csv.reader(csv_file, delimiter=',')
    # skip the header row
    next(csv_reader)
    # loop through each row in the csv file
    for row in csv_reader:
        # print out the row
        print(row)
```

8. This is the version suggested in the book (except that Copilot missed the `import csv` command and the path to the file).

```
import csv
with open('data/nfl.csv', 'r') as nfl_file:
    nfl_reader = csv.reader(nfl_file)
    nfl_data = list(nfl_reader)
```

9. The debugger (Pylint) suggests to specify the text encoding as parameter `'encoding="utf-8"` - Copilot suggest to print out the first row with the column headers `print(nfl_data[0])`:

```
import csv
with open('data/nfl.csv','r') as f:
    reader = csv.reader(f)
    nfl = list(reader)
    #print(nfl[0])
```

10. Let's try it with pandas:

```
import pandas as pd
df = pd.read_csv('./data/nfl.csv')
print(df.head())
```

11. Next prompt:

```
import csv
with open('data/nfl.csv','r') as f:
    reader = csv.reader(f)
    nfl = list(reader)
    #print(nfl[0])
'''
```

In the data we just read in, the fourth column is the player's name. and the 8th column is the number of passing yards for that player. Get the sum of yards from column 8 where the 4th column value is "Aaron Rodgers".

```
'''
total_passing_yards = 0
for row in nfl:
    if row[3] == "Aaron Rodgers":
        total_passing_yards += int(row[7])
    print(total_passing_yards)
```

12. Summary on Copilot:

- (a) It's a powerful tool - basic human prompts can produce correct code and output for a basic data analysis task.
- (b) Breaking problems into small tasks is important - significantly increases the likelihood of Copilot generating the right code.



- (c) We still need to understand code - see e.g. the attempt at using Bard below: the data set is accepted but the code is wrong and because the errors relate to a knowledge of the dataset and the syntax, they'd be hard to fix without code experience.
- (d) Testing is important.

## Step 2: Quarterback performance comparison

- Copilot@Codespaces alone could not solve this. I had to get it started using ChatGPT's GPT-4 with Code Interpreter (share link).
- Copilot then solved some small changes, like sorting the results in descending order and printing only one result per line.
- LLama (at perplexity.ai) did not get it right either.
- Final code (nfl\_data2.py):

```
'''
Code obtained from ChatGPT/GPT-4/Code Interpreter:
https://shareg.pt/h1hmCjr
'''

import pandas as pd

# Read the CSV file
nfl_data = pd.read_csv('./data/nfl.csv')
nfl_data.head()

# Convert the DataFrame to a list of dictionaries, each representing a row
data = nfl_data.to_dict('records')
data[:5] # Display the first 5 rows

# Convert the DataFrame to a list of dictionaries, each representing a row
data = nfl_data.to_dict('records')
data[:5] # Display the first 5 rows

# Initialize a dictionary to store the passing yards for each quarterback
qb_passing_yards = {}

# Loop over the rows in the data
for row in data:
```

```

# Check if the player's position is "QB"
if row['position '] == 'QB':
    # If the player is not yet in the dictionary, add them with their passing
    # If they are already in the dictionary, add their passing yards to their
    qb_passing_yards[row['player']] = qb_passing_yards.get(row['player'], 0) +
    '''
    This addition by Copilot:
    sort the qb_passing_yards dictionary by the values (passing yards) in descending
    '''
    sorted_qb_passing_yards = sorted(qb_passing_yards.items(), key=lambda x: x[1], reverse=True)
    '''
    print the top 10 quarterbacks by passing yards one per line and their passing yards
    '''
for i in range(10):\
    print(sorted_qb_passing_yards[i][0], sorted_qb_passing_yards[i][1])

```

```

Patrick Mahomes 16132
Tom Brady 15876
Aaron Rodgers 13852
Josh Allen 13758
Derek Carr 13271
Matt Ryan 13015
Matthew Stafford 12657
Kirk Cousins 12503
Russell Wilson 12211
Jared Goff 12164

```

- The final experiment, to exclude Tom Brady, lead to more convoluted code by Copilot. The answer is:

```

for player in sorted(qb_passing_yards,
                    key=passing_yards.get,
                    reverse=True):
    if player != "Tom Brady":
        print(player, passing_yards[player])

```

### Step 3: plotting the stats

- Establishing the player/summed yards dictionary `passing_yards`:

```

# import and save csv data as list
import csv
with open('./data/nfl.csv','r') as f:
    data = list(csv.reader(f))

# create dictionary to hold player name and passing yards
passing_yards = {}

# loop through data
for row in data:
    if row[2] == 'QB': # checks for quarterback
        if row[3] in passing_yards: # checks for existing name
            passing_yards[row[3]] += int(row[7]) # add yards
        else:
            passing_yards[row[3]] = int(row[7]) # add item

```

- Plot the players by their number of passing yards only for players with more than 4,000 passing yards.

```

# import and save csv data as list
import csv
with open('./data/nfl.csv','r') as f:
    data = list(csv.reader(f))

# create dictionary to hold player name and passing yards
passing_yards = {}

# loop through data
for row in data:
    if row[2] == 'QB': # checks for quarterback
        if row[3] in passing_yards: # checks for existing name
            passing_yards[row[3]] += int(row[7]) # add yards
        else:
            passing_yards[row[3]] = int(row[7]) # add item

import matplotlib.pyplot as plt
import numpy as np

```

```

# create list of players and of passing yards
players = []
yards = []

# for each player in the dictionary
for player in passing_yards:
    # for players with more than 4,000 yards
    if passing_yards[player] > 4000:
        players.append(player) # add player to player list
        yards.append(passing_yards[player]) # add yards to yard list

# create a numpy array of passing yards and of players
yards = np.array(yards)
players = np.array(players)

# sort players by passing yards
print(np.argsort(yards))
players = players[np.argsort(yards)]

# sort passing yards
yards = np.sort(yards)

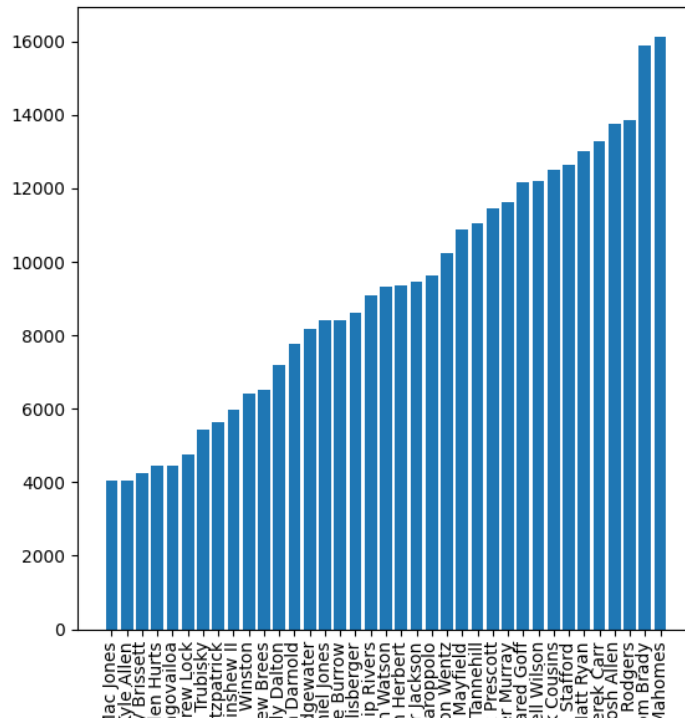
# create figure
plt.figure()

# create bar chart of players and their passing yards
plt.bar(players,yards)

# rotate x axis labels (names are too long)
plt.xticks(rotation=90)

# show the plot
plt.show()

```



## Extended NFL example with Code Interpreter

- Three prompts are sufficient for ChatGPT's "Code Interpreter" plugin. [Link to the shared conversation.](#)
- Step 1: "Read the csv file and save it as a list."

```
import pandas as pd

# Load the CSV file into a pandas dataframe
df = pd.read_csv("./data/nfl.csv")

# Convert the dataframe to a list of dictionaries,
# which is more similar to a list format
data_list = df.to_dict('records')
#print(data_list[:5]) # display the first 5 records for review
```

- Step 2: "Create a dictionary whose key is the name of every player whose position is a quarterback (position == 'QB') and whose value is the sum of all passing yards (pass\_yds) of that player. Sort the dictionary and print the top 10 players with their passing yard count."

```
import pandas as pd

# Load the CSV file into a pandas dataframe
df = pd.read_csv("./data/nfl.csv")

# Convert the dataframe to a list of dictionaries,
# which is more similar to a list format
data_list = df.to_dict('records')
#print(data_list[:5]) # display the first 5 records for review
# Filter the list for records where the position is 'QB'
qb_data = \
    [record for record in data_list if record['position'] == 'QB']

# Initialize an empty dictionary to hold the sum of passing yards for each QB
qb_pass_yds = {}

# Iterate over the QB data
for record in qb_data:
    # If the player's name is not already a key in the dictionary, add it with the
    if record['player'] not in qb_pass_yds:
        qb_pass_yds[record['player']] = record['pass_yds']
    # If the player's name is already a key in the dictionary, add the current
    else:
        qb_pass_yds[record['player']] += record['pass_yds']

# Sort the dictionary by values (passing yards) in descending order and get the first 10
sorted_qb_pass_yds = dict(sorted(qb_pass_yds.items(),
                                key=lambda item: item[1],
                                reverse=True)[:10])

#print(sorted_qb_pass_yds)

{'Patrick Mahomes': 16132, 'Tom Brady': 15876, 'Aaron Rodgers': 13852, 'Josh Allen': 13852, 'Drew Brees': 13852, 'Russell Wilson': 13852, 'Lamar Jackson': 13852, 'Matt Ryan': 13852, 'Deshaun Watson': 13852, 'Dak Prescott': 13852}
```

- Step 3: "Plot the players by their number of passing yards only for players with more than 4,000 passing yards."

```

import pandas as pd

# Load the CSV file into a pandas dataframe
df = pd.read_csv("./data/nfl.csv")

# Convert the dataframe to a list of dictionaries,
# which is more similar to a list format
data_list = df.to_dict('records')
#print(data_list[:5]) # display the first 5 records for review
# Filter the list for records where the position is 'QB'
qb_data =\
    [record for record in data_list if record['position'] == 'QB']

# Initialize an empty dictionary to hold the sum of passing yards for each QB
qb_pass_yds = {}

# Iterate over the QB data
for record in qb_data:
    # If the player's name is not already a key in the dictionary, add it with the
    if record['player'] not in qb_pass_yds:
        qb_pass_yds[record['player']] = record['pass_yds']
    # If the player's name is already a key in the dictionary, add the current
    else:
        qb_pass_yds[record['player']] += record['pass_yds']

# Sort the dictionary by values (passing yards) in descending order and get the first 10
sorted_qb_pass_yds = dict(sorted(qb_pass_yds.items(),
                                key=lambda item: item[1],
                                reverse=True)[:10])

#print(sorted_qb_pass_yds)
import matplotlib.pyplot as plt

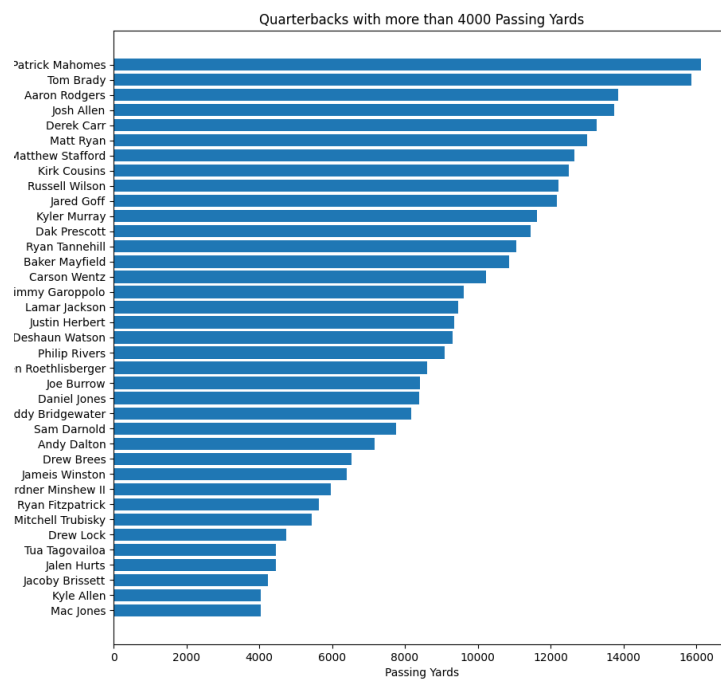
# Filter the dictionary to include only players
# with more than 4000 passing yards
filtered_qb_pass_yds = {player: yds for player,
                        yds in qb_pass_yds.items() if yds > 4000}

# Sort the filtered dictionary by values
# (passing yards) in descending order

```

```
sorted_filtered_qb_pass_yds =\
    dict(sorted(filtered_qb_pass_yds.items(),
               key=lambda item: item[1],
               reverse=True))

# Plot the players by their number of passing yards
plt.figure(figsize=(10,10))
plt.barh(list(sorted_filtered_qb_pass_yds.keys()),
         list(sorted_filtered_qb_pass_yds.values()))
plt.xlabel('Passing Yards')
plt.title('Quarterbacks with more than 4000 Passing Yards')
plt.gca().invert_yaxis()
plt.show()
```





## Extended NFL ex. (step 1) in Bard, Claude, ChatGPT

- Prompt: "For the dataset <https://www.kaggle.com/datasets/dtrade84/nfl-offensive-stats-2019-2022>, give me the number of passing yards for Aaron Rodgers."
- The first attempt returns the wrong numerical result but the correct code (apart from the wrong column names - 'Player' instead of 'player', and 'Passing Yards' instead of 'pass\_yds', and offers a nonsensical explanation for its error:

I checked the dataset again and found that the column for passing yards is actually called Passing Yards. I had misread it as Passing Yards **Total**.

- The code works:

```
import pandas as pd

# Load the dataset
df = pd.read_csv('./data/nfl.csv')

# Select the rows for Aaron Rodgers
rodgers_data = df[df['player'] == 'Aaron Rodgers']

# Get the number of passing yards
passing_yards = rodgers_data['pass_yds'].sum()

print(passing_yards)
```

- Asked to reproduce the result "without pandas, with standard library Python only, Bard replies with incorrect code, mixing up the column vector indices, and doesn't even get it right after repeated corrections, until it is finally instructed to use `row[3]` for the player and `row[7]` for the passing yards.

```
import csv

# Open the dataset in read mode
with open('./data/nfl.csv', 'r') as csvfile:
```

```

# Create a reader object
reader = csv.reader(csvfile, delimiter=',')

# Skip header
next(reader, None)

# Initialize a counter for passing yards
passing_yards = 0

# Iterate over the rows in the dataset
for row in reader:

    # Check if the player is Aaron Rodgers
    if row[3] == 'Aaron Rodgers':

        # Increment the passing yards counter
        passing_yards += int(row[7])

# Print the total number of passing yards
print(passing_yards)

```

- Claude cannot load the dataset (too large).
- ChatGPT does a good job right away with the full dataset uploaded:

```

import pandas as pd

# Load the dataset
nfl_data = pd.read_csv("/mnt/data/nfl.csv")

# Check the first few rows of the dataset to understand its structure
nfl_data.head()

```

- It applies the accurate reasoning to solve the problem:

The dataset contains numerous columns, but for your specific request, we are interested in the **'player'** and **'pass\_yds'** columns. To get the total passing yards for Aaron Rodgers, I will filter the dataset for rows where the **'player'** column matches **'Aaron Rodgers'**, and then sum up the **'pass\_yds'** column. Let's proceed with that.

- The code results in the right number 13852 and it is arguably the most elegant, using nested indexing by row and column:

```
# Filter the data for Aaron Rodgers and calculate the sum of his passing yards
aaron_rodgers_pass_yds = nfl_data[nfl_data['player'] == 'Aaron Rodgers']['pass_yds']
aaron_rodgers_pass_yds
```

- Let's check it here (with a few changes):

```
import pandas as pd
# store CSV data in data frame
df = pd.read_csv("./data/nfl.csv")
# print data frame dimension attribute
print(df.shape)
# print first few rows of data frame
print(df.head())
# save all records with Aaron Rodgers' data
aaron_rodgers_df = df[df['player'] == 'Aaron Rodgers']
aaron_rodgers_pass_yds = aaron_rodgers_df['pass_yds'].sum()
print(aaron_rodgers_pass_yds)
```

## Extended NFL example manually

### Data import

- Data import with standard library:
  1. Import CSV file using `csv`.
  2. The function `csv.reader` returns an iterator
  3. The `list` function turns the iterator in a list whose 1st element contains the header names.

```
import csv
data = 'c:/Users/birkenkrahe/Documents/GitHub/admin/RoamNotes/data/nfl.csv'
file = open(data, 'r')
reader = csv.reader(file)
nfl_data = list(reader)
file.close()
#print(nfl_data[0]) # headers
```

- Code to understand the structure of the list:

```

import csv
data = 'c:/Users/birkenkrahe/Documents/GitHub/admin/RoamNotes/data/nfl.csv'
file = open(data, 'r')
reader = csv.reader(file)
nfl_data = list(reader)
file.close()
#print(nfl_data[0]) # headers
print(len(nfl_data))
print(nfl_data[0])
print(nfl_data[0][3])
for i in range(5):
    print(nfl_data[i])

```

- Note: when looping over the `nfl_data` by row, the header row is lost somehow.
- Data import with pandas library as data frame:

```

import pandas as pd
data = 'c:/Users/birkenkrahe/Documents/GitHub/admin/RoamNotes/data/nfl.csv'
df = pd.read_csv(data)
#print(df.columns)

```

- You can also get a list from a data frame with `pd.tolist`:

```

import pandas as pd
data = 'c:/Users/birkenkrahe/Documents/GitHub/admin/RoamNotes/data/nfl.csv'
df = pd.read_csv(data)
#print(df.columns)
print(df.values.tolist()[0][3])

```

### Step 1: Aaron Rodgers' passing yards result

- Get the sum of passing yards (4th column) played by Aaron Rodgers (player names in 8th column) - the loop goes over all list items checking each item's 4th position for equality with the player 'Aaron Rodgers' and adding up (for that item/row) the values of the 8th position (passing yards):

```

import csv
data = 'c:/Users/birkenkrahe/Documents/GitHub/admin/RoamNotes/data/nfl.csv'

```

```

file = open(data, 'r')
reader = csv.reader(file)
nfl_data = list(reader)
file.close()
#print(nfl_data[0]) # headers
total_passing_yards = 0
for row in nfl_data:
    if row[3] == 'Aaron Rodgers':
        total_passing_yards += int(row[7])
        print(total_passing_yards)

```

## Step 2: Comparison of top quarterbacks

- Load data as list:

```

import csv
csv_data = './data/nfl.csv'
file = open(csv_data, 'r')
nfl_data = list(csv.reader(file))
#print(len(nfl_data)) # number of rows
#print(len(nfl_data[0])) # number of columns

```

- The 3rd column in data is player position, the fourth column is the player, and the 8th column is the passing yards. For each player whose position in column 3 is "QB", determine the sum of yards from column 8.
- This code prints player name and yards for each game:

```

import csv
csv_data = './data/nfl.csv'
file = open(csv_data, 'r')
nfl_data = list(csv.reader(file))
#print(len(nfl_data)) # number of rows
#print(len(nfl_data[0])) # number of columns
for row in nfl_data:
    if row[2] == 'QB':
        print(f'Player: {row[3]}, Yards: {row[7]}')

```

- We want to store the sum of values in row[7] as `sum_of_yards` for each player in row[3].

- Create a similar but shorter list:
  1. loop over list
  2. when you find an 'A' item, store the name and the value in a dictionary `res`
  3. if the name is already in the dictionary, append another

```
# lst is a list of lists
lst = [['A','Joe','10'],
        ['A','Jim','100'],
        ['C','John','1000'],
        ['A','Joe','11'],
        ['C','Jill','1000'],
        ['A','Jim','101']]
res = {}
cnt = 0
# iterate over list items
for row in lst:
    if row[0]=='A':
        # add dict item for each row if item not in there yet
        if row[1] not in res:
            res[row[1]] = [row[2]] # new key and value
        else:
            res[row[1]].append(row[2]) # old key, append value
            # iterate over dictionary and sum converted values changing res
for key in res:
    res[key] = sum(int(x) for x in res[key])
    # print one item per line
for key, value in res.items():
    print(f'{key}: {value}')
```

- Now, for the NFL data: `row[3]` holds the player name, `row[7]` the yards

```
import csv
csv_data = './data/nfl.csv'
file = open(csv_data,'r')
nfl_data = list(csv.reader(file))
#print(len(nfl_data)) # number of rows
```

```

#print(len(nfl_data[0])) # number of columns
res = {}
# iterate over rows
for row in nfl_data:
    # if player position is quarterback
    if row[2] == 'QB':
        # if name is not in dictionary yet, add it with values
        if row[3] not in res:
            res[row[3]] = [row[7]]
            # if name already in dictionary,
            # append values to existing key
        else:
            res[row[3]].append(row[7])
            # sum yards for each (unique) player
for key in res:
    res[key] = sum(int(x) for x in res[key])
    # print one item per line
for key, value in res.items():
    print(f'{key}: {value}')

```

### Step 3: Plotting results

- Plot the players by their number of passing yards only for players with more than 4,000 passing yards.

## Summary

You installed Python and VSCode and set up Copilot so you are able to work along with the book and start writing code yourself!

- The VSCode interface has areas for file management, code editing, and running code that will be used throughout the book.
- Prompts are how we tell Copilot to generate code and, when written carefully, can be a highly effective way of creating software.
- Data analysis is a common programming task and .csv files are a common way for storing data to be processed by computers.
- Copilot may generate code that requires you to install additional Python modules.

- Copilot is a powerful tool that is capable of producing code that is as sophisticated (or more) as that produced by college students finishing their first programming course.

## Observations:

- Copilot is not always able to find the right (or the best) result
- OpenAI's Code Interpreter, equipped with good prompts, is faster and more accurate.
- Copilot presents alternatives, which OpenAI does not (unless asked)

## GitHub Copilot for R - First Impressions

- ChatGPT/VoxScript summary:

The video titled "GitHub Copilot for R - First impressions" by the channel ggnot2 is a walkthrough and first impressions review of GitHub Copilot, an AI tool designed to assist with coding.

The video starts with the presenter explaining her intent to learn and use GitHub Copilot for the first time. She signs up for a free trial of GitHub Copilot and sets it up on Visual Studio Code (VS Code), as RStudio, her usual IDE, does not support GitHub Copilot.

She then tests GitHub Copilot by creating a function in R. The AI tool suggests code snippets as she types, which she finds impressive. She also appreciates that GitHub Copilot is integrated into the IDE, which saves her time from switching between the browser and the IDE, a problem she encountered while using ChatGPT.

She then tries to create a Shiny app, a web application framework for R. GitHub Copilot successfully suggests the necessary code, but she encounters some issues running the app in VS Code. She also notes that GitHub Copilot automatically suggests the `runApp()` function, which she had initially forgotten to include.

Towards the end of the video, she discusses the possibility of integrating GitHub Copilot into RStudio. She finds a



GitHub issue requesting this functionality, but it appears that it's not currently planned. An alternative mentioned is GPT-3 Studio, which offers similar functionality but is not the same as GitHub Copilot.

She concludes the video by discussing the cost of using GitHub Copilot, which she finds reasonable at \$100 per year, especially considering the time it could save for daily programming. She compares this to the pay-as-you-go model of the OpenAI API, which could potentially be more expensive depending on usage. However, she appreciates that GitHub Copilot has a fixed cost, allowing unlimited use without worrying about the cost per use.

She ends the video by asking viewers for their thoughts and experiences with both GitHub Copilot and ChatGPT, and which they would recommend.

## References

One Year On, GitHub Copilot Adoption Soars, by Deborah Yao, AI Business.  
URL: [aibusiness.com](https://aibusiness.com).