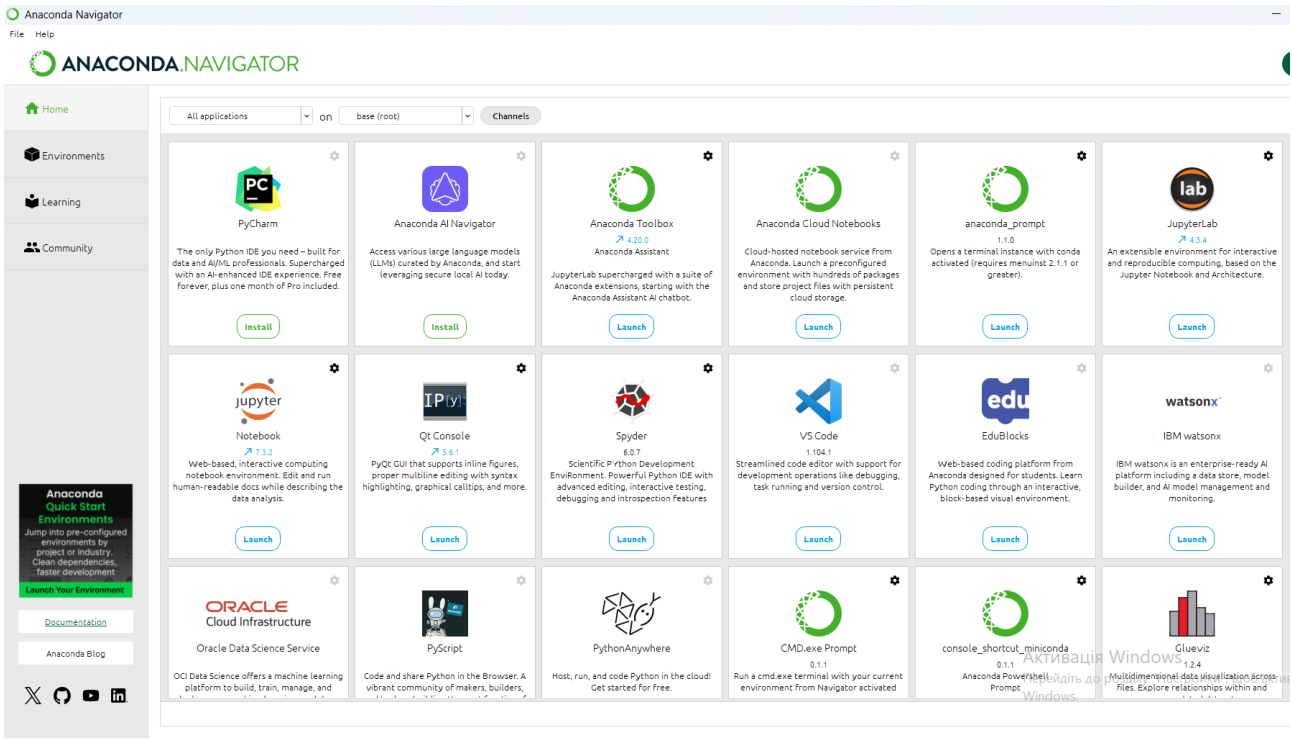
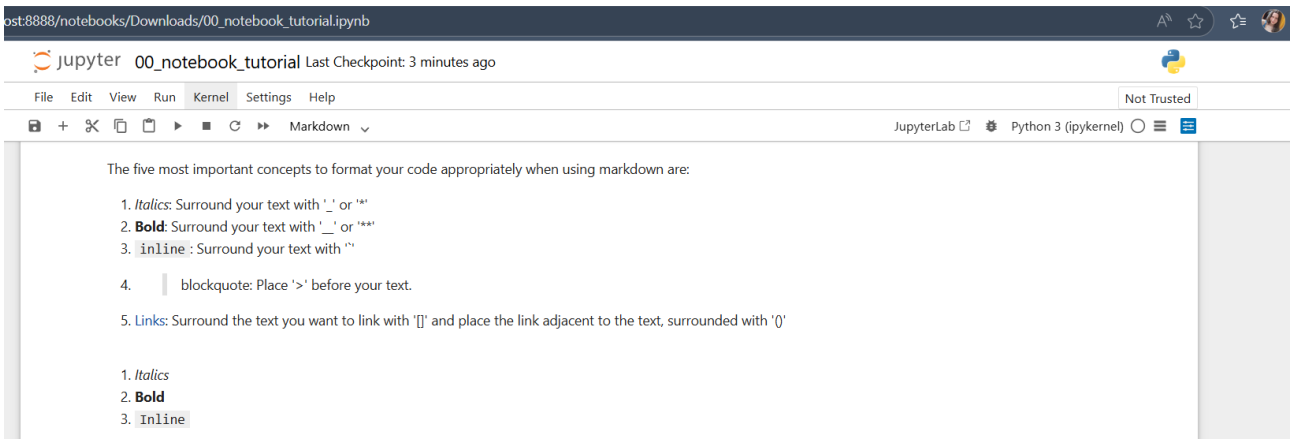
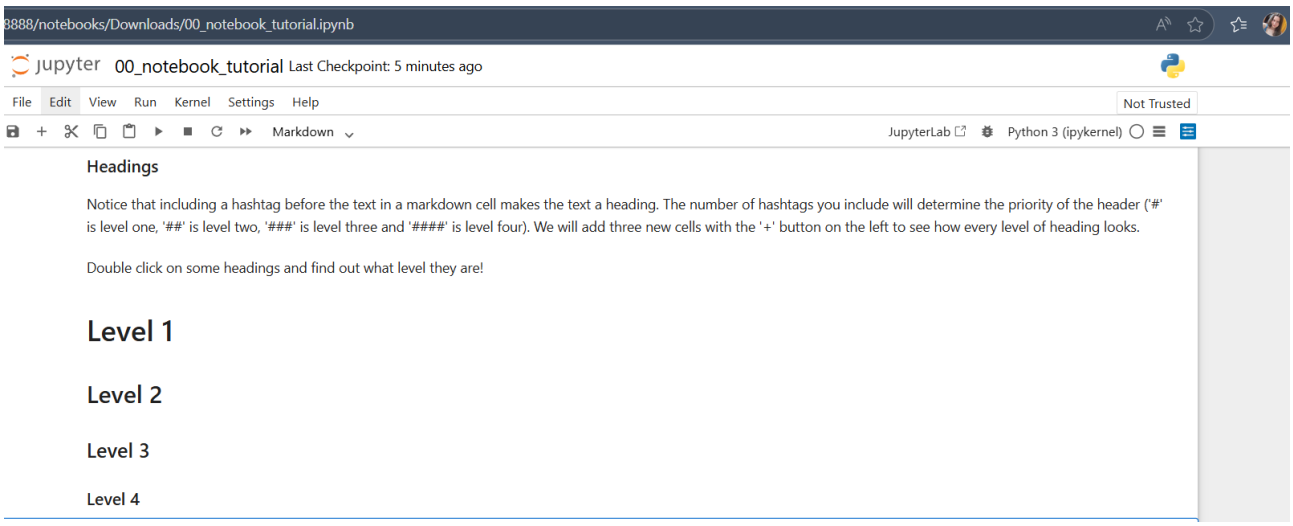
Homework #2. Python basics.

Student ID - 17. Sofiia Zakharuk, CS-2.

Task #1 (35 min spent)

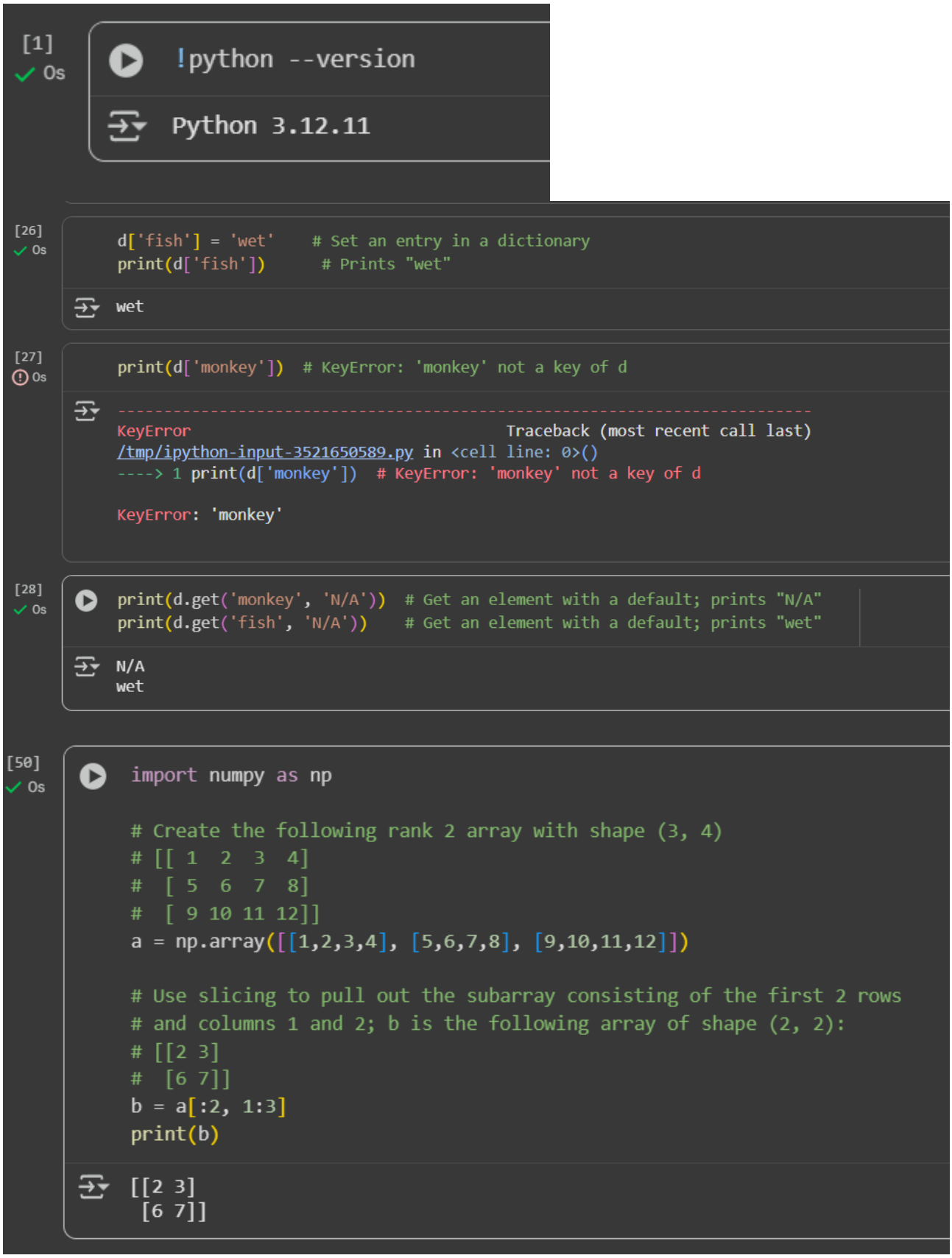






Task #2 (230 min spent)

1.



```
[1]
✓ 0s
!python --version
Python 3.12.11
```

```
[26]
✓ 0s
d['fish'] = 'wet'    # Set an entry in a dictionary
print(d['fish'])      # Prints "wet"
wet
```

```
[27]
! 0s
print(d['monkey'])  # KeyError: 'monkey' not a key of d
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipython-input-3521650589.py in <cell line: 0>()
----> 1 print(d['monkey'])  # KeyError: 'monkey' not a key of d

KeyError: 'monkey'
```

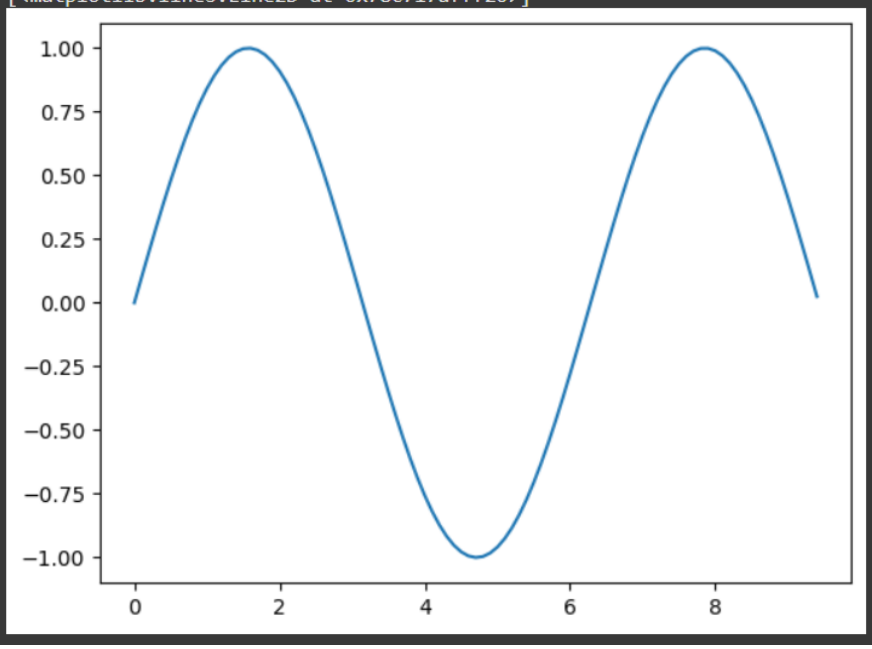
```
[28]
✓ 0s
print(d.get('monkey', 'N/A'))  # Get an element with a default; prints "N/A"
print(d.get('fish', 'N/A'))    # Get an element with a default; prints "wet"
N/A
wet
```

```
[50]
✓ 0s
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]
print(b)
[[2 3]
 [6 7]]
```

[87]
✓ 0s

```
# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)

# Plot the points using matplotlib
plt.plot(x, y)
```

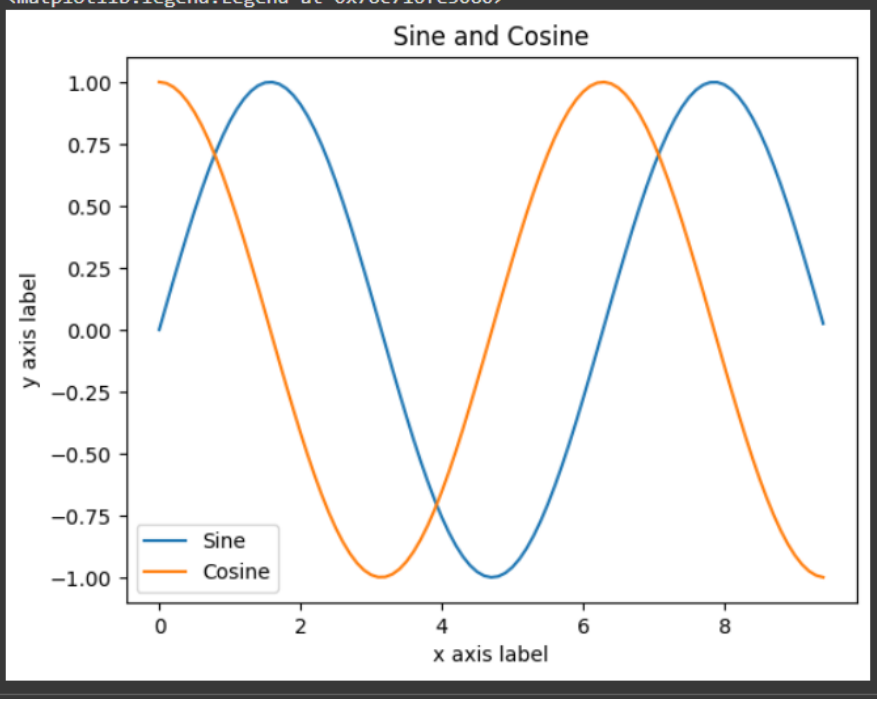
```
[<matplotlib.lines.Line2D at 0x78c717afff20>]
```



[88]
✓ 0s

```
y_sin = np.sin(x)
y_cos = np.cos(x)

# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
```

```
<matplotlib.legend.Legend at 0x78c716fe3080>
```



2.

```
[2]
✓ 0s
```

```
import numpy as np
import pandas as pd
```

```
[10]
✓ 0s
```

```
my_data = np.random.randint(low=0, high=101, size=(3, 4))

my_column_names = ['Eleanor', 'Chidi', 'Tahani', 'Jason']

df = pd.DataFrame(data=my_data, columns=my_column_names)
print(df)

print("Row #1 of the Eleanor column is %d\n\n" % df['Eleanor'][1])

df['Janet'] = df['Tahani'] + df['Jason']
print(df)
```

```
   Eleanor  Chidi  Tahani  Jason
0       58      0      60     43
1       39     70      37     64
2       74     82      67     39
Row #1 of the Eleanor column is 39


   Eleanor  Chidi  Tahani  Jason  Janet
0       58      0      60     43    103
1       39     70      37     64    101
2       74     82      67     39    106
```

Task #3 (145 min spent) https://colab.research.google.com/drive/1kb4pBRr9q-ngnD0jiSPT9Ix_52ysaH_u?usp=sharing – link

1. Working with Lists

```
[2]
✓ 0s
```

```
def extract_and_apply(l, p, f):
    return [f(x) for x in l if p(x)]
```

```
[3]
✓ 0s
```

```
# Example:
l = [1, 2, 3, 4]
p = lambda x: x % 2 == 0
f = lambda x: x
extract_and_apply(l, p, f)

# Expected output:
# [2, 4]
```

```
[2, 4]
```

[4] ✓ 0s

```python
def concatenate(seqs):
    return [x for seq in seqs for x in seq]
```

[5] ✓ 0s

```python
# Example:
concatenate([[1, 2], [3, 4]])

# Expected output:
# [1, 2, 3, 4]
```

```
[1, 2, 3, 4]
```

[6] ✓ 0s

```python
# Example:
concatenate(["abc", (0, [0])])

# Expected output:
# ['a', 'b', 'c', 0, [0]]
```

```
['a', 'b', 'c', 0, [0]]
```

[11] ✓ 0s

```python
def transpose(matrix):
  rows = len(matrix)
  columns = len(matrix[0])

  transposed = [[0] * rows for _ in range(columns)]

  for i in range(rows):
        for j in range(columns):
            transposed[j][i] = matrix[i][j]
  return transposed
```

[12] ✓ 0s

```python
# Example:
transpose([[1, 2, 3]])

# Expected output:
# [[1], [2], [3]]
```

```
[[1], [2], [3]]
```

[13] ✓ 0s

```python
# Example:
transpose([[1, 2], [3, 4], [5, 6]])

# Expected output:
# [[1, 3, 5], [2, 4, 6]]
```

```
[[1, 3, 5], [2, 4, 6]]
```

2. Sequence Slicing

[14] ✓ 0s

```
def copy(seq):
    return seq[:]
```

[18] ✓ 0s

```
def all_but_last(seq):
    return seq[:-1]
```

[23] ✓ 0s

```
def every_other(seq):
    return seq[::2]
```

3. Combinatorial Algorithms

[31] ✓ 0s

```
def prefixes(seq):
    for i in range(len(seq) + 1):
      yield seq[:i]
```

[32] ✓ 0s

```
def suffixes(seq):
    for i in range(len(seq) + 1):
      yield seq[i:]
```

[37] ✓ 0s

```
def slices(seq):
    for i in range(len(seq)):
      for j in range(i + 1, len(seq) + 1):
        yield seq[i:j]
```

4. Text Processing

[40] ✓ 0s

```
def normalize(text):
    return " ".join(text.lower().split())
```

[43] ✓ 0s

```
def no_vowels(text):
    return "".join([char for char in text if char.lower() not in "aoeiu"])
```

[49] ✓ 0s

```python
def digits_to_words(text):
    digit_dict = {
        '0': 'zero', '1': 'one', '2': 'two', '3': 'three', '4': 'four',
        '5': 'five', '6': 'six', '7': 'seven', '8': 'eight', '9': 'nine'
    }
    return " ".join([digit_dict[char] for char in text if char.isdigit()])
```

[52] ✓ 0s

```python
def to_mixed_case(name):
    words = [word for word in name.strip('_').split('_') if word]

    if not words:
        return ""

    first_word = words[0].lower()
    capitalized_words = [word.capitalize() for word in words[1:]]

    return first_word + "".join(capitalized_words)
```

5. DataFrames

[59] ✓ 0s

```python
# hint: look at the pandas method read_csv
def read_file(file):
  return pd.read_csv(file)
```

[61] ✓ 0s

```python
df = read_file('batch.csv')
```

[65] ✓ 0s

```python
def get_workerId_by_row(df, row):
    return df.iloc[row]['WorkerId']
```

[66] ✓ 0s

```python
# EXAMPLE
get_workerId_by_row(df, 3)
# 'A98E8M4QLI9RS'
```

```
'A98E8M4QLI9RS'
```

[67] ✓ 0s

```
# TODO
# convert the given date into a timestamp (in seconds)
def format_date(date):
    return pd.to_datetime(date).timestamp() / 60
```

[68] ✓ 0s

```
# EXAMPLE
# format_date('2020-02-06 23:25:41') -> 26350525.683333334
format_date('2020-02-06 23:25:41')
```

```
26350525.683333334
```

[74] ✓ 0s

```
# hint: look at the dataframe groupby function
def get_work_time(df):
    df['Duration'] = df['SubmitTime'] - df['AcceptTime']
    return df.groupby('WorkerId')['Duration'].sum()
```

[75] ✓ 0s

```
# EXAMPLE
# Should be a series like
# Time
# Time
get_work_time(df)
```

WorkerId	Duration
A1CY7IOJ9YH136	14.016667
A2C84POENS2UNY	255.416667
A98E8M4QLI9RS	35.400000
AMPMTF5IAAMK8	63.316667
ANO5I7E78QMEN	103.366667
AVBRJBJONL47I	24.966667
AY7WPVKHVNBLG	237.166667

[76] ✓ 0s

```
def calculated_work_time(df):
    grouped = df.groupby('WorkerId')
    return grouped['SubmitTime'].max() - grouped['AcceptTime'].min()
```

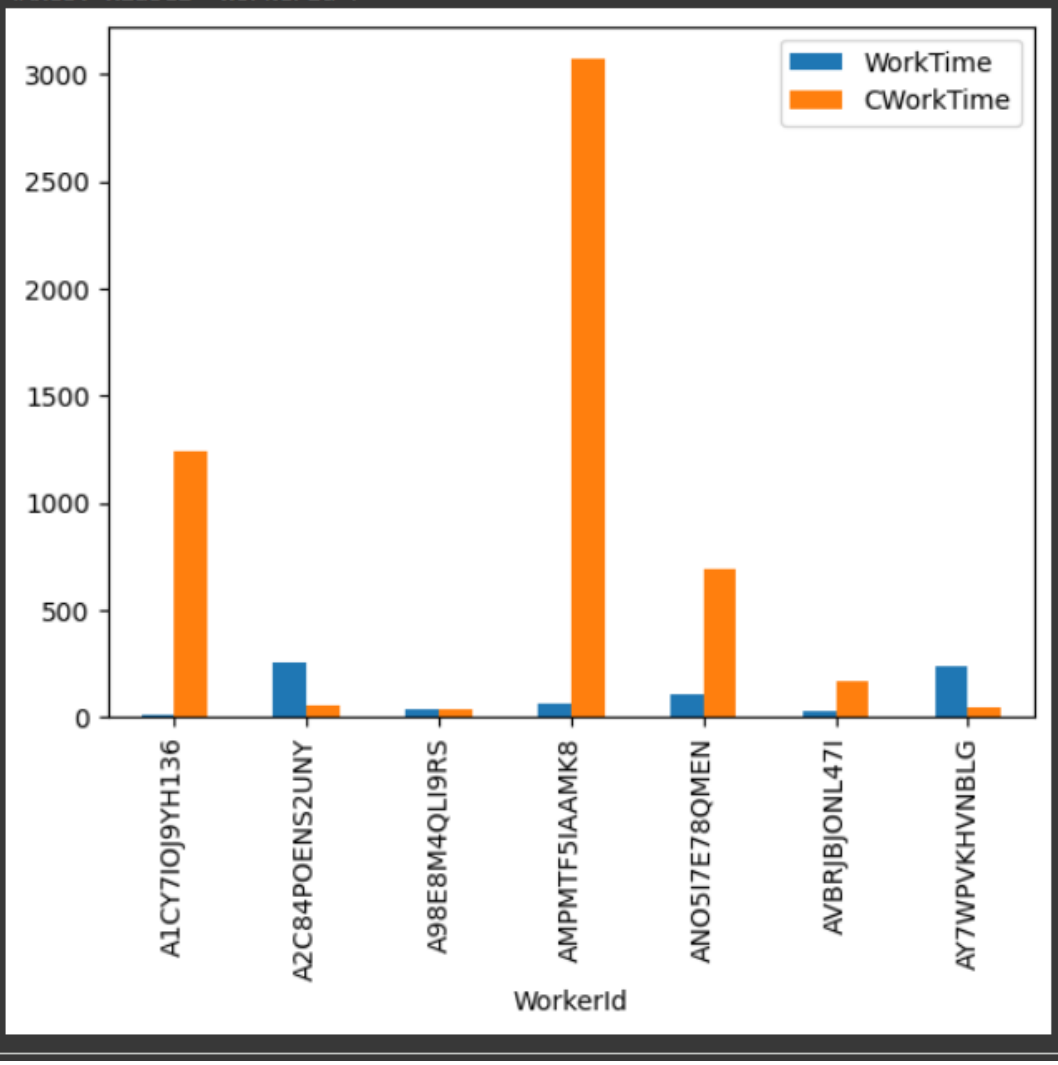
[77] ✓ 0s

```
# EXAMPLE
# Series should look the same as above but with different values for some workers
calculated_work_time(df)
```

WorkerId	0
A1CY7IOJ9YH136	1236.800000
A2C84POENS2UNY	58.550000
A98E8M4QLI9RS	35.500000
AMPMTF5IAAMK8	3068.883333
ANO5I7E78QMEN	692.616667
AVBRJBJONL47I	171.166667
AY7WPVKHVNBLG	45.866667

```
plot_df.plot(kind='bar')
```

<Axes: xlabel='WorkerId'>



Task #4 (25 min spent)