**Vidzeme University of Applied Sciences**

**Faculty of Engineering**

Introduction to Python and Data Exploration

**group 06**

**practical work #2**

Valmiera, 2024

**Table of Contents**

[1.0 Content 4](#_Toc161075183)

[2.0 Tasks and their solutions 5](#_Toc161075184)

[2.1 Task 1 5](#_Toc161075185)

[2.2 Task 2 6](#_Toc161075186)

[3.0 Chapter 2 7](#_Toc161075187)

[4.0 References 10](#_Toc161075188)

[5.0 Appendix (Code) 11](#_Toc161075189)

[5.1 Task\_2.1\_Gr.06.py 11](#_Toc161075190)

[5.2 Task\_2.2\_Gr.06.py 11](#_Toc161075191)

|  |  |  |  |
| --- | --- | --- | --- |
| Document versions | | | |
| Version | Status / Changes | Date | Author |
| 0.1 | First version implementation | 11. 03.2024 | Kristers Kalniņš |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Contacts and responsible (-s) | | | |
| Name Surname | Department | Position | Contact information (e-mail) |
| Emīls Konrāds | Group #06 | Coordinator | emils.konrads@va.lv |
| Gatis Jurisons | Group #06 | Member | gatis.jurisons@va.lv |
| Kristers Kalniņš | Group #06 | Member | kristers.kalnins@va.lv |
|  |  |  |  |
|  |  |  |  |

# Content

**Goal:** The team (further referenced as team 06) must complete the given task by the client (university professor) described in task 1 of unit 2.

**Content**: two charts, 667 words, six images, three appendixes.

# Tasks and their solutions

Team 06 had organized a meeting after being given the task. On the next weeks Monday the team had started organizing a workflow and work management chain. The team 06 decided to split the workload in 2 different categories:

* Documentation
* Coding

For the work flow we used the previously created Github repository. By the end of each week each team participant must send in a flow chain of how they see the app working, each demonstrating their plan of development.

## Task 1

The first task is an updated version of the previous practical works final code, in this code the team had added a function called compute\_wage which takes two parameters: hours and rate. (Osis, Moodle.va.lv, 2024)

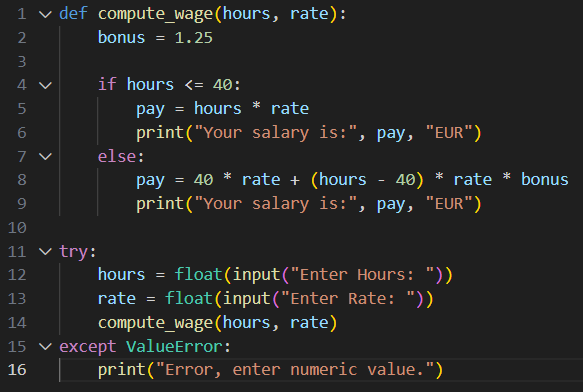


Image: 1.1. Code from (Task\_2.1\_Gr.06.py)

## Task 2

The second task was to try and recreate the number eight from the given example using the symbol asterisks: ‘\*’. (Osis, Moodle.va.lv, 2024)

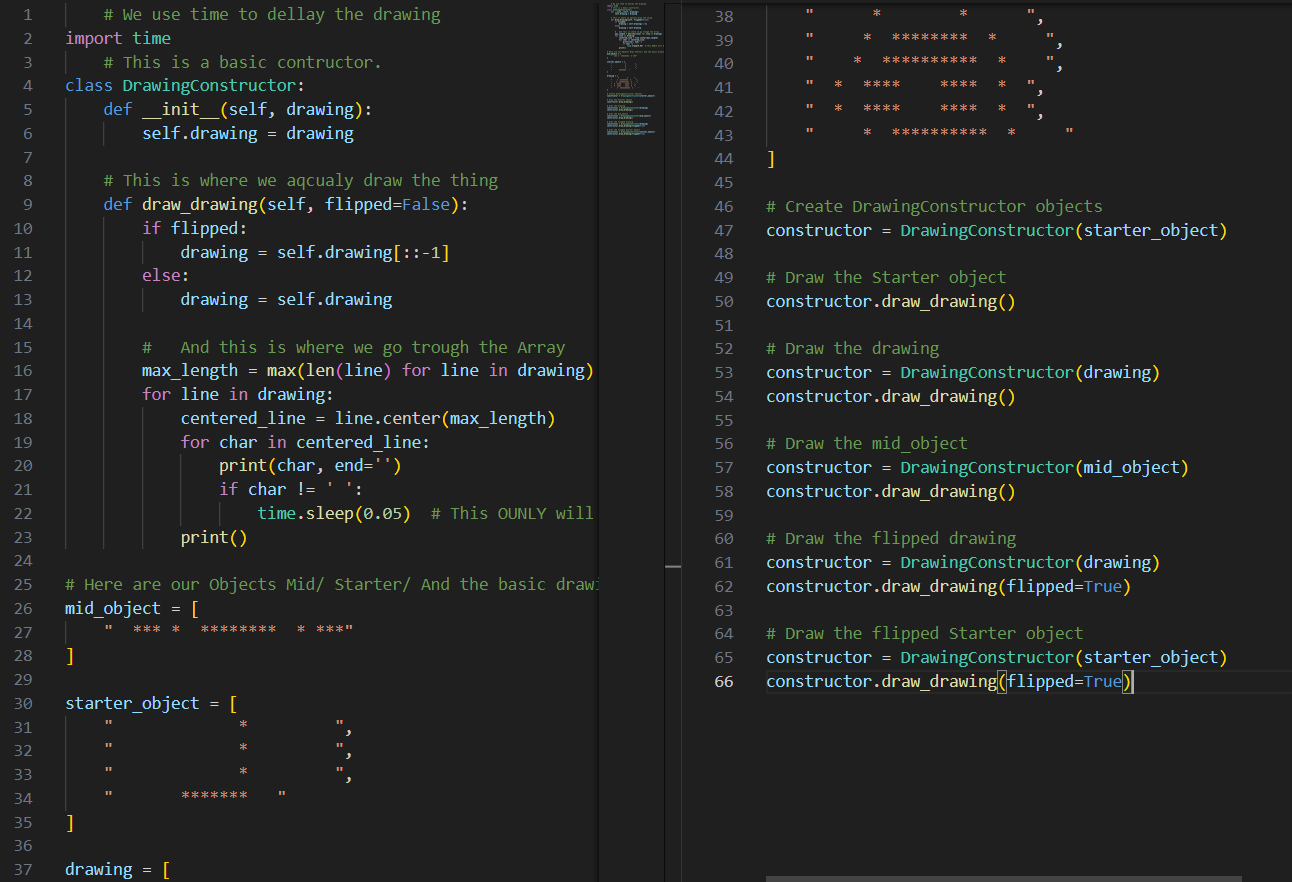


Image: 1.2. Code from (Task\_2.2\_Gr.06.py)

# Chapter 2

As described in the starting chapter before all the participants of group 06 had handed in Flowcharts of how each one of the participants sees the app working. This was all managed using discord where each participant sent their work in for review.

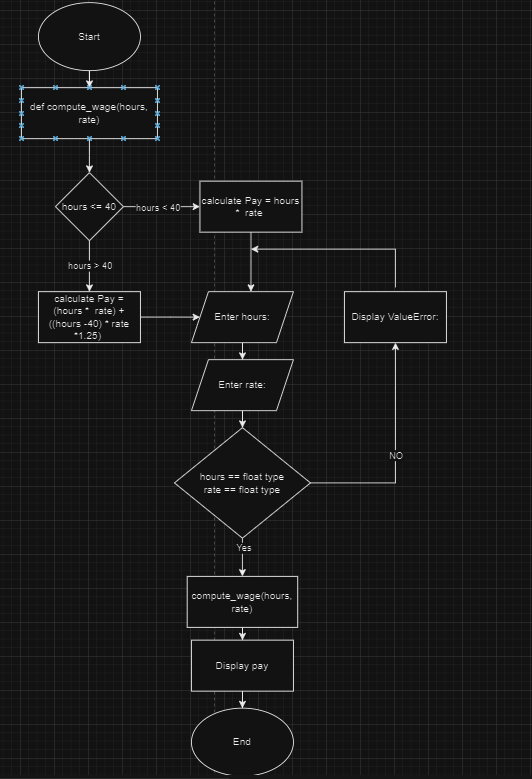


Image: 2.1. Gatis Jurisons Flowchart of Task 1.

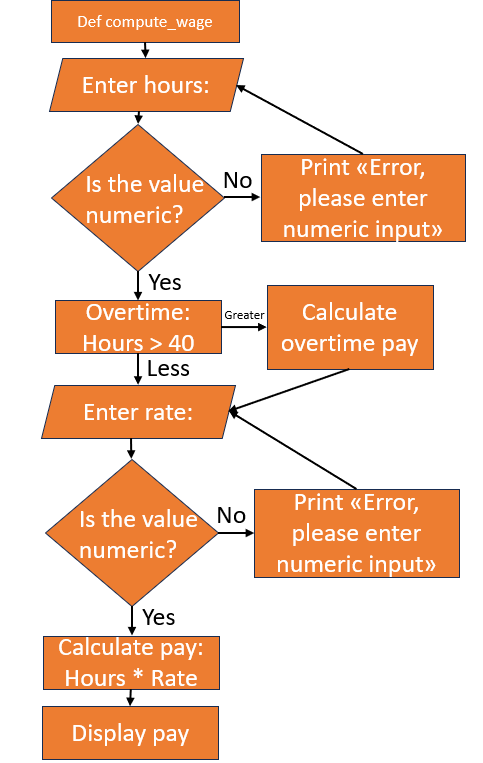


Image: 2.2. Kristera Kalniņa Flowchart of Task 1.

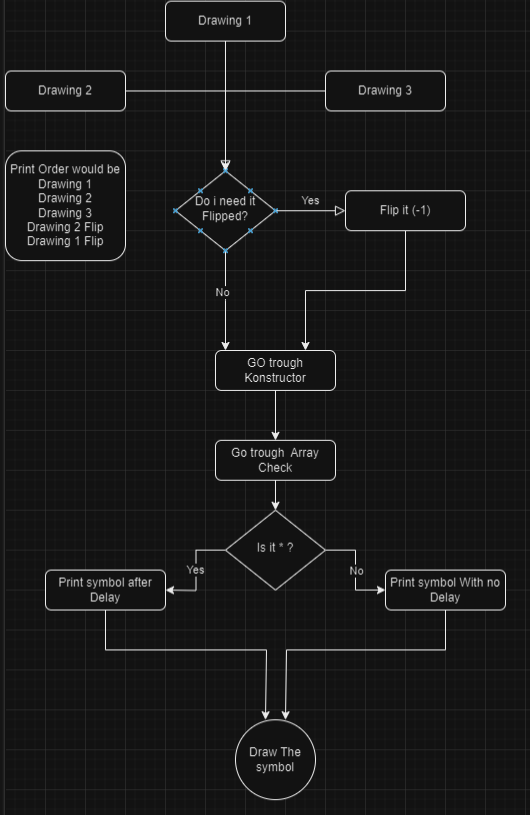


Image: 2.3. Emīla Konrāda Flowchart

# References

Osis, K. (2024, Mar). *Moodle.va.lv.* Retrieved from

https://moodle.va.lv/pluginfile.php/31318/mod\_resource/content/8/2024\_practical\_work\_2.pdf

Osis, K. (2024, Mar). *Moodle.va.lv.* Retrieved from Template of Document: https://moodle.va.lv/mod/resource/view.php?id=11925

# Appendix (Code)

## Task\_2.1\_Gr.06.py

def compute\_wage(hours, rate):

    bonus = 1.25

    if hours <= 40:

        pay = hours \* rate

        print("Your salary is:", pay, "EUR")

    else:

        pay = 40 \* rate + (hours - 40) \* rate \* bonus

        print("Your salary is:", pay, "EUR")

try:

    hours = float(input("Enter Hours: "))

    rate = float(input("Enter Rate: "))

    compute\_wage(hours, rate)

except ValueError:

    print("Error, enter numeric value.")

## Task\_2.2\_Gr.06.py

 # We use time to dellay the drawing

import time

    # This is a basic contructor.

class DrawingConstructor:

    def \_\_init\_\_(self, drawing):

        self.drawing = drawing

    # This is where we aqcualy draw the thing

    def draw\_drawing(self, flipped=False):

        if flipped:

            drawing = self.drawing[::-1]

        else:

            drawing = self.drawing

        #   And this is where we go trough the Array

        max\_length = max(len(line) for line in drawing)

        for line in drawing:

            centered\_line = line.center(max\_length)

            for char in centered\_line:

                print(char, end='')

                if char != ' ':

                    time.sleep(0.05)  # This OUNLY will dellay if its \*

            print()

# Here are our Objects Mid/ Starter/ And the basic drawing

mid\_object = [

    "  \*\*\* \*  \*\*\*\*\*\*\*\*  \* \*\*\*"

]

starter\_object = [

    "             \*         ",

    "             \*         ",

    "             \*         ",

    "       \*\*\*\*\*\*\*   "

]

drawing = [

    "      \*        \*      ",

    "     \*  \*\*\*\*\*\*\*\*  \*     ",

    "    \*  \*\*\*\*\*\*\*\*\*\*  \*    ",

    "  \*  \*\*\*\*    \*\*\*\*  \*  ",

    "  \*  \*\*\*\*    \*\*\*\*  \*  ",

    "     \*  \*\*\*\*\*\*\*\*\*\*  \*     "

]

# Create DrawingConstructor objects

constructor = DrawingConstructor(starter\_object)

# Draw the Starter object

constructor.draw\_drawing()

# Draw the drawing

constructor = DrawingConstructor(drawing)

constructor.draw\_drawing()

# Draw the mid\_object

constructor = DrawingConstructor(mid\_object)

constructor.draw\_drawing()

# Draw the flipped drawing

constructor = DrawingConstructor(drawing)

constructor.draw\_drawing(flipped=True)

# Draw the flipped Starter object

constructor = DrawingConstructor(starter\_object)

constructor.draw\_drawing(flipped=True)