LOGGING

לוגינג הוא כלי שמאפשר לעקוב אחרי השתלשות האירועים כשהתוכנה רצה, המפתח מוסיף קריאת לוג לקוד כדי לתת אינדיקציה שמאורע כלשהו התרחש.
לפייתון יש ספרייה בשם logging המגיעה עם השפה, ומספקת סט של פונקציות נוחות לשימוש.
הספרייה מאפשרת לנו להוסיף הודעות מצב (status massages) לקובץ או לאובייקט output אחר.
ההודעות האלה מכילות מידע כמו איזה חלק בקוד רץ ואילו בעיות עלולות להעלות.
לכל הודעת לוג יש רמה, חמש הרמות הבנויות מראש הן : Debug, Info , Warning, Error ו- Critical.
ניתן גם להגדיר רמות מתואמות אישית, אך רמות אלו אמורות להיות מספיקות.
כדי להוסיף אובייקט לוג חדש צריך דבר ראשון לייבא את הספרייה logging של פייתון. הספרייה היא בסיסית בשפה ואין צורך להוריד איזשהי ספרייה מתואמת אישית.
כדי ליצור אובייקט לוג חדש נצטרך דבר ראשון להגדיר היכן נפלוט את התוכן של הלוג עם הפונקציה basicConfig() אח"כ נשמור את האובייקט שהתקבל במשתנה עם הפונקציה getLogger :

```
import logging

#create and configure logger
logging.basicConfig(filename= 'my_logging.log')
logger = logging.getLogger()
```

לכל רמה של הודעה יש פונקצייה ייעודית, למשל להוספת הודעת info נשתמש בפונקציה עם אותו שם, וכנ"ל לשאר הרמות. לכל רמה יש גם סוג של דירוג, הדירוג נועד להגדיר אילו סוגי הערות לא נראה כאשר אנחנו בקובץ לוג.
כל רמה היא גדולה מהבאה לפניה ב-10 , כלומר הדרגות מסודרות: NOTSET=0 , DEBUG=10, INFO= 20, WARRNING=30 , ERROR= 40, ו- CRITICAL = 50. כברירת מחדל הלוגגר מוגדר להיות Warrning לכן כל ההודעות שהדירוג שלהן נמוך מ-30 לא יפלוט לקובץ.
אם עכשיו ננסה לפלוט לקובץ הודעת info או debug לא נוכל לראות אותה במסמך, לכן נצטרך להוסיף פרמטר level לפונקציה basicConfig() ולשים לו ערך קטן יותר מ-warning למשל debug:

```
import logging

#create and configure logger
logging.basicConfig(filename= 'my_logging.log' ,level= logging.DEBUG)
logger = logging.getLogger()

# Test the logger
logger.info('This is a message')
```

ואם נסתכל בקובץ לוג נראה שההודעה נשמרה כמו שצריך:

```
INFO:root:This is a message
```

ההודעה מכילה שלושה חלקים- את סוג ההודעה, במקרה זה הודעת info, את שם הלוגגר, במקרה זה root , ואת תוכן ההודעה.
אנחנו יכולים גם לשנות את מבנה ההודעה ע"י הוספה של ארגומנט format לפונקציה basicConfig() .
פייתון מספקת לנו כמה משתנים שנוכל להוסיף לפורמט של הודעת לוג למשל זמן עם המשתנה asctime או את מספר השורה lineno , אפשר למצוא את הרימה המלאה של השמות כאן.
הפורמט של ההודעה צריך להיות סוג של מחרוזת עם המשתנים בפנים:

```
import logging

LOG_FORMAT = "%(levelname)s, time: %(asctime)s , line: %(lineno)d- %(message)s "
#create and configure logger
logging.basicConfig(filename= 'my_logging.log' ,level= logging.DEBUG, format= LOG_FORMAT)
logger = logging.getLogger()

# Test the logger
logger.info('This is a message')
```

ואז הטקסט שנקבל הוא:

```
INFO:root:This is a message
INFO, time: 2021-01-31 19:40:44,441 , line: 9- This is a message
```

שימו לב שהטקסט הקודם לא נדרס, זה כי הקבוץ לוג הוא בפורמט 'r+' אם נרצה נוכל לשנות אותו לפורמט אחר ע"י הוספת ארגומנט 'filemode = w'.
ועכשיו נראה דוגמא של שימוש בלוג עם בקובץ.
נבנה פונקציה שמחשבת את נוסחת השורשים בהתאם לקלט של a,b,c:

```
def quadratic_formula(a:float, b:float ,c:float) -> float:
    import math
    """ Returns the solutions to the equation ax^2 + bx + c = 0  """
    logger.info(f'quadratic_formula({a},{b},{c})')
    logger.debug('Compute the discriminant')
    discri = b**2 - 4*a*c
    logger.debug('Compute the two roots')
    root_a = (-b + math.sqrt(discri))/(2*a)
    root_b = (-b - math.sqrt(discri))/(2*a)
    logger.debug('Return the two roots')
    return root_a , root_b
```

הפונקציה אמורה לכתוב לקובץ לוג לפני כל פעולה שמתבצעת בפונקציה.
ובאמת אם נריץ אותה על הקלט 1,0,-4 אנחנו אמורים לקבל את התשובה 2,-2 וזה אמור להופיע גם בקובץ:

```
print(quadratic_formula(1,0,-4))
with open('my_logging.log') as file:
    print(file.read())
```

```
(2.0, -2.0)
INFO, time: 2021-01-31 20:25:54,978 , line: 12- This is a message
INFO, time: 2021-01-31 20:25:59,159 , line: 4- quadratic_formula(1,0,-4)
DEBUG, time: 2021-01-31 20:25:59,159 , line: 5- Compute the discriminant
DEBUG, time: 2021-01-31 20:25:59,159 , line: 7- Compute the two roots
DEBUG, time: 2021-01-31 20:25:59,159 , line: 10- Return the two roots
```

אבל אם נוסיף ניתן לפונקציה קלט שהיא לא אמורה לזהות למשל 1,0, 1 נראה שלא נקבל יודפס ערך החזרה מהפונקציה וכך נדע היכן בדיוק קרתה השגיאה:

```
quadratic_formula(1,0,1)
```

ואז בקובץ לוג נראה:

```
INFO, time: 2021-01-31 20:31:03,650 , line: 4- quadratic_formula(1,0,1)
DEBUG, time: 2021-01-31 20:31:03,653 , line: 5- Compute the discriminant
DEBUG, time: 2021-01-31 20:31:03,653 , line: 7- Compute the two roots
```

במקרה זה הקוד קטן, אבל תחשבו על קוד גדול עם מלא שורות, לאתר את מקום התקלה יהיה קשה מאוד ללא קובץ לוג, בעזרתו נוכל לגעת ישירות באיזו פונקציה נתקענו, באיזו שורה נתקענו, מה היה הקלט, מה הפלט וכך לדעת איך לתקן.