Diario di lavoro

|  |  |
| --- | --- |
| Luogo | Canobbio |
| Data | 25.02.2021 |

|  |
| --- |
| Lavori svolti |
| 08:20 – 09:50  Dopo aver fatto il punto della situazione abbiamo fatto un po’ di ricerca su Python, sulla formattazione delle stringhe, la libreria di logging e pep8; abbiamo sistemato la struttura dei file nella cartella src per poi continuare a lavorare sul logging e sull’ocr.  Mentre una cercava di leggere i files con l’ocr e stamparne l’output in un file txt, l’altra lavorava sul file di script principale sistemandone la struttura in metodi.  10:05 – 11:35  Dopo aver strutturato il file principale (ocr.py) con il metodo *main()*   * Creato file script principale, * sistemata struttura files  (il file principale contiene il main dal quale richiama tuttte le funzioni necessarie per l’ocr; le statistiche sono in stats.py che viene importato come modulo dal main; le azioni principali di reader.py da decidere se usare come modulo o se metterle direttamente in ocr.py), * continuato logging (provato gli handlers, letto documentazione)   Provato a usare logging.handlers, preso file A\_log.py da sender di M143, modificato e testato per capire come ottimizzare il logging  12:30 – 14:00  Sistemato codice ocr e aggiunto controlli in modo da gestire il caso in cui ci siano più file di output con lo stesso nome.  Dopo vari tentativi con il logging e un aiuto finale dal sore ho terminato il modulo di logging.  In pratica: nel *log\_handler* prendo l'oggetto *logger* e lo modifico formattandolo come preferisco;  il metodo *get\_configure\_logger()* mi ritorna però un nuovo oggetto *logger*, quindi se volessi usare il mio *logger* modificato sui vari scripts e moduli dovrei richiamare ogni volta la funzione per sovrascrivere il basic logger (il root).  Per risolvere vado a modificare direttamente il *logger root*:  logger = logging.getLogger('') # salvo la radice del logger per poi modificarla  In questo modo tutte le funzioni dei moduli importati nel file principale (nel quale ho chiamato *get\_configure\_logger()*) potranno usare il logger modificato ereditandolo dal file principale.  Dopo ho sistemato gli argomenti aggiungendo la versione ristretta (-lang/-l) e descrivendo meglio il loro utilizzo nell’help.  14:15 – 15:45  Lavorato sulla numerazione dei prefissi (scan\_1, scan\_2, ecc), aggiunto param –examples per mostrare esempi di utilizzo del comando.  In fine ho sistemato il diario. |

|  |
| --- |
| Problemi riscontrati e soluzioni adottate |
| Mentre cercavo di scrivere il testo letto dall’immagine in un file di testo ho riscontrato problemi di permessi: *PremissionError*  Non mi lascia accedere alla cartella .\scans.  La cartella scans stata creata manualmente, per risolvere ed evitare ulteriori problemi futuri anche all’utente nel codice ho aggiunto una parte per creare la cartella nella quale dentro verrà creato il file .txt dove verrà stampato il contenuto dell'immagine.  Ho preso in pezzo del codice da: <https://stackoverflow.com/questions/35807605/create-a-file-if-it-doesnt-exist>  Per creare la cartella: <https://linuxize.com/post/python-get-change-current-working-directory/#:~:text=To%20find%20the%20current%20working,chdir(path)%20>. |

|  |
| --- |
| Punto della situazione rispetto alla pianificazione |
| Stiamo seguendo il Gantt senza problemi anche se stiamo alterando le dipendenze delle attività svolgendole in ordine differente rispetto al Gantt. |

|  |
| --- |
| Programma di massima per la prossima giornata di lavoro |
| * sistemare problemi accumulati * iniziare test |