| Project Name | Project Description |
|---|---|
| Netflix Data Analysis | 4 python files: **create_.py** [ no input, output: filled csv files] / **dataPrep_.py** [input: dfs or/and csv-file path, output: modified DataFrames] / **dataPlot_.py** [input: dfs and constraints, output: nothing returned, plot is shown] / **play_.py** [input throught user prompts (numbers), output: plots with the wanted data]. **play_.py** is the **main (executive) part** of the program, the **other scripts** do the **background work**. |
| Mental Rotation Experiment | 2 python files:<br>**mental_rotation_task.py** [Expyriment script replicating mental rotation experiment from Ganis et al (2015)]<br>**combine_results_to_csv.py** [Input: xpd files from experimetns, Output: one clean csv file]<br><br>optional part:<br>1 jupyter notebook:<br>**Processing_and_plotting.ipynb** [Input: results.csv; for visualization and interpretation of gathered data] |
| SymPy Bot | Inputs: writing the telegram bot: "derivate 'sin(exp(x^2))'" Output: bot awnsers: "2*x*exp(x^2)*cos(exp(x^2))" Methods: pyTelegrambotapi / Sympy / matplotlib 6 python files: main.py - the actual bot boolean_functions - boolean functions for the bot handlers parsing_functions - parsing inputs saving_data - saving data with pickle plot.py - matplotlib plots math_part.py - wrapper functions around sympy functions |
| WHR Data Analysis | |
| Stock Data Webscraping | |
| Lieferando Data Analysis | 5 python files:<br>lets_scrape.py [input: User choices for graphs and processing data; output: pdf files with plots] scraper.py[Input: City address; methods: scraper to get restaurants; output: formatted data] visualization.py[input: formatted data; methods: plotting methods; output: formed plots] data_helper.py[input: data to be trimmed or refined; methods: helper methods for any kind of data manipulation; output: refined data] ui_helper.py[input: user choices; methods: controlling user interactions; output: UI/terminal printing] |

| | |
|---|---|
| Stroop Task Experiment | 4 python files: **make_design.py** [inputs: empty experiment, random int to determine test group, output: constructed experiment] **combine_results.py** [input: all xpd files with participants' results, output: single csv-file with all results combined] **plot_data.py** [input: combined csv-file with all results, displays results and some information about the collected data] **stroop_task.py** [contains the main block of the experiment, utilizes the other parts to construct and conduct the experiment] |
| PyGame Agent | **genetic_algorithm.py: main GA loop for the asteroids AI / Environment.py: environment for the asteroids agent / Agent.py: PyTorch Brain for the player / Player.py: the actual movement control of the spaceship ("singleplayer") / Enemy.py: Asteroids class (enemies) / Bullet.py: Bullet class such that the player can shoot / helper.py: contains helper and global variables** |
| DecisionTree | 6 python files/ 3 for classes to build the tree/ prepare_data.py: prepares/changes the data before the training/testing / split_data.py: calculates the spliting step of ID3 / main.py: mainpart,  does the comparing and visualizing |
| Flanker Task Experiment | 4 python files: **menu.py:** overlay for combining all functionalities / **FlankerExperiment.py**: running the experiment / **DataWrangling.py:** cleaning and plotting the data / **PersonalSummary.py:** cleaning and plotting own results, as well as contributing data to collection, **DataSummary.ipynb:** summarizing collection of cleaning & all plots + a small linear regression / **data/data.csv:** entire data collection |

| | |
|---|---|
| Kaleidoscope | 2 python files:<br><br>Snowflake.py: generates a Koch snowflake from Koch curve line segments<br>(no input, no output)<br><br>kaleidoscope.py:<br>Input: the user can decide: how many snowflakes will be generated, the colours of the kaleidoscope, if it rotates or not, and how fast the kaleidoscope changes.<br><br>Output: guidance and input request to the user to generate the kaleidoscope. The animation of the kaleidoscope will be saved as a gif file.<br><br>The program sets coordinates for each snowflake (if the kaleidoscope should rotate, every coordinate rotates by 12°)<br><br>Koch snowflakes will be generated and the colours for the lines will be set to generate the animation, which will be saved as a gif. |
| Craik & Lockhardt Experiment | *Experiment: Shallow VS Deep Processing and its Impact on Memory Retention*<br><br>***Inputs:*** *config.json (allows easy access to important settings, managed by setting_constants.py), 2 csv files with stimuli data*<br>***Main Python Scripts:*** *exec_experiment.py (run to create and conduct experiment), helper_functions.py (outsourced functions to create blocks, get and (pre)process stimuli data) [the individual functions are listed with inputs, outputs and “behavior” in the README.md file]*<br>***Outputs:*** *Expyriment logs and experiment design files* |
| Authorship Authentication | |

| | |
|---|---|
| Stroop Task Experiment | Implementation of the numerical variation of the stroop effect. Takes keyboard button input from the subject and puts out a data (.xpd file) and event (.xpe file) |
| WHR Data Analysis | **data_prep.py: clean and prepare the data / viz_outliers.py: find and visualize outliers / data_viz.py: plot the data / final-project-notebook.ipynb: main part, used to call functions from the other files and show the plots** |
| NME Exploration | |
| Clothing Store Scraper | Webscraper for Vinted.com |
| Data Analysis | look at Github rep |
| Mensa Scraping | 4 python files main.py: processes information and draws plots, page.py: prepare page to scrape, scrape.py: scrapes information |
| Vergabeportal Scraping | |
| Medical Data Analysis | |
| Authorship authentication | main is execution.py with following operations: preparation of books for ananlysis, collection of features of each sequence, evaluation of system, prediction of author from unknown book, author guessing game |
| Corsi Experiment | 2 python files: **corsi_block_tapping.py** (Corsi block-tapping test; experiment design and execution), **concatenate_data.py** (convert data log files to .csv file to enable further analysis) |
| Source Code Analysis | 3 packages, each with one file:<br>LoadAndCalculate loads the git log data and calculates all information about code hotspots,<br>Analysis runs these calculations and prints/saves the output,<br>Visualization creates plots out of the calculation.<br>The last file, Main in the root directory, is the only thing the user needs to run, because it starts anything else.<br>Output can be stored in output.txt, or printed to the command line, or shown as 3 plots. |

| | |
|---|---|
| StockPerformanceIndicators | 7 python files & 7 csv files in a cache-dir (containing information of ca. 35.000 stocks). The user can visualize a certain PerformanceIndicator of a certain stock in comparison with the (filtered) database. The only file the user should execute is the main.py and (maybe) the csv_downloader.py). |