Experiment overview:

The project aims at reproducing the visual search task (VST) paradigm developed in 1980 by Treisman and Gelade. The experiment was developed to test behaviourally how the brain manages to perceive scenes as a whole. Does the brain process single features of individual objects, and then puts them together in a coherent whole, or does it fetch individual features only after having built the scene as a whole? The VST paradigm aims at testing both approaches to how brain processes perceptual information.

The task is simple: participants have to search for a target stimulus among a plethora of distractor objects. Both shape and color of target and distractors can vary according to two main search strategies:

1. The 'feature search' strategy is attested by the famous 'pop-up effect', which is a measurable behavioural phenomenon related to human's ability to promptly distinguish a single object among many others, based on one distinctive feature not shared with other stimuli.
2. The 'conjunction search' strategy on the other hand, which is believed to be triggered when target and distractors share features in some combinations (e.g. some distractors might share only the color, some other might share only the shape with the target), is a cognitively demanding process modulated by the quantity of stimuli within the scene.

Accordingly, the VST task was developed to test how/whether selective attention is recruited in more effortful cognitive processes.

For this reason, the main parameter that are varied across trials are:

1. Size of the scene: i.e. the number of distractors (plus the target) to be displayed.
2. Type of search: "feature based" or "conjunction based".
3. Targets are present or absent.

Behavioural measures accounted for are accuracy and reaction times.

Description of the procedure:

With the default number of distractors indicated in the "experiment set up wizard" (4, 8, 16, 32), participants can work through 80 trials, i.e. 5 visual scenes * 16 possible combinations of parameters. This means 5 blocks of 16 trials per participant, varying the number of distractors. Each block would display every possible combination of features and target's presence.

As this is an experiment generator, the variables are all custom for the experimenter, and thus it's not necessary to run so many trials. See below for more in-depth explanation.

As a bonus variable to test, it is also possible to display moving distractors and see how this will affect the participants' proficiency in the search task.

Experimenter's manual:

The present project is a VST task generator, which utilises PyQt5 and Psychopy modules. Thus, it is necessary to import the two libraries to correctly use the program. Psychopy module is supported only for Python 3.8 and it is HIGHLY RECCOMENDED to use only this version. To correctly download the library, I suggest to type `python3.8 -m pip install psychopy` in the terminal.

To start generating trials, run the "main.py" file opening the Experiment set up Wizard (the "experiment_wizard.py" file). In this interface, the experimenter can set up any parameter among:

1. Number of distractors to be displayed.
2. Whether to display moving objects.
3. Target's shape and color.
4. Distractors' shape and color.
5. Number of trials.
6. Number of trials in which the target will be present ("Target presence").

The experimenter can also specify a different combination of keyboard keys (the default is "y,n" for "yes" and "no" respectively; max. two, to be separated by a comma) that will be used by the participant during the experiment.

After the experimenter wizard, a dictionary with conditions labels (e.g. "n_distractors" for number of distractors, etc.) as keys, and all the possible values they can assume within trials (e.g. "4,8,16,32" means that scenes will be built according to one randomly picked value among those), will be generated according to the experimenter's input. This will be used to then generate a unique series of single values per condition per trial, according to the number of trials specified.

Then, a demographic window ("demographic_win.py") will be shown, which gathers demographic information about the participant in the form of a dictionary with labels as keys, and the participant input as values. A window with a template for the consent form ("consent_form.py") is also available, should the experimenter want to collect the participant's agreement electronically.

Only after the consent is being given by the participant, an input file ("conditions_input.csv") will be created with all the conditions per trial specified. This specifies the flow of the experiment. Notice that the input file can be modified adding/removing rows or changing their order, and the experiment will still run with the new set up.

Moreover, an output file will be created with the demographics labels and participant's data written as first two rows. The output file is being stored in a data_output subfolder of the current session's directory. If the data_output folder doesn't exist already, it will be generated automatically. Each output file is unique and contain information about the single experiment session. It will contain:

1. First row: demographics data labels (e.g. Age:, Gender:, etc.)
2. Second row: demographics input of the participant
3. Third row: parameters' labels (e.g.: n_distractors, target_shape; etc.)

From the forth row on, each row corresponds to one trial, and contains the unique values associated to each condition which were used to build the single trial. Keyboard pressed and reaction times are also attached at the end of each row.

After the output file is being created, the experiment can start. The experiment flow ("experiment_flow.py") utilises Psychopy's classes to create the window, the stimuli, and to retrieve information about the participants’ interactions (reaction times and keys pressed). Only after the block of trials finishes, the output file will be written as indicated above.

Program highlights:

The program displays a combination of Object Oriented coding in PyQt and Psychopy modules. The program has been written with flexibility in mind, and any interface displayed should be fairly easy to be modified, for example adding entries in the demographics window, or adding variables in the experimenter wizard. The experimenter can either interact with the interface, or directly altering the input .csv file (“conditions_input.csv” as default) in order to run the experiment. The experiment itself is fully customizable, allowing the experimenter to set up mixed design experiments. The choice of using a conditions input file was inspired by the famous online platform testable.org, which allows for rapid customization of experiments via an excel-like interface. It is not strictly necessary, but allows for a layer more of customization. Can be easily bypassed by substituting the name of the input file passed as argument in the experiment_flow() function with the dictionary created within the main.py.