Alice & Bob's Quantum Doubles:

Demo Files and Source Code

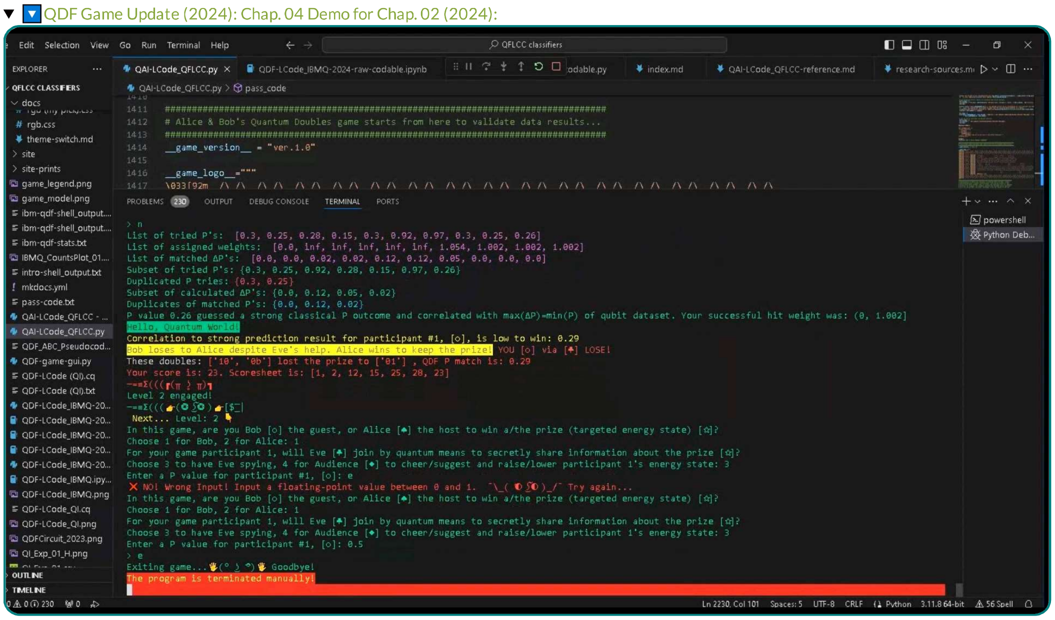
QDF Game Intro Demo: 880 MB mp4 file. ▽ | For Subtitles: SRT file. ▽

▼ ▼ i QDF game description (Start of Heading & Text) . . .

QDF game as part of QFLCC in QFLCA: program description from QAI-LCode_QFLCC.py

```
####################################################################################################
# These are the QDF game description, logo and copyright details.
####################################################################################################
__game_banner__ = """\n"\033[96m( ᵔ ͜ʖ ᵔ) vs ( ᵔ( ᵔ ͜ʖ ᵔ(▧ ͜ʖ▧) ᵔ ͜ʖ ᵔ) ᵔ) vs [$̲̅ ◉ ͜ʖ ◉)$̲̅] vs ¯\_( ◐ ͜ʖ◐ )_/¯ vs \
 ‒=≡Σ((( ⊃◕_ʖ◕)⊃ [♥]]] [♦]]] [♣]]] [♠]]] ‒=≡Σ(((✌(☆ ͜ʖ ☆)✌ ||||||||||||||||||||| \033[92m"""
__game_description__ = "Alice & Bob's Quantum Doubles... This QFLCC game version of the QAI-LCode is basic. \
More revisions to come, as the dataset grows on the number of trials on 1 or more QDF circuits for simulating. \
the QDF game. The QDF game simulates events of the dataset's quantum and classical parameters in observing a \
thermodynamic system = { QDF game environment and its participants Alice, Bob, Eve, Audience } based on Refs. [1-2,4] \
of the Data in Brief, Elsevier J. article."
```

▼ ▼ QDF Game Update (2024): Chap. 04 Demo for Chap. 02 (2024):



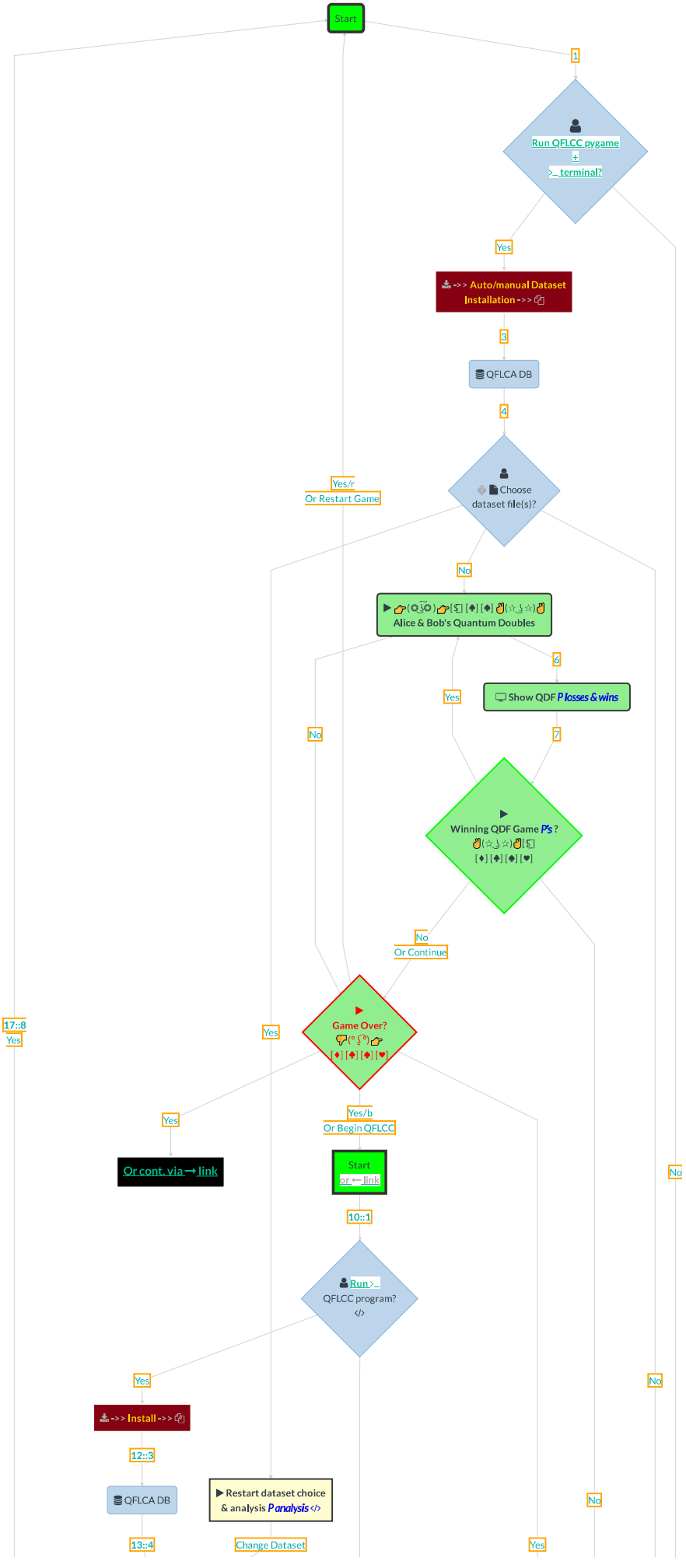
QDF Game 2024 Update: 635 MB mp4 file. ▽ | For Subtitles: SRT file. ▽

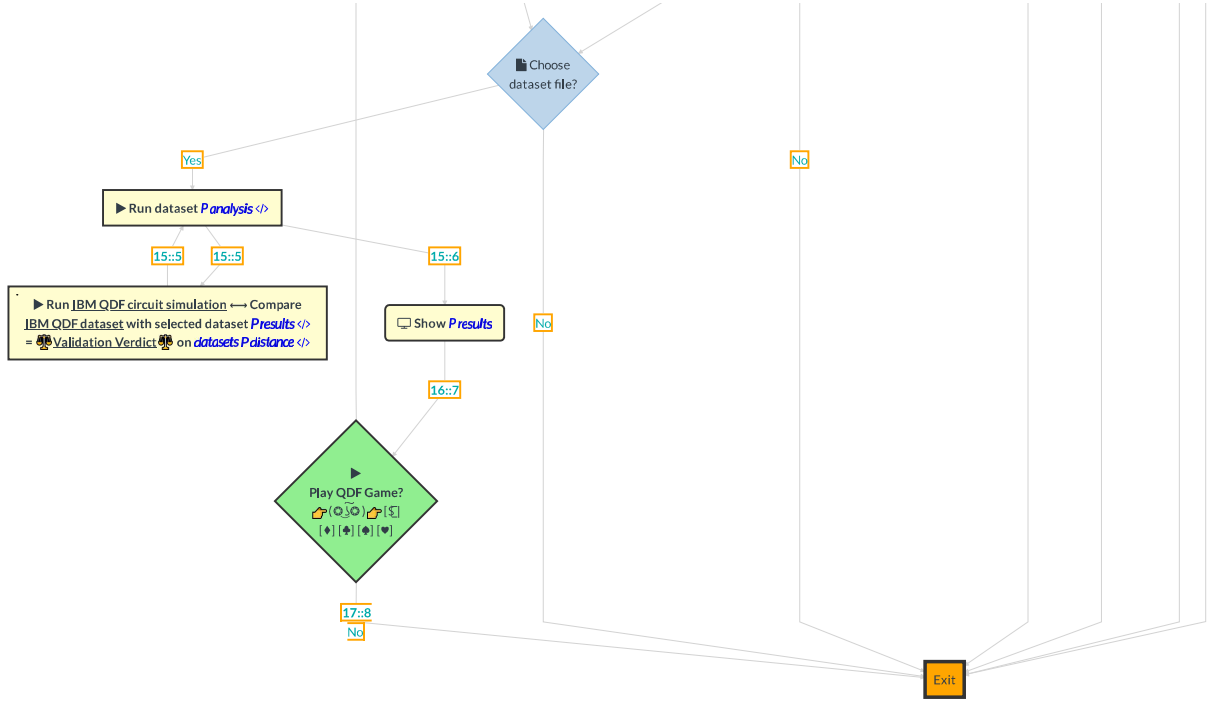
▼ Expand/Collapse All...

* Click Expand All to view all code blocks from QAI-LCode_QFLCC.py on this page.
* Click Collapse All to selectively view a code block from QAI-LCode_QFLCC.py on this page.

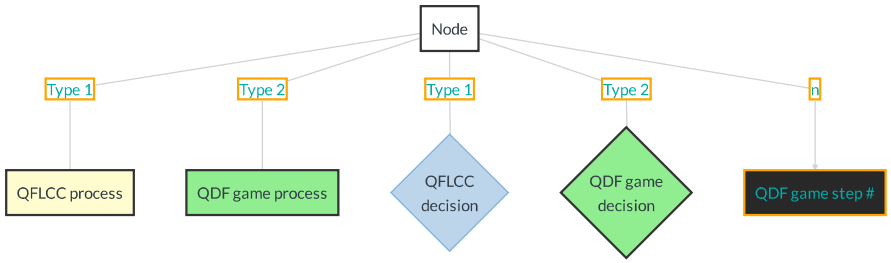
Expand All | Collapse All

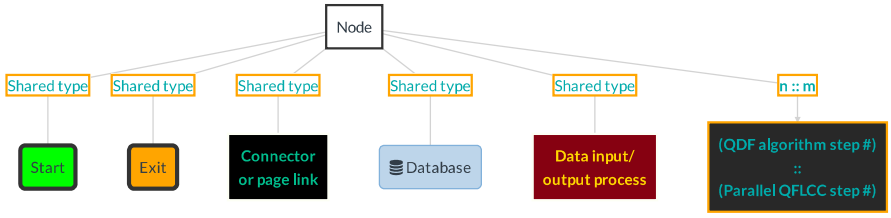
Program execution flowchart


Start
1
Run QFLCC pygame + >_ terminal?
Yes
->> Auto/manual Dataset Installation ->>
3
QFLCA DB
4
Choose dataset file(s)?
Yes/r
Or Restart Game
No
Alice & Bob's Quantum Doubles
6
Yes
Show QDF P losses & wins
No
7
Winning QDF Game Ps ?
No
Or Continue
17::8
Yes
Game Over?
Yes
Yes/b
Or Begin QFLCC
Yes
Or cont. via → link
Start
or ← link
10::1
Run >_ QFLCC program? </>
Yes
->> Install ->>
12::3
QFLCA DB
13::4
Restart dataset choice & analysis P analysis </>
Change Dataset
No
No
No
Yes



▼ Legend...





▼ Flowchart description...

This 17-step diagram represents the flow of the QDF game and QFLCC steps of the QFLCA program. These steps are presented in parallel, side-by-side, between programs/algorithms labeled as `n::m` e.g., [`17::8`] denotes `step 17 (from the QDF game algorithm)::8 (as part of the QFLCC algorithm)` after a successful installation and program run by the user. It starts with dataset files installation, then the selection of one of the datasets for probability `P analysis`. The results are further compared to the `IBM QDF circuit dataset` to `validate P results` between the two datasets by running the IBM QDF simulation module from the QFLCC program. Finally, the program `prompts` `the user to continue` by playing the QDF game for further validation of dataset results (`P analysis`). The diagram continues on the relevant page linked here.

▼ Expand/Collapse All...

```
* Click ▼Expand All to view all code blocks from QAI-LCode_QFLCC.py on this page.
* Click ▶Collapse All to selectively view a code block from QAI-LCode_QFLCC.py on this page.
```

▼Expand All ▶Collapse All

QAI-LCode_QFLCC Source Code

Downloadable source code is:
QAI-LCode_QFLCC file download

Common QDF game constants and variables

▼ Examples are: a constant = ASCII characters and a variable entry_stage ≥ 0 code:

These are QDF Game's frequently used constants and variables within QAI-LCode_QFLCC.py.

- A sample's expected output frequently called by the `QDF game functions`:

- A sample's sound volume change from the `promptGame()` function call via `Volume Settings` section of the QDF game module, `basic mode`:

```
> v
Input sound volume -=≡Σ(((° ͜ʖ °)) between [🔇 = 0 for muted, and 🔊 = 100 for loudest]: 56
Do you want to continue in prompt mode to input command? [y/n]y
Input the relevant command according to 'h' or 'help', 'cv', 'v' or 'volume' for sound volume change...
```

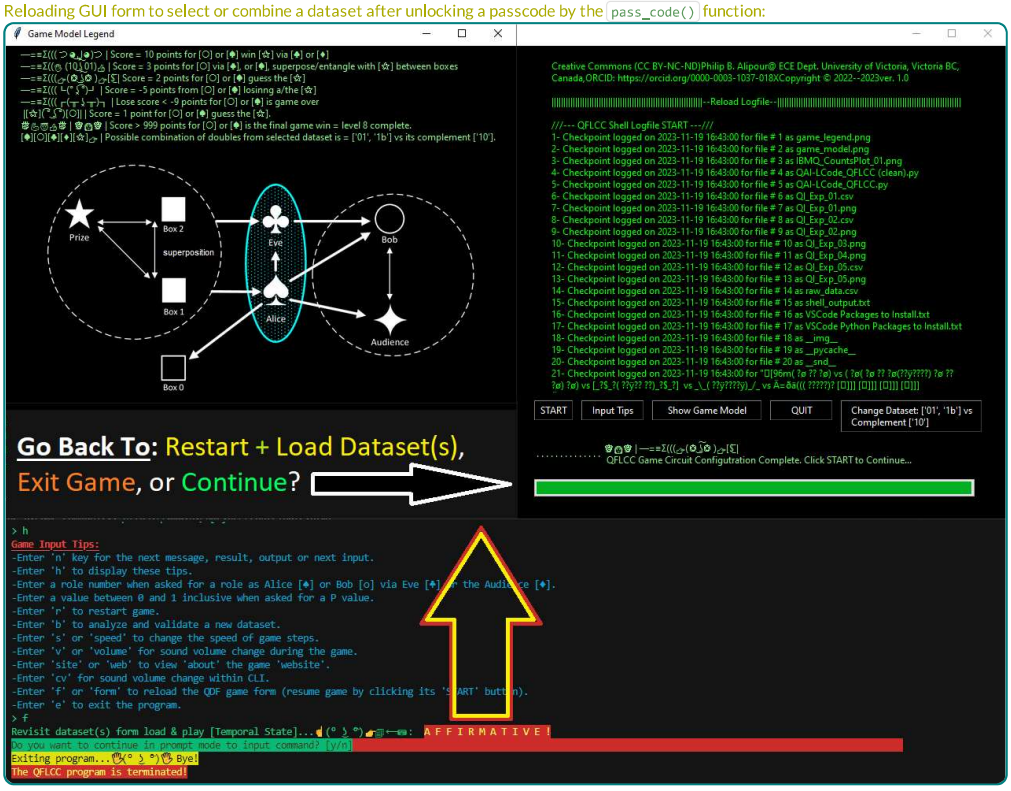
- A sample's sound volume change from the `promptGame()` function call via `Volume Settings` section of the QDF game module, `expert mode`:

```
> cv
Change game volume -=≡Σ(((° ͜ʖ °))... Choose from this range: [{quietist 🔇 ⇄ 🔉 = [0 to 6], quiet 🔉 = 6.1}, loudest 🔊 = 100]: 78.6
Volume converted and adjusted from the upper desktop's endpoint volume range of [-29, 0] = -5.8850
Do you want to continue in prompt mode to input command? [y/n]n
List of tried P's: [0.7, 0.5, 0.99]
```

- A sample's game speed change from the `promptGame()` function:

```
> s
Change game speed -=≡Σ((( ⊃◕‿◕)⊃... Choose from this range: [fastest = 0.1, slowest = 2]:0.5
```

- Reloading GUI form to select or combine a dataset after unlocking a passcode by the `pass_code()` function:



▼ 🔽 ℹ️ = Frequent QDF game constants and variables:

Frequent QDF game constants and variables: from QDF game header in QAI-LCode_QFLCC.py

```python
################################################################################
# Alice & Bob's Quantum Doubles game starts from here to validate data results...
################################################################################
__game_version__ = "ver.1.0"

__game_logo__="""
\033[92m  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\
\033[92m ( o.o )( o.o )( o.o )( o.o )( x.x )( o.o )( o.o )( o.o )( o.o )( o.o )( x.- )( o.o )( o.o )( o.o )( -.- )
\033[92m  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <
\033[92m  /\_/\  \033[96m   ___ __                ___     ____         ___                                 \033[92m   /\_/\
\033[92m ( o.o ) \033[96m  / _ \\ \              /   \   |  _ \       / _ \ //                             \033[92m   ( o.o )
\033[92m  > ^ <  \033[96m | |_| |\ \  ___   _____  \ O /   | |_) ) ___ | |_) )/____                        \033[92m   > ^ <
\033[92m  /\_/\  \033[96m |  _  | > \| \ \ / / __)  / _ \/\ |  _ ( / _ \|  _ < /  ._)                      \033[92m   /\_/\
\033[92m ( x.o ) \033[96m | | | |/ ^ \ |\ v /> _)  ( (_>  < | |_) | (_) ) |_) | () )                       \033[92m  ( x.x )
\033[92m  > ^ <  \033[96m |_| |_/_/ \_\_)> < \___)  \___/\/ |____/ \___/|  __/ \__/                        \033[92m   > ^ <
\033[92m  /\_/\  \033[96m   ____        / ^ \                            | |              _____            \033[92m   /\_/\
\033[92m ( o.x ) \033[96m  / __ \      /_/ \_\                           |/\             / _ \ \           \033[92m  ( o.o )
\033[92m  > ^ <  \033[96m | |__| |_   _  __  ___  _____ _   _ _  _     /  \    ___  _   _| |_) ) \   ___ ____ \033[92m   > ^ <
\033[92m  /\_/\  \033[96m |  __  | | | |/  \/ / |/ (    ) | | | | | |   / /\ \ / _ \| | | |  _ < > \ / __)  ._) \033[92m   /\_/\
\033[92m ( o.o ) \033[96m | |__| | |_| ( ()  <| / / | || |_| | |_| |  / /__\ ( (_) ) |_| | |_) ) ^ \> _| () ) \033[92m  ( -.x )
\033[92m  > ^ <  \033[96m  \____/ \___/ \__/\_\__/   \_)\___/| ._,_| /________\___/ \___/|  __/_/ \_\___)__/  \033[92m   > ^ <
\033[92m  /\_/\  \033[96m                                    | |                        | |                   \033[92m   /\_/\
\033[92m ( -.- ) \033[96m                                    |_|                        |_|                   \033[92m  ( o.o )
\033[92m  > ^ <  \033[96m  \033[91m v.1.0                                                                            \033[92m   > ^ <
\033[92m  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\  /\_/\
\033[92m ( o.x )( o.o )( o.o )( o.o )( -.- )( o.o )( o.o )( x.x )( o.o )( o.o )( o.o )( -.x )( o.o )( x.- )( o.o )
\033[92m  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <  > ^ <
"""
##############################################################################
# Instantiating variables and constants of animated states of loser, winner,
# dual(superposition), unknown. Alternatively, lists can be created in the
# future tests after threads, timer or sleep() and subprocess runs for combined
# datasets. For example, the constants excited1 and excited2 can be redefined
# in a list as excited = [excited1, excited2], or by the removal of the two
# constants hardcoding the emojis in excited = ["─=≡Σ((( つ◕ل͜◕)つ",
# "─=≡Σ(((✌(☆ ͜ʖ ☆)✌ [☆]"] and later called for the participant's win, as
# animated by the function winner_show() within its for-loop.
##############################################################################
prize="[☆]"
alice="[♠]"
bob="[○]"
eve="[♣]"
audience="[♦]"
crownBob="♕💪🤴👍♕"
crownAlice="♔👸♔"
here="👇"
go="👉"

excited1 = "─=≡Σ((( つ◕‿ل◕)つ"
excited2 = "─=≡Σ(((✌(☆ ͜ʖ ☆)✌ [☆]"

grounded1= "─=≡Σ(((┏(° ͜ʖ ͡°)┛"
grounded2= "─=≡Σ(((┗(° ͜ʖ ͡°)┛"
grounded3= "─=≡Σ(((┏(╥ ͜ʖ ╥)┓"

dual3="─=≡Σ((👎 (11 ʖ̯ 00)👍"
dual1="─=≡Σ((🖐 (10 ʖ̯ 01)👍"
dual2="─=≡Σ((☝ (01 ʖ̯ 10)🖐"

guesser1="─=≡Σ(((💪(◐ ͜ʖ◐ )👉"
guesser2="─=≡Σ(((☝(◐ ͜ʖ◐ )👍"
guesser3="─=≡Σ(((👉(⚙ ͜ʖ⚙ )👉[$̲̅|"

classical1="─=≡Σ((🖐(° ͜ʖ ͡°)🖐"
classical2="─=≡Σ((👍(° ͜ʖ ͡°)👍"
classical3="─=≡Σ(((👉(⚙ ͜ʖ⚙ )👉[$̲̅|"

errorP1=" ¯\_( ◐ ͜ʖ◐ )_/¯"
errorP2=" ¯\_( ◕ ͜ʖ◕ )_/¯"

aimTarget=" ||[$̲̅(̲̅ ⊚ ͜ʖ ⊚)̲̅$̲̅]||"
helperTarget1=" |[☆]( ͡° ͜ʖ ͡°)[○]|"
helperTarget2=" |[♠]( ͡° ͜ʖ ͡°)[☆]|"
###########################################################################################
# These are the QDF game description, logo and copyright details.
###########################################################################################
__game_banner__ = """\n"\033[96m( ͡° ͜ʖ ͡°) vs ( ͡°( ͡° ͜ʖ ͡°( ͡▧ ͜ʖ ͡▧) ͡° ͜ʖ ͡°) ͡°) vs [$̲̅(̲̅ ⊚ ͜ʖ ⊚)̲̅$̲̅]  vs \
 ¯\_( ◐ ͜ʖ◐ )_/¯ vs ─=≡Σ((( つ◕‿ل◕)つ [♥]]] [♦]]] [♣]]] [♠]]] ─=≡Σ(((✌(☆ ͜ʖ ☆)✌ |||||||||||||||||||||| \033[92m"""
__game_description__= "Alice & Bob's Quantum Doubles... This QFLCC game version of the QAI-LCode is basic. \
More revisions to come, as the dataset grows on the number of trials on 1 or more QDF circuits for simulating. \
the QDF game. The QDF game simulates events of the dataset's quantum and classical parameters in observing a \
thermodynamic system = { QDF game environment and its participants Alice, Bob, Eve, Audience } based on Refs. [1-2,4] \
of the Data in Brief, Elsevier J article."
print(Back.RESET+fg.lightgreen,__game_logo__,__game_banner__ + __game_description__ ,
      Back.RESET+fg.silver+__game_version__,"\n"), sleep(2)
"""- Colored banner to introduce the QDF game and its goal. These common constants and variables are
     frequently used during game play."""
entry_stage = 4 # Set this to 0 if the game is run as a separate file or program.
dir_flag = 0 # Directory flag is set to 1 if dataset directory access is asked by user to analyze in safe mode.

qdf_bits_flag = [False, False] # Flag state 0 == hide, 1 == show the qdf bit values from any dataset.
qdf_bits_flag[0] = False # By default set at least this flag to 0 == hide the qdf bit values from any dataset.

def qdf_bits_flagset():
 # A function to switch between states of 0 == hide and 1 == show for the qdf bit values of any dataset.
 global qdf, bitpair, qdf_bits_flag
 if (qdf_bits_flag[0] == False) and (qdf_bits_flag[1] == False):
     qdf = '[##]' # Mask this qdf value in case of qdf_bits both flags are false.
     bitpair = '[##]' # Mask this bitpair value in case of qdf_bits both flags are false.
 if (qdf_bits_flag[0] == False) and (qdf_bits_flag[1] == True):
     qdf = qdf # Unmask this qdf value in case of qdf_bits 2nd flag is true.
     bitpair = bitpair # Unmask this bitpair value in case of qdf_bits 2nd flag is true.
```

▼ Expand/Collapse All...

```
* Click Expand All to view all code blocks from QAI-LCode_QFLCC.py on this page.
* Click Collapse All to selectively view a code block from QAI-LCode_QFLCC.py on this page.
```

Expand All | Collapse All

QAI-LCode_QFLCC Module for the QDF Game

▼ ⚠ Module description and code usage warning:

The current main module is the source file `QAI-LCode_QFLCC.py`. This module will be split into the `QDF-game.py` and `QAI-LCode_QFLCC.py` for import. The `QDF-game.py` is already within the file `QAI-LCode_QFLCC.py` file. The updated version to import the smaller `QAI-LCode_QFLCC.py` for the `QDF game` is under development as the next version of the code. QAI-LCode_QFLCC file download

- `ExitMyProgram`

  Bases: `Exception`

  - Exception used to exit program.

  ▼ Source code in `QAI-LCode_QFLCC.py`

```
104 class ExitMyProgram(Exception):
105     """- Exception used to exit program."""
```

`bg`

▼ Source code in `QAI-LCode_QFLCC.py`

```
 91 class bg:
 92         black='\033[40m'
 93         red='\033[41m'
 94         green='\033[42m'
 95         orange='\033[43m'
 96         blue='\033[44m'
 97         purple='\033[45m'
 98         cyan='\033[46m'
 99         lightgrey='\033[47m'
100         """- Colored background (bg) class for sectioning and highlighting the
101         QF-LCC algorithm step, checkpoint, computer operation, error, or HALT."""
```

`lightgrey = '\x1b[47m'` `class-attribute` `instance-attribute`

- Colored background (bg) class for sectioning and highlighting the QF-LCC algorithm step, checkpoint, computer operation, error, or HALT.

`fg`

▼ Source code in `QAI-LCode_QFLCC.py`

```
 70 class fg:
 71         black='\033[30m'
 72         silver="\033[0;38;5;7m"
 73         red='\033[31m'
 74         green='\033[32m'
 75         orange='\033[40m'
 76         blue='\033[34m'
 77         purple='\033[35m'
 78         cyan='\033[36m'
 79         lightgrey='\033[37m'
 80         darkgrey='\033[90m'
 81         lightred='\033[91m'
 82         lightgreen='\033[92m'
 83         yellow='\033[93m'
 84         lightblue='\033[94m'
 85         pink='\033[95m'
 86         lightcyan='\033[96m'
 87
 88         """- Colored foreground (fg) class for sectioning and highlighting the
 89         QF-LCC algorithm step, checkpoint, computer operation, error, or HALT."""
```

`lightcyan = '\x1b[96m'` `class-attribute` `instance-attribute`

- Colored foreground (fg) class for sectioning and highlighting the QF-LCC algorithm step, checkpoint, computer operation, error, or HALT.

`build(debug)`

- Build production assets.

▼ Source code in `QAI-LCode_QFLCC.py`

```
@cli.command()
@click.option("-d", "--debug", help="Include debug output.")
def build(debug):
    """- Build production assets."""
```

`cli()`

Main entry point.

▼ Source code in QAI-LCode_QFLCC.py

```
@click.group()
def cli():
    """Main entry point."""
```

▼ Expand/Collapse All...

```
* Click Expand All to view all code blocks from QAI-LCode_QFLCC.py on this page.
* Click Collapse All to selectively view a code block from QAI-LCode_QFLCC.py on this page.
```

Expand All | Collapse All

QDF Game Functions and Calls

▼ ≙ QDF game step log, colors, GUI and its other settings:

To log game steps as the game progresses, including errors, exit, GUI and help for the user. Function() list from top to bottom as ordered:

- ▼ Source code in QAI-LCode_QFLCC.py

```
def entry_add():
##############################################################################################
# This function logs program events onto the stdout file
##############################################################################################
     global entry_stage  # Redefine the entry count as global.
     entry_stage+=1
     idxplus = len(res)+entry_stage
     subprocess.run("echo {}- Checkpoint logged on {} for the QFLCC Game:\
 Alice and Bob's Quantum Doubles {}".format(idxplus, now.strftime("%Y-%m-%d %H:%M:%S"), __game_version__),
 shell=True, stdout=file_)
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def flash_color(object, color):
    object.config(foreground = next(color), bg='black')
    root.after(100, flash_color, object, color)
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def display_bye_msg():
    # Play Windows exit sound.
    winsound.PlaySound("SystemExit", winsound.SND_ALIAS)
    root.wm_attributes("-topmost", 1)
    root.withdraw() # Hide this form.
    bye = msgbox.askquestion(title=f"Alice & Bob's Quantum Doubles {__game_version__} Message",
                             message = 'Are you sure you want to exit? If Yes, ...👋 (° ͜ʖ °)👋 Goodbye!',
                             icon = 'warning', parent=root)
    if bye == 'yes':
        game_sound = pygame.mixer.Sound('__snd__/goodbye1.wav')
        game_sound.play()
        Quit()
    else:
        root.wm_attributes("-topmost", 1)
        root.deiconify() # Unhide this form.
    """Exit game if user chooses after the goodbye message."""
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def flagClose():
    root.withdraw() # Hide this form.
    root.quit()
    #-------------------------------------------------------------------------------
    # Enable the next lines to destroy this form after quit (or by duration afterwards
    # using sleep) as a permanent close solution. This is to not revisit the dataset(s)
    # load & play root form, as it is already destroyed after t seconds!
    #-------------------------------------------------------------------------------
    #sleep(50)       # Dont close for t = 50 seconds, for example...
    #root.destroy()  # To destroy the root form.
    """Quit or destroy the game app form and get back to CLI as user's interface. The quit
 command gives user the illusion that the form is closed/gone, which in fact can be later
 called upon by the option 'form' from the terminal. """
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def Quit():
    game_sound = pygame.mixer.Sound('__snd__/goodbye1.wav')
    game_sound.play()
    print(fg.black + Back.YELLOW +'\nExiting program...✋ (° ͜ʖ °)✋  Goodbye!'+ Back.RESET), sleep(3)
    print(fg.yellow + Back.RED+"The QFLCC program is terminated!" + Back.RESET)
    sys.exit(1) # Force this in case of abnormal exit or program termination.
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def Help():
    msgbox.showinfo(title='Input Tips', message='Input tips:\n \n-Enter \'n\' key or \'next\' for \
the next message or input. \n-Enter \'h\' or \'help\' to display these tips. \n-Enter the number \
for a game player role when prompted. \n-Enter a value between 0 and 1, or 0 or 1 for a P value. \
\n-Enter \'site\' or \'web\' to view \'about\' the game \'website\'. \
\n-Enter \'s\' or \'speed\' to change the speed of game steps. \
\n-Enter \'v\' or \'volume\' for sound volume change during play. \n-Enter \'cv\' for sound volume \
change within CLI. \n-Enter \'f\' or \'form\' to reload this form. \
\n-Enter \'dir\' or \'sm dir\' for a new dataset analysis & simulation to feed this game.*\
\n*-This command requires a passcode to see dataset results in a Safe Mode (SM:>>) environment!\
\n-Enter \'e\' or \'exit\' to exit the game.')
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def open_new_win():
    ###########################################################################
    # This function creates a new window based on the QDF game legend and rules. #
    ###########################################################################
    imagelist1 = ["game_legend.png"]
    photo = PhotoImage(file=imagelist1)
    width = photo.width()
    height = photo.height()
    fore_win=Toplevel(root)
    fore_win.wm_attributes("-topmost", 1) # these two lines will focus on the window not terminal
    fore_win.focus_force()
    fore_win.title("Game Model Legend")
    fore_win.geometry("650x450")
    fore_win.configure(background='black')
    image = Image.open('game_legend.png').convert("RGB")
    resized = image.resize((550, 300),Image.LANCZOS) # Resize the image and antialias it from the uploaded file.
    display = ImageTk.PhotoImage(resized)
    label = Label(fore_win, image=display, background="black") # Display it within a label.
    legend_label = Label(fore_win,
                         text=excited1+f' | Score = 10 points for {bob} or {alice} win {prize} via {eve} or {audience} \n'
                         + dual1+f' | Score = 3 points for {bob} via {eve}, or {alice}, \
superpose/entangle with {prize} between boxes \n'+ guesser3+f' Score = 2 points for {bob} or {alice} guess the {prize} \n'
                         +  grounded2+f' | Score = -5 points from {bob} or {alice} losing a/the {prize} \n'+ grounded3
                         +f' | Lose score < -9 points for {bob} or {alice} is game over \n'+ helperTarget1
                         +f' | Score = 1 point for {bob} or {alice} guess the {prize}.  \n'+ crownBob+' | '+ crownAlice
                         +f' | Score > 999 points for {bob} or {alice} is the final game win = level 8 complete.  \n'
                         + alice + bob + eve + audience + prize +go
                         +f' | Possible combination of doubles from selected dataset \
is = ' + buttonInvisible["text"] + '.\n', justify="left" , bg="black",  fg="lightgreen").pack()
    label.image = display
    label.pack()
    # Set minimum the foreground window size value
    fore_win.minsize(650, 450)
    # Set maximum the foreground window size value
    fore_win.maxsize(650, 450)
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def restart_QFLCC():
    ###########################################################
    # The following restarts QFLCC for choosing a dataset  #
    ###########################################################
    global win_min
    #root.iconify()
    win_min == True
    flagClose() # An option to choose between destroying/quit the present form.
    subprocess.run(["python", "QAI-LCode_QFLCC.py"])
```

- ▼ ℹ ≡ QDF game intro GUI. . .

Main QDF game GUI: code from QAI-LCode_QFLCC.py or QDF-game-gui.py (under construction)

```
    ################################################################
    # Initialize pygame, then sound mixer to play introductory retro
    # music and welcome message sound files and create other windows
    # as a GUI.
    ################################################################
    from tkinter import *
    import tkinter as tk
    from itertools import cycle
    import pygame  # To play music or sounds in the QDF game.

    def flash_color(object, color):
    object.config(foreground = next(color), bg='black')
    root.after(100, flash_color, object, color)
    """ Coloring the flash object from pygame for the user's interface (UI)."""

    pygame.init()
    pygame.mixer.init()

    game_sound = pygame.mixer.Sound('__snd__/game_music2.wav')
    game_sound.play(), sleep(2)

    game_sound = pygame.mixer.Sound('__snd__/welcome2.wav')
    game_sound.play()

    """ Initialize pygame and sound mixer to play introductory retro
    music and welcome message sound files and other windows as a GUI."""

    from PIL import Image, ImageTk
    import tkinter.messagebox as msgbox

    def display_bye_msg():
    # Play Windows exit sound.
    winsound.PlaySound("SystemExit", winsound.SND_ALIAS)
    root.wm_attributes("-topmost", 1)
    root.withdraw() # Hide this form.
    bye = msgbox.askquestion(title=f"Alice & Bob's Quantum Doubles {__game_version__} Message",
                             message='Are you sure you want to exit? If Yes, ...👋 (° ͜ʖ °)👋 Bye!',
                             icon='warning', parent=root)
    if bye == 'yes':
         game_sound = pygame.mixer.Sound('__snd__/goodbye1.wav')
         game_sound.play()
         Quit()
    else:
         root.wm_attributes("-topmost", 1)
         root.deiconify() # Unhide this form.
    """Exit game if user chooses after the bye message."""

    def flagClose():
    root.withdraw() # Hide this form.
    root.quit()
    #-----------------------------------------------------------------------------
    # Enable the next lines to destroy this form after quit (or by duration afterwards
    # using sleep) as a permanent close solution. This is to not revisit the dataset(s)
    # load & play root form, as it is already destroyed after t seconds!
    #-----------------------------------------------------------------------------
    #sleep(50)       # Dont close for t = 50 seconds, for example...
    #root.destroy()  # To destroy the root form.
    """Quit or destroy the game app form and get back to CLI as user's interface. The quit
    command gives user the illusion that the form is closed/gone, which in fact can be later
    called upon by the option 'form' from the terminal. """

    def Quit():
         game_sound = pygame.mixer.Sound('__snd__/goodbye1.wav')
         game_sound.play()
         print(fg.black + Back.YELLOW +'\nExiting program...👋 (° ͜ʖ °)👋  Bye!'+ Back.RESET), sleep(3)
         print(fg.yellow + Back.RED+"The QFLCC program is terminated!" + Back.RESET)
         sys.exit(1) # Force this in case of abnormal exit or program termination.
    """Exit program and system with a message before the terminal."""

    def Help():
         msgbox.showinfo(title='Input Tips', message='Input tips:\n \n-Enter \'n\' key or \'next\' for \
    the next message or input. \n-Enter \'h\' or \'help\' to display these tips. \n-Enter the number \
    for a game player role when prompted. \n-Enter a value between 0 and 1, or 0 or 1 for a P \
    value. \n-Enter \'s\' or \'speed\' to change the speed of game steps. \n-Enter \'v\' or \'volume\' for \
    sound volume change during play. \n-Enter \'cv\' for sound volume change within CLI. \n-Enter \'f\' or \
    \'form\' to reload this form. \n-Enter \'e\' or \'exit\' to exit the game.')
    """Helping tips to start the game by choosing participants and a P value, according to the QDF game model."""

    def open_new_win():
       ###############################################################################
       # This function creates a new window based on the QDF game legend and rules. #
       ###############################################################################
       imagelist1 = ["game_legend.png"]
       photo = PhotoImage(file=imagelist1)
       width = photo.width()
       height = photo.height()
       fore_win=Toplevel(root)
       fore_win.wm_attributes("-topmost", 1) # these two lines will focus on the window not terminal
       fore_win.focus_force()
       fore_win.title("Game Model Legend")
       fore_win.geometry("650x450")
       fore_win.configure(background='black')
       image = Image.open('game_legend.png').convert("RGB")
       resized = image.resize((550, 300),Image.LANCZOS) # Resize the image and antialias it from the uploaded file.
       display = ImageTk.PhotoImage(resized)
       label = Label(fore_win, image=display, background="black") # Display it within a label.
       legend_label = Label(fore_win,
                            text=excited1+f' | Score = 10 points for {bob} or {alice} win {prize} via {eve} or {audience} \n'
                            + dual1+f' | Score = 3 points for {bob} via {eve}, or {alice}, \
    superpose/entangle with {prize} between boxes \n'+ guesser3+f' Score = 2 points for {bob} or {alice} guess the {prize} \n'
                            +  grounded2+f' | Score = -5 points from {bob} or {alice} losing a/the {prize} \n'+ grounded3
                            +f' | Lose score < -9 points for {bob} or {alice} is game over \n'+ helperTarget1
                            +f' | Score = 1 point for {bob} or {alice} guess the {prize}.  \n'+ crownBob+' | '+ crownAlice
                            +f' | Score > 999 points for {bob} or {alice} is the final game win = level 8 complete.  \n'
                            + alice + bob + eve + audience + prize +go
                            +f' | Possible combination of doubles from selected dataset \
```

```
is = ' + buttonInvisible["text"] + '.\n', justify="left" , bg="black",  fg="lightgreen").pack()
   label.image = display
   label.pack()
   # Set minimum the foreground window size value
   fore_win.minsize(650, 450)
   # Set maximum the foreground window size value
   fore_win.maxsize(650, 450)

win_min=0
def restart_QFLCC():
#########################################################
# The following restarts QFLCC for choosing a dataset  #
#########################################################
global win_min
#root.iconify()
win_min == True
flagClose() # An option to choose between destroying/quit the present form.
subprocess.run(["python", "QAI-LCode_QFLCC.py"])

#########################################################
# Mimic an animated GIF displaying a series of GIFs an
# animated GIF was used to create the series of GIFs with
# a common GIF animator utility.
# Buttons and usage help options are appended and displayed
# at the end of log file and other game file
# installation/download.
#########################################################
from tkinter import ttk
import asyncio

root = Tk()

# Adjust size...
#root.geometry("1550x600")
width = 1550 # Width
height = 600 # Height
screen_width = root.winfo_screenwidth()  # Width of the screen
screen_height = root.winfo_screenheight() # Height of the screen

# Calculate Starting X and Y coordinates for Window
x = (screen_width/2) - (width/2)
y = (screen_height/2) - (height/2)

root.geometry('+%d+%d' % (x, y))
root.wm_attributes("-topmost", 1) # These two lines will focus on the window not terminal.
root.focus_force()

def disable_event():
pass
root.protocol("WM_DELETE_WINDOW", disable_event)  # This disables the [X] button on the root window to satisfy user's game options.
     # No matter how many times the user clicks on [X], nothing happens, and so is directed to other exit or resume game options.

# Create a sprite bank (array list) to animate from files.
imagelist = ["__img__/logo_01.gif", "__img__/logo_01.gif", "__img__/logo_02.gif", "__img__/logo_02.gif",
          "__img__/logo_02.gif", "__img__/logo_03.gif",  "__img__/logo_03.gif", "__img__/logo_04.gif",
          "__img__/logo_05.gif", "__img__/logo_06.gif", "__img__/logo_06.gif", "__img__/logo_06.gif"]

# Create a sprite bank (array list) from the defined ASCII characters by their corresponding char variables.
charslist = [f'{crownBob} | {dual1}' , f'{crownBob} | {dual1}', f'{crownBob} | {dual2}',
          f'{crownBob} | {dual2}', f'{crownBob} | {dual3}', f'{crownBob} | {dual3},' f'{crownBob} | {excited1}',
          f'{crownBob} | {excited1}', f'{crownBob} | {excited1}', f'{crownBob} | {excited2}',
          f'{crownBob} | {excited2}', f'{crownBob} | {excited2}', f'{crownAlice} | {guesser1}',
          f'{crownAlice} | {guesser1}', f'{crownAlice} | {guesser2}', f'{crownAlice} | {guesser2}',
          f'{crownAlice} | {guesser3}', f'{crownAlice} | {guesser3}', f'{crownAlice} | {classical1}',
          f'{crownAlice} | {classical1}', f'{crownAlice} | {classical2}', f'{crownAlice} | {classical3}',
          f'{crownAlice} | {classical3}']
color_list = ["cyan", "yellow", "red", "yellow", "blue", "lightgreen"]

progressbar = ttk.Progressbar(length=750, style="green.Horizontal.TProgressbar")

# Full progress bar...
progressbar.step(99.9)
progressbar.place(x=950,y=550, width=560)

# Set minimum window size value.
root.minsize(1550, 600)

# Set maximum window size value.
root.maxsize(1550, 600)

# Extract width and height info.
photo = PhotoImage(file=imagelist[0])
width = photo.width()
height = photo.height()

canvas = Canvas(width=width, height=height)

# Position the canvas for the gif image.
canvas.place(x=30, y=20)

root.title(f"Alice & Bob's Quantum Doubles {__game_version__}")
root.configure(background='black')

filename='shell_output.txt'

def update_text():
# Configuring the text in Label widget.
my_label0.configure(text="Loading the game model and circuit. Click START to continue...")
subprocess.call(['game_model.png'], shell=True)
subprocess.call([f'{pngfile}.png'], shell=True)

def change_color():
my_label0.config(bg= "gray51", fg= "red")

f = open("shell_output.txt", "r")
file_string = f.read()
```

```
    my_label0 = tk.Label(root, text =  __license__ + __author__+ __location__+','
                        +__ORCID__+ __copyright__+ __version__
                        +"\n \n|||||||||||||||||||||||||||||||||||||||||||||||||\
|||||||||||||||--Reload Log_file--|||||||||||||||||||||||||||||||||||||||||||||||||\
||||||||||||||||||||||||||||||\n \n"+str(file_string), fg="#000fff000", background="black",
    width=80, height =28, wraplength=525, anchor='n', justify="left")
    my_label0.place(x=950, y=15)

    # Create a list of image objects.
    giflist = [] # This is to store and run image sprite in .gif.
    for imagefile in imagelist:
    photo = PhotoImage(file=imagefile)
    giflist.append(photo)

    anim_label1 = tk.Label(root, text = '. . . . . .',
                        fg="lightgreen" , background="black", width=80, height =2, wraplength=500,
                        anchor="w", justify="left")
    anim_label1.place(x= 950, y= 500)
    anim_label0 = tk.Label(root, text = dual1, fg="cyan" , background="black", width=80, height=2,
                        wraplength=500, anchor="w", justify="left")
    anim_label0.place(x= 950, y= 500)

    async def mainT():
    await asyncio.sleep(0.001)  # Sprite is updated every second and then other asynchronous updating tasks
                                # take place in the outer loops.

    # Loop through the gif image objects for a while.
    for j in range(0, 30):
    for gif in giflist:
         canvas.delete(ALL)
         canvas.create_image(width/2.0, height/2.0, image=gif)
         canvas.update()
         progressbar.step(j*5)
         anim_label0.place(x= 950+j*4, y= 500)
         #sleep(0.001)
    for clr in color_list:
         for sprite in charslist:
              anim_label0.config(text=sprite+' . . . . . Loading . . .')
              anim_label0.update()
              #asyncio.run(mainT()) # Sprite is updated every second and then other asynchronous updating tasks
                                # take place in the outer loops.
              #asyncio.sleep(0.5)  # Sprite is updated every second and then other asynchronous updating tasks
                                # take place in the outer loops.
         anim_label0.config(fg=clr)
         anim_label0.update()
         sleep(0.02)

    pygame.time.delay(2)

    button0 = Button(root, text = 'START', bg='black', fg='white', command = flagClose)
    button1 = Button(root, text = 'Input Tips', bg='black', fg='white', command = Help)
    button3 = Button(root, text="QUIT", bg='black', fg='white', command = display_bye_msg)
    button2 = Button(root, text="Show Game Model", bg='black', fg='white', command = open_new_win)
    button4 = Button(root, text=f'Change Dataset: [##] vs Complement [##]', bg='black', fg='white',
                    justify=LEFT, wraplength=160, command = restart_QFLCC)
    buttonInvisible = Button(root, text=f'[##] vs its complement [##]', bg='black', fg='white', command = restart_QFLCC)
    buttonInvisible.pack_forget() # This button remains invisible until made visible through .pack() command.
    button0.place(x=950, y=450, width=50)
    button1.place(x=1010, y=450, width=80)
    button2.place(x=1100, y=450, width=140)
    button3.place(x=1250, y=450, width=80)
    button4.place(x=1340, y=450, width=180)

    def update_btn4_text():
    # This function focuses on Button4 changes and updates in including/excluding datasets per user request...
        global qdf, bitpair, qdf_bits_flag
        if  (qdf_bits_flag[0] == False) and (qdf_bits_flag[1] == True):
             qdf_bits_flagset() # Call this function to show the qdf bit values of the employed dataset.
             button4.configure(text=f'Change Dataset: {qdf} vs Complement {bitpair}')
             buttonInvisible.configure(text=f'{qdf} vs its complement {bitpair}')
             button4.update()
        else:
            button4.configure(text=f'Change Dataset: [##] vs Complement [##]')
            buttonInvisible.configure(text=f'[##] vs its complement [##]')

    for k in range(0, 1):
    for clr in color_list:
         for sprite in charslist:
              anim_label0.config(text='Configuring Logic Conditions for the chosen Circuit and Dataset...\n'+sprite,
                                 wraplength=400)
              anim_label0.update()
              #asyncio.run(mainT())
              anim_label0.place(x= 950-j/2+100, y= 500)
              sleep(0.1)  # Sprite is updated every second and then other asynchronous updating tasks take place in the outer loops.
         anim_label0.config(fg=clr)
         anim_label0.update()

    anim_label0.config(text=sprite+'\n QFLCC Game Circuit Configuration Complete. Click START to Continue...',
                       wraplength=400)
    anim_label0.update()

    root.mainloop() # Run the Tkinter main loop.
```

- ▼ ▼ i ≙ Common QDF game state variables:

Variables for defining QDF game states in QAI-LCode_QFLCC.py

```
1820  ######################################################
1821  # Variables for defining QDF game states and their
1822  # animation functions.
1823  ######################################################
1824  game_points = []  # This is to keep win/lose score.
1825  level_points= []  # This is to keep track of the level completion score.
1826  levels = 0
1827  points = 0
1828  participantMain=""
1829  participantMid=""
1830  spd = [] # Game speed to be changed by user and register to alter the QDF game steps pace
1831           # of execution.
1832  spd = 1.1 # QDF game speed of 1.1 is a the default value.
1833  ######################################################
1834  # Defining a simple animation function of game states.
1835  ######################################################
```

▼ ▼ i Expand/Collapse All...

* Click ▼Expand All to view all code blocks from QAI-LCode_QFLCC.py on this page.
* Click ▶Collapse All to selectively view a code block from QAI-LCode_QFLCC.py on this page.

▼Expand All ▶Collapse All

▼ ▼ i ≡ QDF game level, P Classification and Animation:

i Animation, P classification and game state functions for an assigned participant given by a role number. Function() list from top to bottom as ordered:

- ▼ ▼Source code in QAI-LCode_QFLCC.py

```
1790  def update_btn4_text():
1791  # This function focuses on Button4 changes and updates in including/excluding datasets per user request...
1792      global qdf, bitpair, qdf_bits_flag
1793      if  (qdf_bits_flag[0] == False) and (qdf_bits_flag[1] == True):
1794           qdf_bits_flagset() # Call this function to show the qdf bit values of the employed dataset.
1795           button4.configure(text=f'Change Dataset: {qdf} vs Complement {bitpair}')
1796           buttonInvisible.configure(text=f'{qdf} vs its complement {bitpair}')
1797           button4.update()
1798      else:
1799          button4.configure(text=f'Change Dataset: [##] vs Complement [##]')
1800          buttonInvisible.configure(text=f'[##] vs its complement [##]')
```

- ▼ ▼Source code in QAI-LCode_QFLCC.py

```
1835  def participants():
1836       global participantMain, participantMid
1837       if takeUIn == 1:
1838           participantMain = bob
1839       elif takeUIn == 2:
1840            participantMain = alice
1841       if takeUIn2 == 3:
1842            participantMid = eve
1843       elif takeUIn2 ==4:
1844            participantMid = audience
1845       """Animation function for an assigned participant by a role number."""
```

- ▼ ▼Source code in QAI-LCode_QFLCC.py

```
1847  def winner_show():
1848       global points
1849       game_sound = pygame.mixer.Sound('__snd__/gameprize_state.wav')
1850       game_sound.play()
1851       points += 10 # Add points for each win of a QDF.
1852       game_points.append(points) # Add result to the scoresheet.
1853       print(f'Your score is: {points}. Scoresheet is: {game_points}')
1854       for count in range(5):
1855          print(fg.lightgreen + excited1, end="\r", flush=True), sleep(spd)  # Default QDF game speed is = 1.1.
1856          print(excited2, end="\r"+Fore.LIGHTYELLOW_EX, flush=True), sleep(spd)
1857          if count == 5 or keyboard.is_pressed("n"):
1858               break
```

- ▼ ▼Source code in QAI-LCode_QFLCC.py

```
def winner_next():
            entry_add()
            level_show()
            subprocess.run('echo "{}- ///--- Win Entry --- \
{} /// Level: {} Scored: {} at {}"'.format(entry_stage, excited2, levels, points, now.strftime("%Y-%m-%d %H:%M:%S")),
shell=True, stdout=file_)  # Date last I/O file entry.
            print("\n",f'Next...'+fg.lightgreen+' Level:', levels, f'{here}'+fg.orange)
```

- ▼ Source code in `QAI-LCode_QFLCC.py`

```
def loser_show():
     global points
     game_sound = pygame.mixer.Sound('__snd__/gameloser_state.wav')
     game_sound.play()
     points -= 5 # Lose points for each loss of a QDF.
     game_points.append(points) # Add result to the scoresheet.
     print(f'Your score is: {points}. Scoresheet is: {game_points}')
     for count in range(4):
            print(fg.orange + grounded1, end="\r", flush=True), sleep(spd)
            print(grounded2, end="\r", flush=True), sleep(spd)
            print(fg.lightred+grounded3, end="\r"+Fore.LIGHTYELLOW_EX, flush=True), sleep(spd)
            if count == 4 or keyboard.is_pressed("n" or "s"):
             break
```

- ▼ Source code in `QAI-LCode_QFLCC.py`

```
def loser_next():
            entry_add() # Reopen log file and register an event entry.
            level_show()
            subprocess.run('echo "{}- ///--- Lose Entry --- {} /// Level: {} \
Scored: {} at {}"'.format(entry_stage, grounded3, levels, points, now.strftime("%Y-%m-%d %H:%M:%S")),
shell=True, stdout=file_) # Date last I/O file entry.
            print("\n",f'Next...'+fg.lightgreen+' Level:', levels, f'{here}'+fg.orange)
```

- ▼ Source code in `QAI-LCode_QFLCC.py`

```
def dual_show():
     global points
     game_sound = pygame.mixer.Sound('__snd__/gamedual_state.wav')
     game_sound.play()
     points += 2 # Add points for each gain of a duality in QDF.
     game_points.append(points) # Add result to the scoresheet.
     print(f'Your score is: {points}. Scoresheet is: {game_points}')
     for count in range(5):
            print(fg.lightgreen + dual1, end="\r", flush=True), sleep(spd)
            print(fg.orange + dual2, end="\r", flush=True), sleep(spd)
            print(fg.red + dual3, end="\r"+Fore.LIGHTYELLOW_EX, flush=True), sleep(spd)
            if count == 5 or keyboard.is_pressed("n" or "s"):
             break
```

- ▼ Source code in `QAI-LCode_QFLCC.py`

```
def dual_next():
            sleep(1)
            entry_add()
            level_show()
            subprocess.run('echo "{}- ///--- Dual Entry --- {} /// Level: {} \
Scored: {} at {}"'.format(entry_stage, dual3, levels, points, now.strftime("%Y-%m-%d %H:%M:%S")),
shell=True, stdout=file_) # Date last I/O file entry.
            print("\n",f'Next...'+fg.lightgreen+' Level:', levels, f'{here}'+fg.orange)
```

- ▼ Source code in `QAI-LCode_QFLCC.py`

```
def guesser_show():
     global points
     game_sound = pygame.mixer.Sound('__snd__/gamedual_state.wav')
     game_sound.play()
     points += 3 # Add points for each guess of a QDF, close but not concluded.
     game_points.append(points) # Add result to the scoresheet.
     print(f'Your score is: {points}. Scoresheet is: {game_points}')
     for count in range(5):
            print(fg.red + guesser1, end="\r", flush=True), sleep(spd)
            print(fg.orange + guesser2, end="\r", flush=True), sleep(spd)
            print(fg.green + guesser3, end="\r"+Fore.LIGHTYELLOW_EX, flush=True), sleep(spd)
            if count ==5 or keyboard.is_pressed("n") or keyboard.is_pressed("s"):
              break
```

- ▼ Source code in `QAI-LCode_QFLCC.py`

```
def guesser_next():
             sleep(1)
             entry_add()
             level_show()
             subprocess.run('echo "{}- ///--- Guesser Entry --- {} /// Level: {} \
Scored: {} at {}"'.format(entry_stage, guesser3, levels, points, now.strftime("%Y-%m-%d %H:%M:%S")),
shell=True, stdout=file_) # Date last I/O file entry.
             print("\n",f'Next...'+fg.lightgreen+' Level:', levels, f'{here}'+fg.orange)
```

- ▼ Source code in `QAI-LCode_QFLCC.py`

```
def helper_show():
     global points
     game_sound = pygame.mixer.Sound('__snd__/gamehelper_state.wav')
     game_sound.play()
     points += 1 # Add points for each guess of a QDF, close but not concluded.
     game_points.append(points) # Add result to the scoresheet.
     print(f'Your score is: {points}. Scoresheet is: {game_points}')
     for count in range(2):
            print(fg.pink + helperTarget1, end="\r", flush=True), sleep(spd)
            print(fg.orange + helperTarget2, end="\r", flush=True), sleep(spd)
            print(fg.green + aimTarget, end="\r"+Fore.LIGHTYELLOW_EX, flush=True), sleep(spd)
            if keyboard.is_pressed("n") or keyboard.is_pressed("s"):
              break
```

- ▼ Source code in `QAI-LCode_QFLCC.py`

```
def helper_next():
             sleep(1)
             entry_add()
             level_show()
             subprocess.run('echo "{}- ///--- Helper Win Entry --- {} /// Level: {} \
Scored: {} at {}"'.format(entry_stage, helperTarget1, levels, points, now.strftime("%Y-%m-%d %H:%M:%S")),
shell=True, stdout=file_) # Date last I/O file entry.
             print("\n",f'Next...'+fg.lightgreen+' Level:', levels, f'{here}'+fg.orange)
```

- ▼ Source code in `QAI-LCode_QFLCC.py`

```
def level_show():
#-------------------------------------------
# Register level entry or update given the
# accumulated and/or lost energy score points.
#-------------------------------------------
     global points, levels
     game_sound = pygame.mixer.Sound('__snd__/gamelevel_complete.wav')
     game_sound.play()
     level_points.append(levels)
     if points < 10 and points > 0:
          levels=0
          print(f'\nLevel {levels} initiated! Your score is: {points}.')
          print(fg.lightgreen + guesser3, end="\r"+Fore.LIGHTYELLOW_EX, flush=True), sleep(spd)
          level_next()
     elif points >=10 and points < 20:
          levels=1
     elif points >=20 and points < 30:
          levels=2
     elif points >=30 and points < 40:
          levels=3
     elif points >=40 and points < 50:
          levels=4
     elif points >=50 and points < 60:
          levels=5
     elif points >=60 and points < 100:
          levels=6
     elif points >=100 and points < 1000:
          levels=7
     elif points >=1000:
          levels=8
          print(bg.green+fg.yellow+ f'\nLevel {levels} complete! You won all game rounds! \
Your score is: {points}. Scoresheet is: {game_points}')
          print(fg.lightgreen + guesser3, end="\r"+Fore.LIGHTYELLOW_EX, flush=True), sleep(spd)
          game_sound = pygame.mixer.Sound('__snd__/game_music2.wav')
          game_sound.play()
          level_next()
     print(Fore.LIGHTGREEN_EX + f'\nLevel {levels} engaged!')
     if levels >= 1 and levels < 8:
          level_dance()
     elif levels==8:
          game_win()
     elif points <= -10:
          game_over()  # Log event, either restart or end program by calling this function.
```

- ▼ Source code in `QAI-LCode_QFLCC.py`

```
def level_dance():
     for count in range(2):
            print(fg.orange + guesser2, end="\r", flush=True), sleep(spd)
            print(fg.lightgreen + guesser3, end="\r"+Fore.LIGHTYELLOW_EX, flush=True), sleep(spd)
            if keyboard.is_pressed("n"):
                  break
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def game_win():  # Conditions to play exit sound and animation, then exit game.
     for count in range(2):
            print(fg.lightgreen + classical1, end="\r", flush=True), sleep(spd)
            print(fg.orange + classical2, end="\r", flush=True), sleep(spd)
            print(fg.lightgreen + excited2, end="\r"+Fore.LIGHTYELLOW_EX, flush=True), sleep(spd)
            if participantMain == alice:
                 print(Back.YELLOW+ fg.lightgreen + crownAlice, end="\r"+Fore.LIGHTYELLOW_EX,
                       flush=True), sleep(spd)
            elif participantMain == bob:
                 print(Back.YELLOW +fg.lightgreen + crownBob, end="\r"+Fore.LIGHTYELLOW_EX,
                       flush=True), sleep(spd)
     # Play Windows exit sound.
     winsound.PlaySound("SystemExit", winsound.SND_ALIAS)
     sys.exit() # End of game as a winner, then HALT.
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def errorP(): # P Error state displayed with sound effect  after wrong user's input.
     game_sound = pygame.mixer.Sound('__snd__/errorP.wav')
     game_sound.play()
     game_sound = pygame.mixer.Sound('__snd__/gameno_state.wav')
     game_sound.play()
     for count in range(0, 2):
        print(bg.green+fg.yellow+ f'{errorP1}', end="\r", flush=True), sleep(spd)
        print(bg.green+fg.yellow+ f'{errorP2}', end="\r", flush=True), sleep(spd)
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def repeatP(): # P repeated displayed with sound effect after reentering a close or identical P by user.
     game_sound = pygame.mixer.Sound('__snd__/errorP.wav')
     game_sound.play()
     game_sound = pygame.mixer.Sound('__snd__/gameno_state.wav')
     game_sound.play()
     for count in range(0, 2):
        print(bg.green+fg.yellow+ f'{errorP1}', end="\r", flush=True), sleep(spd)
        print(bg.green+fg.yellow+ f'{errorP2}', end="\r", flush=True), sleep(spd)
        print(bg.green+fg.yellow+ f'{aimTarget}', end="\r", flush=True), sleep(spd)
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def game_over():
       game_sound = pygame.mixer.Sound('__snd__/game_countdown.wav')
       print(bg.cyan + fg.red+'QFLCC Game Over')
       game_sound.play()

       for j in range(9,-1,-1):
            print(bg.blue+fg.pink+'Enter \'r\' to restart this game, \'b\' to restart QFLCC, or else to end program...',
                  Back.YELLOW + fg.red+f"{j}"+Back.RESET+fg.blue, end="\r")
            sleep(1)
            if j == 6:
                 game_sound.play() # Partitioned sound replayed for the entire countdown sequence.
            if j == 0:
                uCommand = str(input(fg.lightcyan+"\n\r> "))
       if uCommand =='b' or keyboard.is_pressed("b"):
           subprocess.run(["python", "QAI-LCode_QFLCC.py"])  # Restart program.
           file_.close() # Close the log file recording events from the user's I/O terminal.
       elif keyboard.is_pressed("r") or uCommand =='r': # keyboard.wait("r"):
           level_start()
       else:
           print(bg.cyan+fg.red+'End of Program...'+ Back.RESET + Fore.RESET)
           subprocess.run('echo "///--- QFLCC Shell Log_file HALT --- /// on {}"'.format(entry_stage,
                          now.strftime("%Y-%m-%d %H:%M:%S")), shell=True, stdout=file_) # Date last I/O file entry.
           sys.exit() # Game Over and HALT.
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def level_next():
            entry_add()
            subprocess.run('echo "///--- Level Entry --- {}{} ///, Level: {} Scored: \
{} at {}"'.format(entry_stage, guesser3, levels, points, now.strftime("%Y-%m-%d %H:%M:%S")),
shell=True, stdout=file_) # Date last I/O file entry.
```

▼ Expand/Collapse All...

* Click Expand All to view all code blocks from QAI-LCode_QFLCC.py on this page.
* Click Collapse All to selectively view a code block from QAI-LCode_QFLCC.py on this page.

Expand All | Collapse All

▼ ≙ QAI/QFLCC P and correlation distance measure code:

QAI/QFCC code based on weighted P, correlation distance in QAI-LCode_QFLCC.py

```
2074  ##############################################################################################
2075  # QAI functions to compare correlated values based on User Input and weight them through
2076  # the elimination process of repeated successful hits of P values.
2077  ##############################################################################################
2078  PInMem = 0   # P value taken from User to store and update later.
2079  rep_lst0 = []
2080  rep_lst1 = []
2081  rep_lst2 = []
2082  w = 0.0 # ...for weighted guess by the user.
2083  DeltaC = 0.0
2084  DeltaP = 0.0
2085  rndDeltaC = 0.0
2086  dp = 0.0 # ...for a duplicate identified as a switch.
2087
2088  def reset_PPar():
2089       global DeltaC, PInMem, w, DeltaP
2090       DeltaC = 0.0
2091       DeltaP = 0.0
2092       PInMem = 0.0
2093       w=0.0
2094
2095  def uIn_Repeat():
2096  ###############################################
2097  # Function for P and weighted P calculations.
2098  ###############################################
2099       global PInMem, w, dp, DeltaP
2100       dp=0
2101
2102       #for PInMem in rep_lst0:
2103       rep_lst0.append(round(PInMem, 2))  # Append the registered input with two decimal points accuracy.
2104       print("List of tried P\'s: "+ Fore.LIGHTMAGENTA_EX, rep_lst0, Fore.LIGHTGREEN_EX)
2105
2106       rep_lst1.append(round(w, 3))
2107       print("List of assigned weights: "+ Fore.LIGHTMAGENTA_EX, rep_lst1, Fore.LIGHTGREEN_EX)
2108
2109       rep_lst2.append(round(DeltaP, 2))
2110       print("List of matched ΔP\'s: "+ Fore.LIGHTMAGENTA_EX, rep_lst2, Fore.LIGHTGREEN_EX)
2111
2112       # Create a set from the list.
2113       sub_lst0 = set(rep_lst0)
2114       print('Subset of tried P\'s:', sub_lst0)
2115
2116       # Compare the length and print if the list contains duplicates.
2117       dup0 = {x for x in rep_lst0 if rep_lst0.count(x) > 1}
2118       print('Duplicated P tries:'+fg.red, dup0, ''+fg.lightgreen)
2119
2120       # Create a set from the list.
2121       sub_lst2 = set(rep_lst2)
2122       print('Subset of calculated ΔP\'s:',sub_lst2)
2123
2124       dup2 = {y for y in rep_lst2 if rep_lst2.count(y) > 1}
2125       print('Duplicates of matched P\'s:'+fg.lightcyan, dup2, ''+fg.lightgreen)
2126
2127       if (round(PInMem, 2) in dup0): # or (round(DeltaP, 2) in dup2):
2128            print(f'Duplicates {dup0} found in your tried P\'s list! Try again...')
2129            dp=1
2130            repeatP()
2131            print('\n')
2132
2133       while dp==0:
2134            if round(PInMem, 2) in rep_lst0 and w>=9 and dp==0:
2135                 print(f'P value {PInMem} guessed a weak classical P outcome and correlated with \
2136  max(ΔP)= min(P) of qubit dataset. Your successful hit weight was: {w}')
2137                 break
2138            elif round(PInMem, 2) in rep_lst0 and (w<9 and w> 2) and dp==0:
2139                 print(f'P value {PInMem} guessed a moderate classical P outcome and correlated with \
2140  some <ΔP>=<P> of qubit dataset. Your successful hit weight was: {w}')
2141                 break
2142            elif round(PInMem, 2) in rep_lst0 and (w< 1.9 and w > 1.1) and dp==0:
2143                 print(f'P value {PInMem} guessed a classical P outcome and correlated with \
2144  min(ΔP)= max(P) of qubit dataset. Your successful hit weight was: {w}')
2145                 break
2146            elif round(PInMem, 2) in rep_lst0 and (w==0) and dp==0:
2147                 w=float('inf') # Define a positive infinite integer assigned to the weight variable w.
2148                 print(f'P value {PInMem} guessed an undefined P outcome of the qubit dataset. Your \
2149  successful hit weight was: [0, {w}]')
2150                 break
2151            elif round(PInMem, 2) in rep_lst0 and (w<=1.1 and w>0) and dp==0:
2152                 print(f'P value {PInMem} guessed a strong classical P outcome and correlated with \
2153  max(ΔP)=min(P) of qubit dataset. Your successful hit weight was: (0, {w}]')
2154                 break
2155            elif round(PInMem, 2) in rep_lst0 and (w<=2.2 and w>=1.9) and dp==0:
2156                 print(f'P value {PInMem} guessed a superposition P outcome and correlated with a \
2157  ΔP=1/2 uncertainty of the qubit dataset. Your successful hit weight was: {w}')
2158                 break
2159            elif dp==1:
2160                 reset_PPar()
2161                 continue
2162            else:
2163                 print(f'P value not guessed and uncorrelated. Your successful hit weight was: {w}')
2164                 reset_PPar()
2165                 break
```

▼ Expand/Collapse All...

* Click Expand All to view all code blocks from QAI-LCode_QFLCC.py on this page.
* Click Collapse All to selectively view a code block from QAI-LCode_QFLCC.py on this page.

Expand All | Collapse All

▼ ≙ QDF game prompt and help options:

Calling help, prompt and sound volume setting functions during the game. Function() list from top to bottom as ordered:

- ▼ ≙ QDF game sound volume settings::

QDF game sound volume settings for basic and expert modes: source code in QAI-LCode_QFLCC.py or QDF-game-gui.py (under construction)

```
#### Volume Settings ####
import pyautogui as pvol

def set_game_vol(new_volume):
############################################################
# The volume function sets volume according to your OS.
# This option is available as 'v' or 'volume' during the game.
############################################################
    pvol.press('volumedown', presses = 50) # Sets volume to zero.
    time.sleep(0.5) # Using time.sleep to space the presses.
    x = math.floor(new_volume / 2) # Setting the amount of presses required.
    pvol.press('volumeup', presses = x) # Setting the volume.
    winsound.PlaySound("SystemQuestion", winsound.SND_ALIAS) # Sound test.
    """- End of basic mode volume settings."""

############################################################
# Packages and variables for sound volume change from within CLI
# during the game (expert mode with decimal range approximation).
############################################################
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume

devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
volume = cast(interface, POINTER(IAudioEndpointVolume))
### End of expert mode volume settings -###
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def set_game_vol(new_volume):
############################################################
# The volume function sets volume according to your OS.
# This option is available as 'v' or 'volume' during the game.
############################################################
    pvol.press('volumedown', presses = 50) # Sets volume to zero.
    time.sleep(0.5) # Using time.sleep to space the presses.
    x = math.floor(new_volume / 2) # Setting the amount of presses required.
    pvol.press('volumeup', presses = x) # Setting the volume.
    winsound.PlaySound("SystemQuestion", winsound.SND_ALIAS) # Sound test.
    """- End of basic mode volume settings."""
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def stayIn():
#########################################################################
# To stay in prompting environment and respond to a repeated question.
#########################################################################
 while True:
     promptIn = str(input(bg.green + fg.yellow +'Do you want to continue in prompt mode to input command? [y/n]'+ Back.RESET))
     if (promptIn == 'y' or promptIn == 'yes') and pFlag == True:
           print(bg.green + fg.yellow +
                 "Input the relevant command according to \'h\' or \'help\', \'cv\', \'v\' or \'volume\' for sound volume change..."
                 + Back.RESET)
           promptGame() # Restart the prompt function.
           break
     elif (promptIn == 'y' or promptIn == 'yes') and pFlag == False:
           print(bg.green + fg.yellow +
                 "Input command according to \'help\'. To view the list of commands input \'h\' or \'help\':"
                 + Back.RESET)
           promptGame() # Restart the prompt function.
           break
     elif (promptIn == 'n' or promptIn == 'no') and (pFlag == True or pFlag == False):
           break  # Skip this step and continue the game based on n or any other relevant command.
```

- ▼ Source code in QAI-LCode_QFLCC.py

```
def pass_code():
#############################################################################
# Enters Safe Mode via passcode to cheat and change dataset feed to the game.
#############################################################################
 global dir_flag
 # Data here lists hardcoded data. This array can be customized and elements replaced
 # via index value replacement by a password txt file once loaded.
 y = random.randint(0, 9999) # Can crack and override this when value is stored and accessed realtime
                             # not after view.
 y = str(y)
 pass_file = "pass-code.txt"

 x = random.randint(0, 1)
 x= str(x)

 data = {
  "passcode": ['$ Alice & Bob cheat $', 'qdf cheat sheet 2121', 'dataset 0110 cheat',
               'qdf dataset 00,01,10,11', x, len(y[:-1])*'#'+y[-1:], ''],
  f"{Back.LIGHTGREEN_EX}qualified{Back.RESET}": [f"{Fore.LIGHTGREEN_EX}{True}{Fore.LIGHTGREEN_EX}",
                f"{Fore.LIGHTGREEN_EX}{True}{Fore.LIGHTGREEN_EX}",
                f"{Fore.LIGHTGREEN_EX}{True}{Fore.LIGHTGREEN_EX}",
                f"{Fore.LIGHTGREEN_EX}{True}{Fore.LIGHTGREEN_EX}",
                f"{Fore.LIGHTRED_EX}binary to pass [{False}]{Fore.LIGHTGREEN_EX}",
                f"{Fore.LIGHTYELLOW_EX}one-time pass [{True}]{Fore.LIGHTGREEN_EX}",
                f"{Fore.LIGHTRED_EX}else to pass [{False}]{Fore.LIGHTCYAN_EX}"]
 }
 df = pd.DataFrame(data, index=['pass_1', 'pass_2', 'pass_3', 'pass_4', 'pass_5', 'pass_6', 'pass_7'])
 index_ = ['pass_1', 'pass_2', 'pass_3', 'pass_4', 'pass_5', 'pass_6', 'pass_7'] # Create the index.
 df.index = index_ # Set the index.

 z=['','','','','','','']
 for i in range (0,6):
      z[i] = df.loc[f'pass_{i+1}','passcode']
      #print(z[i])  # Any other z[i], for a combo code restrictions and switches can updated to the growing passcode list below.

 user_input = input(f"{fg.lightgreen}>>{fg.lightcyan} This command is supported only in the QFLCA's Safe Mode (SM:>>)\
 environment! \n-- Enter [Passcode + Enter] to proceed. If 'qualified' enter 'n' when prompted.{Fore.LIGHTRED_EX}* \
\n*- Some passcodes{fg.lightcyan} can unlock lower and/or higher level codes to access different game levels and datasets. \
\n-- Here is your chance to use a one-time passcode, if you know where it is locally?! Press Enter... ")

 with open(pass_file, 'w+') as passf_to_write:
              # Write information about the original image sequence and final image frame.
              passf_to_write.write(y)
 passf_to_write.close()
 game_sound = pygame.mixer.Sound('__snd__/game_countdown.wav')
 game_sound.play()
 hline = "=========================================================================================================="
 for cnt_down in range(9, -1, -1): # 10 seconds left to see this code from its file location.
    print(f"{fg.lightgreen}>>{fg.lightcyan} One-time passcode is unmasked in the\
 background for 10 seconds...{Back.YELLOW + fg.red}{cnt_down}{Back.RESET+ fg.lightcyan}", end = '\r')
    sleep(1)
    if cnt_down == 6:
         game_sound.play() # Partitioned sound replayed for the entire countdown sequence.
    elif cnt_down == 0:
         #print(end=' ')
         print(f"{hline}")

 with open(pass_file, 'w+') as passf_to_write:
     passf_to_write.write(len(y[:-1])*"#"+y[-1:])
     passf_to_write.close()
 with open(pass_file) as passf_to_read:
      print(f"{fg.lightgreen}>>{fg.lightcyan} One-time passcode is now masked... {Fore.LIGHTRED_EX}"
           + passf_to_read.read() + Fore.LIGHTCYAN_EX), sleep(2)
 passf_to_read.close()

 user_input = input(f"{fg.lightgreen}>>{fg.lightcyan} Enter your passcode, or press Enter to skip back to \
regular prompt ({fg.lightgreen}>{fg.lightcyan}): {Fore.LIGHTRED_EX}")

 newdf = df.mask(df["passcode"] >= user_input[2:]) # Mask code until revealed with the
                                                   # right one(s) entered by user
                                                   # (the first 2 chars of input ignored).
 if (user_input == z[0] or user_input == z[1] or user_input == z[2] or user_input == z[3] or user_input == z[4]
     or user_input == z[5]):
        dir_flag = 1  # Directory flag is set to 1 for directory access in safe mode.
        qdf_bits_flag[1] = True # Set this flag to 1 or reveal the qdf bit values from any selected dataset.
        qdf_bits_flagset() # Call this function to show the qdf bit values of the employed dataset.
        update_btn4_text() # Make sure the form button4 has qdf bit values revealed by calling this function.
        print(f"{bg.cyan + user_input} is qualified as [True]! \nThe following list shows if there are any more \
passcodes you can use later: {Fore.LIGHTGREEN_EX +  Back.RESET}")
        print(newdf)
        prompt()
        safeMode_dir()
        PAnalysis_model() # Run QDF circuit simulation in case of successfully accessing the cheat sheet
                          # for a new dataset feed. Dir flag will be reset back to 0 in the process.
 if (user_input == "n" or user_input == z[6] or user_input == ' ' or user_input == "next"):
        dir_flag = 0
        prompt()
 if user_input == y:  # Reveal all passcodes if the random number is caught/guessed and entered by the user.
        dir_flag = 1
        qdf_bits_flag[1] = True # Set this flag to 1 or reveal the qdf bit values from any selected dataset.
        qdf_bits_flagset() # Call this function to show the qdf bit values of the employed dataset.
        update_btn4_text() # Make sure the form button4 has qdf bit values revealed by calling this function.
```

```
2307            print(f"{bg.cyan + user_input} You unlocked all the passcodes for future use! This is your chance to \
2308    write them down: {Fore.LIGHTGREEN_EX +  Back.RESET} \n", df)
2309            prompt()
2310            safeMode_dir()
2311            PAnalysis_model()
2312            """- End of passcode conditions to change the game's dataset feed."""
```

- ▼ 🔽Source code in `QAI-LCode_QFLCC.py`

```
2314    def promptGame():
2315    ########################
2316    # QDF Game Help function.
2317    ########################
2318        global spd, pFlag, site_name   # A global variable for game speed as the QDF game steps pace up or slow down.
2319                  # pflag is a boolean global variable to raise this flag for the prompt state for a specific prompt.
2320        while True:
2321            print(fg.lightgreen + "\r< ", end=""), sleep(spd)
2322            print("\r> ", end=""+fg.orange), sleep(spd)
2323            try:
2324                 promptIn = str(input()) # Respond to the user's choice or command.
2325                 if promptIn == 'n' or promptIn == 'next' or promptIn == 'no':
2326                    break
2327                 elif promptIn == 's' or promptIn == 'speed':
2328                      takeUIn_speed = float(input(bg.orange+'Change game speed −=≡Σ((( つ◕_◕)つ ... Choose from this range:'
2329                                                  + Back.RESET+ fg.yellow  +' [fastest = 0.1, slowest = 2]: '))
2330                      spd = takeUIn_speed
2331                      if spd >2.0 or spd < 0.1:
2332                           print(fg.red +"Wrong value entered! Type in the same \'s\' command or other for an input...")
2333                           promptGame()
2334                      else:
2335                          spd = takeUIn_speed
2336                      break
2337                 elif promptIn == 'v' or promptIn == 'volume':
2338                      vol = float(input(bg.orange+'Input sound volume −=≡Σ(((° ͜ʖ °)🔊  between'+ Back.RESET+ fg.yellow
2339                                        + '[🔇 = 0 for muted, and 🔊 = 100 for loudest]: '))
2340                      if vol >= 0 and vol <= 100:
2341                         set_game_vol(vol)
2342                         pFlag = True # Raise this flag for message containing a volume prompt message.
2343                         stayIn()  # To remain to make further volume change or input other command.
2344                         break
2345                      elif vol < 0:
2346                           set_game_vol(vol)
2347                           print(fg.red +"Out of range or wrong value entered! Readjusted volume to muted or 0!")
2348                           promptGame()  # Restart prompt.
2349                           break
2350                      elif vol > 100:
2351                           set_game_vol(vol)
2352                           print(fg.red +"Out of range or wrong value entered! Readjusted volume to maximum or 100!")
2353                           promptGame() # Restart prompt.
2354                           break
2355                 elif promptIn == 'cv' or promptIn == 'cliv':
2356                      pFlag = True # Raise this flag for message containing a volume prompt message.
2357                      takeUIn_volume = float(input(bg.orange+'Change game volume −=≡Σ(((° ͜ʖ °)🔊 ... Choose from this range:'
2358                                                   + Back.RESET+ fg.yellow +
2359                                                   ' [{quietist 🔇 ⇄ 🔉 = [0 to 6], quiet 🔉 = 6.1}, loudest 🔊 = 100]: ')) # Input value
2360                                                                                                                  # should be between >= 0 and 100.
2361                      cvol = takeUIn_volume
2362                      if cvol > 100 or cvol < 0:
2363                           print(fg.red +"Out of range or wrong value entered! Type in the same \'cv\' command or other for an input...")
2364                           promptGame() # Restart prompt.
2365                           break
2366                      elif cvol >= 20 and cvol <= 100:
2367                          cvol = takeUIn_volume
2368                          min_vol = ((11*cvol - 1100)/40)  # My lower conversion formula for volume range of [-29, 0]
2369                                                           # is e.g.: 100-(100*(-5-0)/((0-55)/2)) as simplified.
2370                          print ("Volume converted and adjusted from the upper desktop's endpoint volume range of [-29, 0] =" , "%.4f" % min_vol)
2371                          #-------------------------------------------
2372                          # Control volume proximation:
2373                          #-------------------------------------------
2374                          #volume.SetMasterVolumeLevel(0, None) #max
2375                          #volume.SetMasterVolumeLevel(-5.0, None) #72%
2376                          volume.SetMasterVolumeLevel(min_vol, None) #alternative 72% from 80 as user input.
2377                          #volume.SetMasterVolumeLevel(-10.0, None) #51%
2378                          #volume.SetMasterVolumeLevel(-20.0, None) #26% alternative is 27 from user input (very close).
2379                          #volume.SetMasterVolumeLevel(-30.0, None) #13%
2380                          #volume.SetMasterVolumeLevel(-55.0, None) #1%
2381                          #-------------------------------------------
2382                          winsound.PlaySound("SystemQuestion", winsound.SND_ALIAS)
2383                          stayIn() # To remain to make further volume change.
2384                          break
2385                      elif cvol > 6  and cvol < 20:
2386                          cvol = takeUIn_volume
2387                          min_vol = -4000/(11*cvol)  # My lower conversion formula for volume range of [-40, -30] is:
2388                                                     # 100*(100/(-55-0)/((0-55)/2))=6.61 as simplified.
2389                          print ("Volume converted and adjusted from the lower desktop's endpoint volume range of [-41, -30] ="
2390                                 , "%.4f" % min_vol)
2391                          volume.SetMasterVolumeLevel(min_vol, None)
2392                          winsound.PlaySound("SystemQuestion", winsound.SND_ALIAS)
2393                          stayIn() # To remain to make further volume change.
2394                          break
2395                      elif cvol >= 0 and cvol <= 6: # This condition ranges the module's close to muted state or 0% volume.
2396                          cvol = 6  # The minimum limit has reached for the lowest bound conversion for the customized volume.
2397                          min_vol = -4000/(11*cvol)  # My lower conversion formula for volume range of
2398                                                     # [-60.606060..., -40) as simplified.
2399                          print ("Volume converted and adjusted from the upper desktop's endpoint volume range of [-60.6, -40) ="
2400                                 , "%.4f" % min_vol)
2401                          volume.SetMasterVolumeLevel(min_vol, None)
2402                          winsound.PlaySound("SystemQuestion", winsound.SND_ALIAS)
2403                          stayIn() # To remain to make further volume change.
2404                          break
2405                 elif promptIn == 'b' or promptIn == 'begin':
```

```
                    subprocess.run(["python", "QAI-LCode_QFLCC.py"])  # Restart program to analyze and validate a new dataset.
                    break
               elif promptIn == 'f' or promptIn == 'form':
                    print('Revisit dataset(s) form load & play [Temporal State]...☝ (° ͜ʖ °)👉🗐 ⟵ 🎟 : '
                          + bg.red + fg.yellow +'A F F I R M A T I V E !')
                    game_sound = pygame.mixer.Sound('__snd__/gameaffirmative_state.wav')
                    game_sound.play()
                    root.deiconify() # Unhide this form and show its options to resume game, reload or choose dataset(s).
                    pFlag = False # Lower this flag for a message containing a regular prompt message
                    stayIn() # To remain in prompt mode to input command.
                    break
               elif promptIn == 'r' or promptIn == 'restart':
                    print('Restarting program...☝(° ͜ʖ °)☝')
                    open_new_win() # Restart program.
                    break
               elif promptIn == 'website' or promptIn == 'site' or promptIn == 'about' or promptIn == 'web':
                    site_name = "game-site" # This html file is stored in the same site folder to read.
                    site_doc() # Load website.
               elif (promptIn == 'dir' or promptIn == 'sm dir') and (dir_flag == 0):
                    pass_code()  # Call this function to enter passcode to cheat and feed a new dataset to the game.
                                 # Dir flag will be set to 1 in the process.
                    continue
               elif promptIn == 'h' or promptIn == 'help':
                    userGame_Help()  # Show help.
                    continue
               elif promptIn == 'e' or promptIn == 'exit':
                    # Play Windows exit sound.
                    game_sound = pygame.mixer.Sound('__snd__/goodbye1.wav')
                    game_sound.play()
                    print('Exiting game...✋ (° ͜ʖ °)✋  Goodbye!'), sleep(3)
                    main_exit()  # Terminate game program.
               else:
                  print('Input command or response. For help, enter \'h\'... ')
                  continue
          except ValueError:
               print("Invalid input. Please enter a response.")
```

- ▼ ▼ Source code in `QAI-LCode_QFLCC.py`

```
def userGame_Help():
#########################
# QDF Game Help function.
#########################
     while True:
          print(f'\033[0;31m\033[1m\x1B[1;4mGame Input Tips:\x1B[0m\033[0m \033[0;36m\n-Enter \'n\' key for the next message, result, \
output or next input. \n-Enter \'h\' to display these tips. \n-Enter a role number when asked for a role as Alice {alice} or Bob \
{bob} via Eve {eve} or the Audience {audience}. \n-Enter a value between 0 and 1 inclusive when asked for a \
P value. \n-Enter \'r\' to restart game. \n-Enter \'b\' to analyze and validate a new dataset. \n-Enter \'s\' \
or \'speed\' to change the speed of game steps. \n-Enter \'v\' or \'volume\' for sound volume change during the game. \
\n-Enter \'site\' or \'web\' to view \'about\' the game \'website\'. \
\n-Enter \'cv\' for sound volume change within CLI. \n-Enter \'f\' or \'form\' to reload the QDF \
game form (resume game by clicking its \'START\' button). \
\n-Enter \'dir\' or \'sm dir\' for a new dataset analysis & simulation to feed this game.*\
\n*-This command requires a passcode to see dataset results in a Safe Mode (SM:>>) environment!\
     \n-Enter \'e\' to exit the program.')
          break
```

- ▼ ▼ ℹ ≙ QDF game conditions for a QFLCC...

QDF game QFLCC conditions: source code in QAI-LCode_QFLCC.py or QDF-game-gui.py (under construction)

```
########################################
# QFLCC Game Conditions...
########################################
p = 1 # classical state of the probability
k = 10 # math.inf #float('inf')

# Set an initial condition.
game_active = True
# Set up the while loop.

def level_start():
     entry_add()  # Date last I/O file entry.

level_start()
level_show() # Register Level no. Levels engage and get completed given the lost
             # vs. accumulated points.
```

▼ ▼ ℹ Expand/Collapse All...

* Click ▼Expand All to view all code blocks from QAI-LCode_QFLCC.py on this page.
* Click ▶Collapse All to selectively view a code block from QAI-LCode_QFLCC.py on this page.

▼Expand All ▶Collapse All

▼ ▼ ℹ ≙ Main QDF game code to display P results for game participants and QFLCC...

```
Main QDF game code: game_active code in QAI-LCode_QFLCC.py or QDF-game-gui.py (under construction)
```

```
##############################################
# The main game code block for taking in P's from
# the user and dataset(s), and determine game's
# participant role, wins, losses and levels.
##############################################
while game_active:
    dp=0
    winsound.PlaySound("SystemExclamation", winsound.SND_ALIAS)
    try:
     takeUIn = int(input(fg.orange+f'In this game, are you Bob {bob} the guest, or Alice {alice} \
the host to win a/the prize (targeted energy state) {prize}?\nChoose 1 for \
Bob, 2 for Alice: '))
     sys.stdout.flush() # Clear line.
    except ValueError:
         game_sound = pygame.mixer.Sound('__snd__/gameno_state.wav')
         game_sound.play()
         print(fg.lightred + f"❌ NO! Wrong Input! Input must be 1 or 2. {errorP1} Try again..."+fg.lightgreen)
         continue
    try:
     takeUIn2 = int(input(f'For your game participant {takeUIn}, will Eve {eve} join by quantum means \
to secretly share information about the prize {prize}?\nChoose 3 to have Eve \
spying, 4 for Audience {audience} to cheer/suggest and raise/lower \
participant {takeUIn}\'s energy state: '))
    except ValueError:
         game_sound = pygame.mixer.Sound('__snd__/gameno_state.wav')
         game_sound.play()
         print(fg.lightred + f"❌ NO! Wrong Input! Input must be 3 or 4. {errorP2} Try again..."+fg.lightgreen)
         continue
    try:
     takePIn = float(input(f'Enter a P value for participant #{takeUIn}, \
{participantMain}: '))  # The var input will be a P value for the current game participant.
    except:
         game_sound = pygame.mixer.Sound('__snd__/gameno_state.wav')
         game_sound.play()
         print(fg.lightred + f"❌ NO! Wrong Input! Input a floating-point value between 0 and 1. {errorP1} Try again..."
               +fg.lightgreen)
         continue

    promptGame()
    participants()
    PInMem = takePIn

    uIn_Repeat() # Register P value and compare for a weight assigned to the closest correlation vs farthest
                 # for a strong prediction between two datasets. Not just comparingP values within this dataset
                 # for validation purposes, according to ref. [1].
    if ((takePIn < abs(p) and takePIn >0.5) or (takePIn < abs(p)/2 and
                                                takePIn > 0)) and dp==0: # and takeO == 'p game':
         print(bg.green+fg.yellow+'Hello, Quantum World!'+Back.RESET)
         ##################################################
         # Now test correlation with prediction results
         ##################################################
         if (takePIn < float(df3Mem) and takePIn>0.5):
              DeltaP = float(df3Mem)-takePIn
              if DeltaP < 0.5:
                   rndDeltaC = round(1-DeltaP, 2)
                   DeltaC = 1-DeltaP
                   w= round((1/(DeltaC)),3)  # Weighted value of the correlation distance.
                                             # The closer to infinity the greater the weight and
                                             # inaccuracy of player's guess/hit on P value.
                   print(f'Correlation to strong prediction result for participant #{takeUIn}, \
{participantMain}, is high to win:'+fg.yellow, DeltaC, Fore.RESET)
                   if (takeUIn == 1 and takeUIn2 == 3 and (DeltaC >0.5 or DeltaC ==1)):
                        print(Back.LIGHTYELLOW_EX + 'Alice loses to Bob. Bob wins the prize with Eve\'s help!'
                              +Back.RESET+fg.green+' YOU WIN!')
                        print(fg.green+'These doubles won the prize:'+fg.pink, qdf, fg.green+ ', QDF P match is:'
                              +fg.pink, rndDeltaC)
                        winner_show()
                        winner_next()
                   elif (takeUIn == 1 and takeUIn2 == 4 and (DeltaC > 0.5 or DeltaC ==1)):
                        print(Back.LIGHTYELLOW_EX +
                              'Alice loses to Bob. Bob wins the prize as the Audience cheer without Eve\'s help!'
                              + Back.RESET+fg.green+' YOU WIN!'+Fore.RESET)
                        print(fg.green+'These doubles won the prize:'+fg.pink, qdf, fg.green+ ', QDF P match is:'
                              +fg.pink, rndDeltaC)
                        winner_show()
                        winner_next()
                   elif (takeUIn == 2 and takeUIn2 == 3 and (DeltaC >0.5 or DeltaC ==1)):
                        print(Back.LIGHTYELLOW_EX + 'Bob loses to Alice. Alice wins to keep the prize with Eve\'s help!'
                              + Back.RESET + fg.green +' YOU WIN!'+ Fore.RESET)
                        print(fg.green+'These doubles won the prize:'+fg.pink, qdf, fg.green+ ', QDF P match is:'
                              +fg.pink, rndDeltaC)
                        winner_show()
                        winner_next()
                   elif (takeUIn == 2 and takeUIn2 == 4 and (DeltaC >0.5 or DeltaC ==1)):
                        print(Back.LIGHTYELLOW_EX + 'Bob loses to Alice. Alice wins to keep the prize as the Audience \
cheer without Eve\'s help!'+ Back.RESET + fg.green +' YOU WIN!'+ Fore.RESET)
                        print(fg.green+'These doubles won the prize:'+fg.pink, qdf, fg.green+ ', QDF P match is:'
                              +fg.pink, rndDeltaC)
                        winner_show()
                        winner_next()
         if (takePIn < float(df3Mem) and takePIn>0) and dp==0:
              DeltaP = float(df3Mem)-takePIn
              if DeltaP > 0.2 and dfcompMem <= 0.5: # This condition could intersect with the winning condition
                                                    # if results are close to DeltaP relative to df3Mem. This
                                                    # creates a win hit and a loss hit as two consecutive rounds
                                                    # (event outcomes) of the game.
                   rndDeltaC = round(1-DeltaP, 2)
                   DeltaC = 1-DeltaP
                   w= round((1/(DeltaC)),3)#-DeltaC # Enable this for conditional DeltaC calculation.
                   print(f'Correlation to strong prediction result for participant #{takeUIn}, {participantMain}, \
is low to win:'+fg.yellow, rndDeltaC, Fore.RESET)
                   if (takeUIn == 1 and takeUIn2 == 3 and (DeltaC >0 and DeltaP>=(1/3))):
                        print(Back.LIGHTYELLOW_EX +
                              'Bob loses to Alice despite Eve\'s help. Alice wins to keep the prize!'
                              + Back.RESET+fg.red+f' YOU {participantMain} via {participantMid} LOSE!'+Fore.RESET)
                        print(fg.orange+'These doubles:'+fg.lightred, qdf, fg.orange+'lost the prize to'
```

```
                                +fg.lightred, bitpair, fg.orange+', QDF P match is:'+fg.lightred, rndDeltaC)
                            loser_show()
                            loser_next()
                    elif (takeUIn == 1 and takeUIn2 == 4 and
                          (DeltaC >0 and DeltaP>=(1/3))): # This operation got registered with values like 02, .03 etc.
                          print(Back.LIGHTYELLOW_EX +
                                'Bob loses to Alice despite the Audience cheer without Eve\'s help. \
Alice wins to keep the prize!'+Back.RESET +fg.red +f' YOU {participantMain} via {participantMid} LOSE!'+Fore.RESET)
                          print(fg.green+'These doubles lost the prize:'+fg.lightred, qdf, fg.orange
                                +'lost the prize to'+fg.lightred, bitpair, fg.orange+', QDF P match is:'+fg.pink, rndDeltaC)
                          loser_show()
                          loser_next()
                    elif (takeUIn == 2 and takeUIn2 == 3 and (DeltaC >0 and DeltaP>=(1/3))):
                          print(Back.LIGHTYELLOW_EX + 'Bob loses to Alice despite Eve\'s help. Alice wins to keep the prize!'
                                + Back.RESET+fg.red+f' YOU {participantMain} via {participantMid} LOSE!'+Fore.RESET)
                          print(fg.green+'These doubles lost the prize:'+fg.lightred, qdf, fg.orange+'lost the prize to'
                                +fg.lightred, bitpair, fg.orange+', QDF P match is:'+fg.pink, rndDeltaC)
                          loser_show()
                          loser_next()
                    elif (takeUIn == 2 and takeUIn2 == 4 and (DeltaC >0 and DeltaP>=(1/3))):
                          print(Back.LIGHTYELLOW_EX +
                                'Bob loses to Alice despite the Audience cheer without Eve\'s help. \
Alice wins to keep the prize!'+Back.RESET+fg.red+f' YOU {participantMain} via {participantMid} LOSE!'+Fore.RESET)
                          print(fg.green+'These doubles lost the prize:'+fg.lightred, qdf,
                                fg.orange+'lost the prize to'+fg.lightred, bitpair, fg.orange+', QDF P match is:'
                                +fg.pink, rndDeltaC)
                          loser_show()
                          loser_next()
           else:
                print(bg.orange + fg.green+f'Continue guessing the P as participant #{takeUIn2}, {participantMid}, \
helps you {participantMain}...'+Back.RESET + fg.orange)
                helper_show()  # Grant only 1 point as a helper to the progression of user guesses to come.
                helper_next()
    if (takePIn == abs(p) or takePIn == 0) and dp==0:
        print(bg.blue+'Hello, Classical World!\n', bg.orange + fg.green
              +'Now guess what the QDF outcome would be relative to your next classical guess?'
              +Back.RESET + fg.orange)
    elif (takePIn < float(df3Mem) and (takePIn >= 0.45 and dfcompMem <=0.5)) and dp==0:
        print(bg.blue +'Hello, Quantum-classical World!'+Back.RESET + fg.orange)
        print(f'{fg.purple}Welcome {participantMain} to the world of superposition or entanglement!')
        DeltaP = float(df3Mem)-takePIn
        rndDeltaC = round(1-DeltaP, 2)
        DeltaC = 1-DeltaP
        w= round((1/(DeltaC)),3) # -DeltaC), # Include this in case of further DeltaC calculation.
        print('These doubles \"are all in\" for the prize:' +fg.pink, qdf, fg.green+'QDF P match is:'
              +fg.pink, rndDeltaC)
        print(bg.orange + fg.green+'Now guess what the next QDF outcome would be?'
              +Back.RESET + fg.orange)
        guesser_show()
        guesser_next()
    elif (((takePIn < float(df3Mem) and takePIn >= 0.5) and (dfcompMem <= 0.5 and dfcompMem > 0.48))) and dp==0:
            DeltaP = float(df3Mem)-takePIn
            rndDeltaC = round(1-DeltaP, 2)
            DeltaC = 1-DeltaP
            w = round((1/(DeltaC)),3) # -DeltaC), include in case of further calculation.
            print(bg.blue+'Hello, Quantum-classical World!'+Back.RESET + fg.orange)
            print(f'{fg.purple}Welcome {participantMain} to the world of superposition or entanglement!')
            print(f'These doubles \"are all in\" for the prize: {qdf}', f'QDF P match is: {rndDeltaC}')
            print(bg.orange + fg.green+'Now guess what the next QDF outcome would be?'+Back.RESET + fg.orange)
            dual_show()
            dual_next()
            continue
    if takePIn > 1 or takePIn < 0:
        print('Wrong P value!')
        errorP()
        dp=1
        print('\n')
        continue
#########--END OF PROGRAM--#########
```

▼ Expand/Collapse All...

* Click Expand All to view all code blocks from QAI-LCode_QFLCC.py on this page.
* Click Collapse All to selectively view a code block from QAI-LCode_QFLCC.py on this page.

Expand All | Collapse All