| 1  | Tutorials                      | 3  |
| 2  | Examples                       | 13 |
| 3  | Documentation                  | 19 |
| 4  | Installation                   | 35 |
| 5  | Questions? Help? Movie...      | 37 |
| 6  | Index                          | 39 |
Neuropsydia is a Python module that allows to easily create experiments, tasks or questionnaires.

Contents:
This courses were crafted by psychologists, neuropsychologists and neuroscientists for psychologists, neuropsychologists and neuroscientists. As such, it is a straightforward introduction to Neuropsydia for Python with a special focus on how to actually do something with it. It is not a programming course on Python, nor a course on programming per se.

Contents:

1. Getting Started

Contact

For remarks, complaints, suggestions or anything else, please create an issue.

Installation

Installation steps can be found here.

Hands on!

Well, first of all, welcome and thank you for trying the Python version of Neuropsydia for research. I’m sure you will like it as, while still in development, it already has some powerful functions that will help you create your tasks, experiments, and more.

But enough talking! After the installation, open your python code editor (e.g., spyder, available within the WinPython bundle). Here, you will write functions that will almost magically come to life once the program is launched.

In spyder, one pane is the CONSOLE, where Python actually lives. It’s its interface with us humans. If you type something in there, it will swallow it, do something with it, and then, maybe, return something. Also, it’s great for calculus. For example, try typing \( 5^2 \) and hit Enter.
Pretty cool, huh?

But for now, let’s focus on some simple things, such as how to load, start and close neuropsydia.

Import Neuropsydia

Neuropsydia for research is a module which, like any other module, must be loaded in order to be used. In Python, we load modules by running:

```python
import themoduleiwanttoload
```

Then, we can use its functions:

```python
themoduleiwanttoload.function1()
themoduleiwanttoload.function2()
```

However, as you can see it, it’s pretty annoying to write the full name of the module each time we use one of its function. Fortunately for us, Python allows us to load a module under another name (or alias), for example the letter “x”.

```python
import themoduleiwanttoload as x
x.function1()
x.function2()
```

Better. That’s what we are going to do with neuropsydia, loading it under the name “n”.

However, unlike many other packages, Neuropsydia CANNOT be loaded without two of its function, :code:`start()` and :code:`close()`.

Start and Close

So, when importing `neuropsydia`, what happens is basically that it needs to initialize several things before being ready to use. And those things are initialized with the :code:`start()` function. Moreover, adding the :code:`close()` function at the end will ensure a clean ending.

Therefore, every experiments, after loading the neuropsydia module, will begin by the :code:`start()` function and end with the :code:`close()` function.

Let’s try it. Write in the editor the following lines of code:

```python
import neuropsydia as n
n.start()
n.close()
```

Execute your code

The rest of your code will go between the :code:`start()` and the :code:`close()` functions. But these 3 lines are a minimal working program, so let’s try it.

To execute an entire python program, always open a new python console (in spyder, go to console, then click on open a new python console, then press F5, - or the green arrow).

Do it.

...  
Tadaaaa, voilà, you’ve created your first neuropsydia-based program :)

Chapter 1. Tutorials
2. Create a Stroop Task

Contact

For remarks, complaints, suggestions or anything else, please create an issue.

Denomination

The first phase of the Stroop task consists of a training/baseline part referred to as the denomination. Neutral stimuli ("XXXX") are colored in red, blue or green, and the participant must respond as quick as possible.

Let’s take our program skeleton:

```python
import neuropsydia as n
n.start()
# Here will go the program code
n.close()
```

To display stimuli, we will use the `write()` function that we will insert in a `for` loop.

```python
import neuropsydia as n  # Load neuropsydia
n.start()  # Start neuropsydia
n_trials = 10
for trial in range(n_trials):
    n.newpage("grey")  # Neutral grey background
    n.write("+")  # Fixation cross
    n.refresh()  # Display it
    n.time.wait(250)  # Wait 250 ms

    n.newpage("grey")  # Neutral grey background
    n.write("XXXX")  # Load the stimulus
    n.refresh()  # Display it
    n.response()  # Wait for response

n.close()  # Close neuropsydia
```

Run the program (F5). As you can see, it works, but it’s far from being usable. We have to add colors to the stimuli and record the response. Also, add instructions.

```python
import neuropsydia as n  # Load neuropsydia
import numpy as np  # Load numpy

n.start()  # Start neuropsydia

n.instructions("Press LEFT when RED, DOWN when GREEN and RIGHT when BLUE.")

n_trials = 10
for trial in range(n_trials):
    n.newpage("grey")  # Neutral grey background
    n.write("+")  # Fixation cross
    n.refresh()  # Display it
    n.time.wait(250)  # Wait 250 ms
```

1.2. Create a Stroop Task
Much better. Now, we’re gonna analyze the response (if correct or not) and store them within a dict.

```python
import neuropsydia as n  # Load neuropsydia
import numpy as np  # Load numpy
n.start()  # Start neuropsydia

# Initialize data storage
data = {
    "Stimulus": [],
    "Stimulus_Color": [],
    "Answer": [],
    "RT": [],
    "Condition": [],
    "Correct": []
}

# Part 1: Denomination
n.instructions("Press LEFT when RED, DOWN when GREEN and RIGHT when BLUE.")

n_trials = 10
for trial in range(n_trials):
    n.newpage("grey")  # Neutral grey background
    n.write("+")  # Fixation cross
    n.refresh()  # Display it
    n.time.wait(250)  # Wait 250 ms

    stim_color = np.random.choice(["raw_red", "raw_green", "raw_blue"])  # Choose a
    stim = "XXXX"
    n.newpage("grey")  # Neutral grey background
    n.write(stim, style="bold", color=stim_color, size=3)  # Load the stimulus
    n.refresh()  # Display it
    answer, RT = n.response()  # Record response and response time
    if answer == "ESCAPE":  # Enable experiment quit by pressing escape
        quit()

    # Append trial info to
data["Stimulus"].append(stim)
data["Stimulus_Color"].append(stim_color)
data["Answer"].append(answer)
data["RT"].append(RT)
data["Condition"].append(Internal"

    # Categorize the response
```
if answer == "LEFT" and stim_color == "raw_red":
data["Correct"].append(1)
elif answer == "DOWN" and stim_color == "raw_green":
data["Correct"].append(1)
elif answer == "RIGHT" and stim_color == "raw_blue":
data["Correct"].append(1)
else:
data["Correct"].append(0)

n.close()  # Close neuropsydia
print(data)

Conflict

The only thing that will change in the second part is that the stimulus will not always be XXXX, but a color name.

import neuropsydia as n  # Load neuropsydia
import numpy as np  # Load numpy

n.start()  # Start neuropsydia

# Initialize data storage
data = {
    "Stimulus": [],
    "Stimulus_Color": [],
    "Answer": [],
    "RT": [],
    "Condition": [],
    "Correct": []
}

# Part 1: Denomination

n.instructions("Press LEFT when RED, DOWN when GREEN and RIGHT when BLUE.")

n_trials = 10
for trial in range(n_trials):
    n.newpage("grey")  # Neutral grey background
    n.write("+")      # Fixation cross
    n.refresh()       # Display it
    n.time.wait(250)  # Wait 250 ms
    stim_color = np.random.choice(["raw_red", "raw_green", "raw_blue"])  # Choose a color
    stim = "XXXX"
    n.newpage("grey")  # Neutral grey background
    n.write(stim, style="bold", color=stim_color, size=3)  # Load the stimulus
    n.refresh()       # Display it
    answer, RT = n.response()  # Record response and response time
    if answer == "ESCAPE":  # Enable experiment quit by pressing escape
        quit()

    # Append trial info to
    data["Stimulus"].append(stim)
data["Stimulus_Color"].append(stim_color)
data["Answer"].append(answer)
data["RT"].append(RT)
data["Condition"].append("Neutral")

# Categorize the response
if answer == "LEFT" and stim_color == "raw_red":
data["Correct"].append(1)
elif answer == "DOWN" and stim_color == "raw_green":
data["Correct"].append(1)
elif answer == "RIGHT" and stim_color == "raw_blue":
data["Correct"].append(1)
else:
    data["Correct"].append(0)

#==============================================================================
# Part 2: Conflict
#==============================================================================

n.instructions("Press LEFT when RED, DOWN when GREEN and RIGHT when BLUE.")
n_trials = 10
for trial in range(n_trials):
    n.newpage("grey")  # Neutral grey background
    n.write("+")  # Fixation cross
    n.refresh()  # Display it
    n.time.wait(250)  # Wait 250 ms
    stim_color = np.random.choice(["raw_red", "raw_green", "raw_blue"])  # Choose a color
    stim = np.random.choice(["RED", "GREEN", "BLUE"])  # Choose a color
    n.newpage("grey")  # Neutral grey background
    n.write(stim, style="bold", color=stim_color, size=3)  # Load the stimulus
    n.refresh()  # Display it
    answer, RT = n.response()  # Record response and response time
    if answer == "ESCAPE":  # Enable experiment quit by pressing escape
        quit()
    # Append trial info to
    data["Stimulus"].append(stim)
data["Stimulus_Color"].append(stim_color)
data["Answer"].append(answer)
data["RT"].append(RT)

# Categorize the condition
if stim == "RED" and stim_color == "raw_red":
data["Condition"].append("Congruent")
elif stim == "GREEN" and stim_color == "raw_green":
data["Condition"].append("Congruent")
elif stim == "BLUE" and stim_color == "raw_blue":
data["Condition"].append("Congruent")
else:
    data["Condition"].append("Incongruent")

# Categorize the response
if answer == "LEFT" and stim_color == "raw_red":
data["Correct"].append(1)
elif answer == "DOWN" and stim_color == "raw_green":
data["Correct"].append(1)
Finally, just before the end (before the close), we can transform the data dict into a pandas.DataFrame, that can then be easily saved. Don’t forget to import pandas at the beginning.

```python
import neuropsydia as n  # Load neuropsydia
import numpy as np       # Load numpy
import pandas as pd      # Load pandas

n.start()  # Start neuropsydia

# Initialize data storage
data = {"Stimulus": [],
        "Stimulus_Color": [],
        "Answer": [],
        "RT": [],
        "Condition": [],
        "Correct": []}

# Part 1: Denomination
n.instructions("Press LEFT when RED, DOWN when GREEN and RIGHT when BLUE.")

n_trials = 10
for trial in range(n_trials):
    n.newpage("grey")  # Neutral grey background
    n.write("+")      # Fixation cross
    n.refresh()        # Display it
    n.time.wait(250)   # Wait 250 ms

    stim_color = np.random.choice(["raw_red", "raw_green", "raw_blue"])  # Choose a color
    stim = "XXXX"
    n.newpage("grey")  # Neutral grey background
    n.write(stim, style="bold", color=stim_color, size=3)  # Load the stimulus
    n.refresh()        # Display it
    answer, RT = n.response()  # Record response and response time
    if answer == "ESCAPE":  # Enable experiment quit by pressing escape
        quit()

    # Append trial info to
    data["Stimulus"].append(stim)
    data["Stimulus_Color"].append(stim_color)
    data["Answer"].append(answer)
    data["RT"].append(RT)
    data["Condition"].append("Neutral")

    # Categorize the response
    if answer == "LEFT" and stim_color == "raw_red":
        data["Correct"].append(1)
    elif answer == "DOWN" and stim_color == "raw_green":
        data["Correct"].append(1)
```

1.2. 2. Create a Stroop Task
data["Correct"].append(1)
    elif answer == "RIGHT" and stim_color == "raw_blue":
        data["Correct"].append(1)
    else:
        data["Correct"].append(0)

# Part 2: Conflict

n.instructions("Press LEFT when RED, DOWN when GREEN and RIGHT when BLUE.")

n_trials = 10
for trial in range(n_trials):
    n.newpage("grey")  # Neutral grey background
    n.write("+")      # Fixation cross
    n.refresh()       # Display it
    n.time.wait(250)  # Wait 250 ms

    stim_color = np.random.choice(["raw_red", "raw_green", "raw_blue"])
    stim = np.random.choice(["RED", "GREEN", "BLUE"])
    n.newpage("grey")  # Neutral grey background
    n.write(stim, style="bold", color=stim_color, size=3)  # Load the stimulus
    n.refresh()  # Display it
    answer, RT = n.response()  # Record response and response time
    if answer == "ESCAPE":  # Enable experiment quit by pressing escape
        quit()

    # Append trial info to
    data["Stimulus"].append(stim)
    data["Stimulus_Color"].append(stim_color)
    data["Answer"].append(answer)
    data["RT"].append(RT)

    # Categorize the condition
    if stim == "RED" and stim_color == "raw_red":
        data["Condition"].append("Congruent")
    elif stim == "GREEN" and stim_color == "raw_green":
        data["Condition"].append("Congruent")
    elif stim == "BLUE" and stim_color == "raw_blue":
        data["Condition"].append("Congruent")
    else:
        data["Condition"].append("Incongruent")

    # Categorize the response
    if answer == "LEFT" and stim_color == "raw_red":
        data["Correct"].append(1)
    elif answer == "DOWN" and stim_color == "raw_green":
        data["Correct"].append(1)
    elif answer == "RIGHT" and stim_color == "raw_blue":
        data["Correct"].append(1)
    else:
        data["Correct"].append(0)

df = pd.DataFrame.from_dict(data)  # Convert dict to a dataframe
df.to_csv("data.csv")  # Save data
n.close()  # Close neuropsydia

3. Computerize the Beck Depression Inventory (BDI)

Contact

For remarks, complaints, suggestions or anything else, please create an issue.

Code

Now that you’re familiar with the basics, we will try the inverse method. Here’s the full code of the BDI, that we will try to understand, step by step.

```python
import neuropsydia as n  # Load neuropsydia
import pandas as pd

items = {
    1: {0: "I do not feel sad.",
        1: "I feel sad.",
        2: "I am sad all the time and I can't snap out of it.",
        3: "I am so sad and unhappy that I can't stand it."}
    ,
    2: {0: "I am not particularly discouraged about the future.",
        1: "I feel discouraged about the future.",
        2: "I feel I have nothing to look forward to.",
        3: "I feel the future is hopeless and that things cannot improve."}
    ,
    3: {0: "I do not feel like a failure.",
        1: "I feel I have failed more than the average person.",
        2: "As I look back on my life, all I can see is a lot of failures.",
        3: "I feel I am a complete failure as a person."}
    }
    # I cannot legally show the rest of the questions
}

n.start()  # Initialize neuropsydia

participant_id = n.ask("Participant ID:", order=1)  # Get participant id
participant_gender = n.ask("Gender:", order=2)  # Get participant's gender
participant_age = n.ask("Age:", order=3)  # Get participant's age

n.instructions("Please tell if you agree with each following proposition.")  # Instructions

data = {}  # Initialize empty data dict
for item in items:
    data[item] = {}  # For each item, initialize empty sub-dict
    for proposition_number in items[item]:
```

1.3. 3. Computerize the Beck Depression Inventory (BDI)
Problem

The thing is that, unlike many other questionnaires, the BDI has a complex scoring procedure, that prevents us from using the \texttt{questionnaire()} function.
CHAPTER 2

Examples

Contents:

Questionnaires

State-Trait Anxiety Inventory (STAI-Y)

```python
import neuropsydia as n  # Load neuropsydia

questions_dictionary = {

    "Item": {
        1: "I feel calm.",
        2: "I feel secure.",
        3: "I am tense.",
        4: "I feel strained.",
        5: "We're not allowed to reveal all the questions :(")  # As it is the last dict item, no comma after that.
    },

    "Reverse": {
        1: True,
        2: True,
        3: False,
        4: False,
        5: False
    }
}

n.start()  # Initialize neuropsydia

participant_id = n.ask("Participant ID:", order=1)  # Get participant id
participant_gender = n.ask("Gender:", order=2)  # Get participant's gender
```
participant_age = n.ask("Age:", order=3)  # get participant's age

df = n.questionnaire(questions_dictionary,  # The questions
    participant_id=participant_id,
analog=False,  # Lickert-like
    edges=[1, 4],  # Values underneath
    labels=['Almost never', 'Sometimes', 'Often', 'Almost always'],
    style='blue',  # The cursor's color
    instructions_text="A number of statements which people have used to
describe themselves are given below. \nSelect the number that indicate how you feel
right now, that is, at this moment. \nThere are no right or wrong answers. Do not
spend too much time on any one statement but give the answer which seems to
describe your present feelings best.")  # Add instructions at the beginning

# Scoring
score = df["Answer"].sum()

# Cutoff based on Crawford et al. (2011). This just for illustration purposes, adapt
it following your activity.
if score > 56:
    interpretation = "Possible Anxiety"
else:
    interpretation = "No anxiety"

# Add info and save
df["Participant_ID"] = participant_id
df["Participant_Gender"] = participant_gender
df["Participant_Age"] = participant_age
df["Score"] = score
df["Interpretation"] = interpretation
df.to_csv("STAI_" + participant_id + ".csv")
n.close()  # Close neuropsydia

---

**Hint:** Try to run this questionnaire!
- Can you add all missing questions from a legal version?
- Can you computerize the trait version?

---

**Tasks**

**Digit Span**

```python
import neuropsydia as n  # Load neuropsydia
import numpy as np  # For generation of random sequence

n.start()  # Initialize neuropsydia
n.instructions("Listen to the experimenter.")  # Display instructions
```

# Initialize values
```python
number_of_fails = 0  # Initial number of errors
span = 2  # Initial span

while number_of_fails < 3:  # Do it while the number of errors is smaller than 3
    sequence = np.random.randint(10, size=span)  # Generate sequence of size span
    # with ints ranging from 0 to 9
    good_answer = ""  # Initiate an empty good_answer

    for digit in sequence:  # For every digit in the sequence...
        good_answer = good_answer + str(digit)  # Add the current stimulus to the
        good_answer

    n.newpage("grey")  # Load a grey background
    n.time.wait(250)  # Display an empty screen for 250 ms
    n.newpage("grey")  # Load a grey background
    n.write(digit, size=3)  # Load the stimulus
    n.refresh()  # Display the stimulus on screen
    n.time.wait(1000)  # Wait 1000 ms

    # Get answer
    n.newpage("white")
    answer = n.ask("Answer:")

    # Manage result
    if answer == good_answer:
        span = span + 1  # Increase span
        number_of_fails = 0  # Reset value
    else:
        number_of_fails = number_of_fails + 1

n.newpage()  # Load a white background
n.write("Max span: " + str(span-1))  # Write task result
n.refresh()  # Render it on screen
n.time.wait(3000)  # Wait for 3s
n.close()  # Close neuropsydia
```

**Hint:** Try to run this task!

- Can you change the sequence generation so it contains letters rather than digits?
- Can you change the rules so the sequence length increases every two good answers?
- Can you store the results and save them?

---

**Go/No Go**

```python
import neuropsydia as n  # Load neuropsydia
import random  # Import the random module
import pandas as pd  # To manipulate and save data
import numpy as np  # To do some maths

n.start()  # Start neuropsydia
n.instructions("Goal: Hit SPACE whenever a GREEN circle appears. \nIf RED, don't press anything!")  # Display instructions and break line with \\

n.newpage("grey")  # Fill the screen
```

---

2.2. Tasks
n.countdown()  # Display countdown

# Initialize the data storage with a dictionary containing empty lists
data = {"Trial": [],
        "Stimulus": [],
        "ISI": [],
        "RT": [],
        "Response": []}

n_trials = 10  # Number of trials
for trial in range(n_trials):  # Iterate over the number of trials
    stimulus = random.choice(["green", "green", "green", "red"])  # Select a stimulus type
    ISI = random.randrange(start=250, stop=1250, step=250)  # Select the inter-stimuli interval (ISI)

    n.newpage("grey")  # Fill the screen
    n.write("+")  # Fixation cross
    n.refresh()  # Display it on screen
    n.time.wait(ISI)  # Wait

    n.circle(size=2, fill_color=stimulus)  # Display the stimulus (filled with the color selected above)
    n.refresh()  # Display it on screen
    response, RT = n.response(time_max=1000)  # Wait until 1 s and collect the response and its time

    # Categorize the response
    if response == "SPACE" and stimulus == "green":
        response_type = "HIT"  # Hit
    if response != "SPACE" and stimulus == "green":
        response_type = "MISS"  # Miss
    if response == "SPACE" and stimulus == "red":
        response_type = "FA"  # False Alarm
    if response != "SPACE" and stimulus == "red":
        response_type = "CR"  # Correct Rejection

    # Store data by appending each item to its list
    data["Trial"].append(trial)
    data["Stimulus"].append(stimulus)
    data["ISI"].append(ISI)
    data["RT"].append(RT)
    data["Response"].append(response_type)

# Data saving
df = pd.DataFrame.from_dict(data)  # Transform the data dictionary into a proper and savable dataframe
df.to_csv("data.csv")  # Save it

# Quick analysis
RTs = df[df["Response"]=='HIT']["RT"]  # Select the Hits’ RTs
print("Mean RT: " + str(round(RTs.mean(), 2)))  # Print the mean
print("SD RT: " + str(round(RTs.std(), 2)))  # Print the standard deviation
print("Number of False Alarms: " + str(len(df[df["Response"]=='FA'])))  # Print the number of intrusions (false alarms)

n.close()  # Close neuropsydia
Hint: Try to run this task!

- Can you change the number of trials?
- Can you change the ratio of no go trials?

Flanker

```python
import neuropsydia as n  # Load neuropsydia
import pandas as pd     # To manipulate and save data
import numpy as np      # To do some maths

n.start()  # Start neuropsydia
n.instructions("Hit RIGHT or LEFT arrow according to the direction of the CENTRAL arrow.")  # Display instructions

# Initialize cache
cache = {}
for possible_angle in [0, 90, 180]:
    cache = n.preload("arrow-left.png", size=2, rotate=possible_angle, cache=cache)  # Preload images

# Initialize the data storage with a dictionary containing empty lists
data = {"Trial": [],
        "Trial_Type": [],
        "Stimulus_Orientation": [],
        "RT": [],
        "Response":[]}

n.newpage("grey")  # Fill the screen
n.countdown()       # Display countdown

n_trials = 10  # Number of trials
for trial in range(n_trials):  # Iterate over the number of trials

    n.newpage("grey")  # Fill the screen
    n.write("+")      # Fixation cross
    n.refresh()        # Display it on screen
    n.time.wait(500)   # Wait

    # Trial characteristics
    stimulus_angle = np.random.choice([0, 180])  # select target orientation
    trial_type = np.random.choice(["Congruent", "Neutral", "Incongruent"])  # select trial type

    if trial_type == "Congruent":
        distractors_angle = stimulus_angle
    elif trial_type == "Incongruent":
        if stimulus_angle == 0:
            distractors_angle = 180
        else:
            distractors_angle = 0
    else:
        distractors_angle = 90

    # Other code...
```

2.2. Tasks
response, RT = n.response(time_max=1000)  # Wait until 1 s and collect the
response and its time

# Response check
if (response == "LEFT" and stimulus_angle == 0) or (response == "RIGHT" and stimulus_angle == 180):
    response = 1
else:
    response = 0

# Store data by appending each item to its list
data["Trial"].append(trial)
data["Trial_Type"].append(trial_type)
data["Stimulus_Orientation"].append(stimulus_angle)
data["RT"].append(RT)
data["Response"].append(response)

# Data saving
df = pd.DataFrame.from_dict(data)  # Transform the data dictionary into a proper and
# savable dataframe
df.to_csv("data.csv")  # Save it

# Quick analysis
mean_cong = df[(df["Trial_Type"]=="Congruent") & (df["Response"]==1)]["RT"].mean()
mean_neu = df[(df["Trial_Type"]=="Neutral") & (df["Response"]==1)]["RT"].mean()
mean_incong = df[(df["Trial_Type"]=="Incongruent") & (df["Response"]==1)]["RT"].mean()
print("Mean RT Congruent: " + str(round(mean_cong, 2)))  # Print the mean of congruent
print("Mean RT Neutral: " + str(round(mean_neu, 2)))  # Print the mean of neutral
print("Mean RT Incongruent: " + str(round(mean_incong, 2)))  # Print the mean of incongruent

n.close()  # Close neuropsydia

Hint: Try to run this task!
- Can you count the number of errors?
- Can you ask for the participant name at the beginning, and save data using it?
This part of the documentation details the complete neuropsydia for python API.

Core

newpage

```
neuropsydia.newpage(color_name='white', opacity=100, fade=False, fade_speed=60,
                    fade_type='out', auto_refresh=True)
```

Fill the background with a color.

**Parameters**

- `color_name (str, tuple, optional)` – name of the color (see color() function), or an RGB tuple (e.g., (122,84,01)).
- `opacity (int, optional)` – opacity of the color (in percents).
- `fade (bool, optional)` – do you want a fade effect?
- `fade_speed (int, optional)` – frequency (speed) of the fading.
- `fade_type (str, optional)` – “out” or “in”, fade out or fade in.

**Example**

```python
>>> import neuropsydia as n
>>> n.start()
>>> n.newpage("blue")
>>> n.refresh()
>>> n.time.wait(500)
>>> n.close()
```
Notes

Authors

- Dominique Makowski (https://github.com/DominiqueMakowski)

Dependencies

- pygame
- time

refresh

neuropsydia.refresh()

Refresh / flip the screen: actually display things on screen.

Example

```python
>>> import neuropsydia as n
>>> n.start()
>>> n.newpage("blue")
>>> n.refresh()
>>> n.time.wait(500)
>>> n.close()
```

Notes

Authors

- Dominique Makowski (https://github.com/DominiqueMakowski)

Dependencies

- pygame

Display

write

neuropsydia.write(text='Write something here', style='body', x=0, y=0, size=1.0, rotate=0, color='black', background=None, outline=False, outline_size=0.1, outline_color='black', allow=None, wait=None, long_text=False, fast=False)

Display some text on screen.

Parameters

- **text** (*str*) – The text to display.
- **style** (*str*) – ‘body’, ‘psychometry’, ‘psychometry_bold’, ‘light’, ‘bold’, ‘title’, ‘subtitle’ or ‘end’. Can overwrite other parameters such as position, size or allow. You can also insert the name of a system font, or the path to a specific font.
- **x** (*float*) – Horizontal position of the center (from -10 (left) to 10 (right)).
• **y** *(float)* – Vertical position of the center (from -10 (bottom) to 10 (top)).
• **size** *(float)* – Text size.
• **rotate** *(int)* – Rotation angle (0 to 360).
• **color** *(str or tuple)* – Text color. See *neuropsydia.color()*.
• **background** *(str)* – Background color. See *neuropsydia.color()*.
• **outline** *(bool)* – Text outline (unperfect for now, as the outline is larger for horizontal than for vertical lines).
• **outline_size** *(float)* – Outline size.
• **outline_color** *(str or tuple)* – Outline color. See *neuropsydia.color()*.
• **allow** *(str or list)* – Wait until a specific key is pressed (e.g., ‘ENTER’, ['LEFT', ‘RIGHT']) or ‘any’).
• **wait** *(float)* – Wait time (in milliseconds).
• **long_text** *(bool)* – Set to True for longer texts on multiple lines. Then, the x and y parameters are not working, but you can jump lines using ‘backslash + n’ in your text. Some parameters are disabled. Unperfect for now.
• **fast** *(bool)* – Disables some parameters for improved speed.

**Example**

```python
>>> import neuropsydia as n
>>> n.start()
>>> n.write('here is my title', style='title')
>>> n.write('here is my text', font_color='red')
>>> n.write('press ENTER to quit', style='end')
>>> n.close()
```

**Notes**

**Authors**

• Dominique Makowski (https://github.com/DominiqueMakowski)
• Léo Dutriaux (https://github.com/LeoDutriaux)

**Dependencies**

• pygame

**image**

```python
neuropsydia.image(file, x=0, y=0, cache=None, path='', extension='', size=1.0, unit='n', scale_by='height', fullscreen=False, rotate=0, scramble=False, background=None, compress=False, compression=0, allow=None, wait=None, opacity=100, monitor_diagonal=24)
```

Display an image on screen.

**Parameters**
• **file** *(str)* – Image filename.
• **x** *(float)* – Horizontal position of image center (from -10 (left) to 10 (right)).
• **y** *(float)* – Vertical position of image center (from -10 (bottom) to 10 (top)).
• **cache** *(dict)* – Cache of preloaded files.
• **path** *(str)* – File’s path.
• **extension** *(str)* – File’s extension.
• **size** *(float)* – Image size.
• **unit** *(str)* – Size unit. ‘n’ for neuropsydia’s space, can be ‘cm’.
• **scale_by** *(str)* – ‘height’ or ‘width’.
• **fullscreen** *(bool)* – Fullscreen. Disable the size parameter.
• **rotate** *(int)* – Rotation angle.
• **scramble** *(bool)* – Scramble (randomize pixels).
• **background** *(str)* – Background colour.
• **compress** *(bool)* – Enable compression.
• **compression** *(int)* – Compression rate.
• **allow** *(list)* – Wait until a specific key is pressed (e.g., “ENTER”, [“LEFT”, “RIGHT”] or “any”).
• **wait** *(int)* – Wait time (in milliseconds).
• **opacity** *(100)* – Opacity (in percentage).
• **monitor_diagonal** *(int)* – Monitor size (in inches).

Returns cache – The updated cache.

Return type dict

Example

```python
>>> import neuropsydia as n
>>> n.start()
>>> for file in ["img1.png", "img2.png"]:  
>>>     n.newpage()  
>>>     n.image(file)  
>>>     n.refresh()  
>>>     n.time.wait(1000)  
>>> n.close()
```

Notes

Authors

• Dominique Makowski (https://github.com/DominiqueMakowski)
• Léo Dutriaux (https://github.com/LeoDutriaux)
Dependencies

- pygame
- PIL

**line**

`neuropsydia.line(left_x=-5, left_y=0, right_x=5, right_y=0, line_color='black', thickness=1)`

Draw a line.

**Parameters**

- `left_x` *(float)* – Left end horizontal position.
- `left_y` *(float)* – Left end vertical position.
- `right_x` *(float)* – Right end horizontal position.
- `right_y` *(float)* – Right end vertical position.
- `line_color` *(str)* – Line color.
- `thickness` *(float)* – Line thickness.

**Example**

```python
>>> import neuropsydia as n
>>> n.start()
>>> n.line()
>>> n.close()
```

**Notes**

**Authors**

- Dominique Makowski (https://github.com/DominiqueMakowski)

**Dependencies**

- pygame
- pygame.gfxraw

**circle**

`neuropsydia.circle(x=0, y=0, size=10, line_color='black', thickness=0, fill_color='white')`

Draw a circle.

**Parameters**

- `x` *(float)* – Center’s horizontal position.
- `y` *(float)* – Center’s vertical position.
- `size` *(float)* – Diameter.
- `line_color` *(str)* – Circle’s edges color.
• **thickness** (*float*) – Circle’s edges thickness.
• **fill_color** (*str*) – Circle’s fill color.

**Example**

```python
>>> import neuropsydia as n
>>> n.start()
>>> n.circle()
>>> n.close()
```

**Notes**

**Authors**

- Dominique Makowski (https://github.com/DominiqueMakowski)

**Dependencies**

- pygame
- pygame.gfxraw

**rectangle**

```python
def neuropsydia.rectangle(x=0, y=0, width=10, height=10, line_color='black', thickness=1, fill_color=None)
```

Draw a rectangle.

**Parameters**

- **x** (*float*) – Center’s horizontal position.
- **y** (*float*) – Center’s vertical position.
- **width** (*float*) – Rectangle’s width.
- **height** (*float*) – Rectangle’s height.
- **line_color** (*str*) – Rectangle’s edges color.
- **thickness** (*float*) – Rectangle’s edges thickness.
- **fill_color** (*str*) – Rectangle’s fill color.

**Example**

```python
>>> import neuropsydia as n
>>> n.start()
>>> n.rectangle()
>>> n.close()
```
**Notes**

**Authors**

- Dominique Makowski (https://github.com/DominiqueMakowski)

**Dependencies**

- pygame
- pygame.gfxraw

### Input

**response**

```python
neuropsychia.response(allow=None, enable_escape=True, time_max=None, get_RT=True)
```

Get a (keyboard, for now) response.

**Parameters**

- `allow` *(str or list)* – Keys to allow.
- `enable_escape` *(bool)* – Enable escape key to exit.
- `time_max` *(int)* – Maximum time to wait for a response (ms).
- `get_RT` *(bool)* – Return response time.

**Returns**

returns a tuple when get_RT is set to True

**Return type**

str or tuple

**Notes**

**Authors**

- Dominique Makowski (https://github.com/DominiqueMakowski)

**Dependencies**

- pygame

### ask

```python
neuropsychia.ask(text='Write something here:', style='light', x=-8, y=0, order=None, size=1.0, color='black', background='white', hide=False, detach_question=False, question_style='light', question_x=0, question_y=0, question_size=1, question_color='black', question_long_text=False, allow=None, allow_length=None, allow_type=None, allow_max=None, allow_NA=True)
```

Display a question and get the subject’s answer.

**Parameters**

- `text` *(str)* – The question to be displayed.
- `style` *(str)* – “body”, “light” or “bold”.

**Notes**

**Authors**

- Dominique Makowski (https://github.com/DominiqueMakowski)

**Dependencies**

- pygame
• order (int) – For series of questions, it’s sometimes easier to just specify the order (1, 2, 3, ...) and the questions will appear one under the other.

• x (float) – Horizontal position (from -10 (left) to 10 (right)).

• y (float) – Vertical position of the center (from -10 (bottom) to 10 (top)).

• size (float) – Text size.

• color (str or tuple) – Text color. See neuropsydia.color().

• background (str or tuple) – Background color. See neuropsydia.color().

• hide (bool) – Display “****” (stars) instead of the actual answer.

• detach_question (bool) – If set to true, then the question can be manipulated separately using the parameters below.

• question_style (str) – ‘body’, ‘psychometry’, ‘psychometry_bold’, ‘light’, ‘bold’, ‘title’, ‘subtitle’ or ‘end’. Can overwrite other parameters such as position, size or allow. You can also insert the name of a system font, or the path to a specific font.

• question_x (float) – Horizontal position of the question (from -10 (left) to 10 (right)).

• question_y (float) – Vertical position of the question (from -10 (bottom) to 10 (top)).

• question_size (float) – Question size.

• question_color (str) – Question color. See neuropsydia.color().

• question_long_text (bool) – Set to True for longer texts on multiple lines. Then, the x and y parameters are not working, but you can jump lines using ‘backslash + n’ in your text. Some parameters are disabled. Unperfect for now.

• allow (list) – Allow only specific answers (e.g., ["yes", “no”]).

• allow_length (int) – Allow only a specific answer length.

• allow_type (str) – “str”, “int” or “float”. Allow only a specific type.

• allow_max (int) – When allow_type is int or float, set a maximum.

• allow_NA (bool) – Allow absence of response.

Returns answer – Input.

Return type str

Example

```python
>>> import neuropsydia as n
>>> n.start()
>>> answer = n.ask("Hey, you're good?")
>>> print(answer)
>>> n.close()
```
Dependencies
• pygame

scale

```python
def neuropsydia.scale(style='red', x=0, y=-3.3, anchors=None, anchors_space=2, anchors_size=0.7, edges=[0, 100], validation=True, analog=True, step=1, labels='numeric', labels_size=0.8, labels_rotation=0, labels_space=-1, labels_x=0, line_thickness=4, line_length=8, line_color='black', background='white', title=None, title_style='body', title_size=1.75, title_space=1.75, points=True, point_edges=True, reverse=False, separation_labels=None, separation_labels_size=1, separation_labels_rotate=0, separation_labels_space=-1, show_result=False, show_result_shape='circle', show_result_shape_fill_color='white', show_result_shape_line_color='black', show_result_shape_size=0.8, show_result_space=1.25, show_result_size=0.5, show_result_color='black', show_result_decimals=1, cursor_size=1)
```

Draw a scale.

**Parameters**

- `style` *(str)* – style, check `neuropsychia.scale_styles()` function to see what’s available.
- `x` *(float)* – Horizontal position of the center (from -10 (left) to 10 (right)).
- `y` *(float)* – Vertical position of the center (from -10 (bottom) to 10 (top)).
- `anchors` *(list of two str)* – a list of two propositions to be displayed on the sides of the scale (e.g., [not at all, very much]).
- `anchors_space` *(float)* – spacing between the edge and the anchors.
- `anchors_size` *(float)* – size of the anchors’ font.
- `edges` *(list)* – the underlying numerical edges of the scale.
- `validation` *(bool)* – confirm the response with a second left click or withdraw with a right click.
- `analog` *(bool)* – continuous (discrete) scale.
- `step` *(int)* – if analog is True, what are the step to go between the edges (determine the number of points on the scale).
- `labels` *(str or list)* – “num”, “numeric” or “numbers” or list of actual text labels (e.g., [“not at all”, “a bit”, “very much”]).
- `labels_size` *(float)* – Size of labels.
- `labels_rotation` *(float)* – Labels rotation angle.
- `labels_space` *(float)* – Space between scale and labels.
- `labels_x` *(float)* – Horizontal dodging position.
- `line_thickness` *(float)* – Scale line thickness.
- `line_length` *(float)* – Scale line length.
- `line_color` *(str)* – Scale line color.
- `background` *(str)* – Scale background color.
- `title` *(str)* – Scale title/question to ask.
• **title_style** *(str)* – ‘body’, ‘psychometry’, ‘psychometry_bold’, ‘light’, ‘bold’. You can also insert the name of a system font, or the path to a specific font.

• **title_size** *(float)* – Title size.

• **title_space** *(float)* – Space between scale and title.

• **point_center** *(bool)* – Place a point at the center.

• **point_edges** *(bool)* – Place points at the edges.

• **reverse** *(bool)* – The result is scored as inverse.

• **force_separation** *(int)* – Creates visual separations with points.

• **separation_labels** *(list)* – Place labels corresponding to the separations.

• **separation_labels_size** *(float)* – Separation labels size.

• **separation_labels_rotate** *(float)* – Separation labels rotation angle.

• **separation_labels_space** *(float)* – Space between scale and separation labels.

• **show_result** *(bool)* – Add a marker to show the value on scale.

• **show_result_shape** *(str)* – Shape of this marker. “circle” or “rectangle”.

• **show_result_shape_fill_color** *(str)* – Fill color of the marker.

• **show_result_shape_line_color** *(str)* – Line color of the marker.

• **show_result_shape_size** *(float)* – Marker’s shape size.

• **show_result_space** *(float)* – Space between scale and marker.

• **show_result_size** *(float)* – Results text size.

• **show_result_color** *(str)* – Results text color.

• **show_result_decimals** *(int)* – How many decimals to show.

• **cursor_size** *(float)* – Size of the circle cursor.

**Returns** response – Response value.

**Return type** float

### Example

```python
>>> import neuropsydia as n
>>> n.start()
>>> response = n.scale()
>>> n.close()
```

### Notes

**Authors**

- Dominique Makowski (https://github.com/DominiqueMakowski)

**Dependencies**

- pygame

---

28 Chapter 3. Documentation
Create clickable choice boxes.

**Parameters**

- `choices (list)` – List of choices.
- `write_choices (bool)` – Write choices inside the boxes.
- `overwrite_choices_display (list)` – Display different choices (but does not affect what is returned).
- `choices_size (float)` – Choices text size.
- `choices_color (str)` – Choices text color.
- `choices_style (str)` – Choices text style.
- `y (float)` – Vertical position.
- `height (float)` – Boxes height (Should be negative).
- `boxes_space (float)` – Spaces between boxes.
- `boxes_background (str)` – Boxes background color.
- `boxes_edge_color (str)` – Boxes edges color.
- `boxes_edge_size (float)` – Boxes edges width.
- `confirm_edge_color (str)` – When clicked once, to what color should the edges change?
- `confirm_edge_size (float)` – When clicked once, to what width should the edges change?
- `help_list (list)` – Display some additional text for each choice.
- `help_background (str)` – Background color of the help band.
- `title (str)` – Title text.
- `title_position (str)` – Title position. “top” or “left”.
- `title_x (float)` – Title horizontal position.
- `title_space (float)` – Space between choices and title.
- `title_style (str)` – Title style.
- `pictures (list)` – Picture filenames (and path) to put within each box.
- `pictures_size (float)` – Size of those pictures.

**Returns** `response` – The selected choice.

**Return type** `str` or `float`
Example

```python
>>> import neuropsydia as n
>>> n.start()
>>> answer = n.choice(['No', 'Maybe', 'Yes'])
>>> n.close()
```

Notes

Authors

• Dominique Makowski (https://github.com/DominiqueMakowski)

Dependencies

• pygame

Meta

questionnaire

```python
neuropsydia.questionnaire(questions_dictionary, questions_list_key_name='Item', background='white', size=1, show_page_number=True, randomize=False, reverse=False, results_save=False, results_name='questionnaire_results', results_path='', participant_id='', dimensions_mean=False, dimensions_key_name='Dimension', style='red', x=0, y=-3.3, anchors=None, anchors_space=2, anchors_size=0.7, edges=[0, 100], validation=True, analog=True, step=1, labels='numeric', labels_size=0.8, labels_rotation=0, labels_space=-1, labels_x=0, line_thickness=4, line_length=8, line_color='black', title=None, title_style='body', title_size=1, title_space=2, point_center=False, point_edges=True, force_separation=False, separation_labels=None, separation_labels_size=1, separation_labels_rotate=0, separation_labels_space=1, show_result=False, show_result_shape='circle', show_result_shape_fill_color='white', show_result_shape_line_color='red', show_result_size=0.8, show_result_space=1.2, show_result_color='black', instructions_text=None, instructions_top_space=5, show_result_decimals=1, cursor_size=1)
```

A wrapper function for easily creating questionnaires. You can go back and forth in the questions using the LEFT and RIGHT keyboard arrows.

Parameters

• `questions_dictionary (dict)` – A dict of the following structure:

```python
>>> questions_dictionary = {
>>>     "Item": {
>>>         1: "Is Neuropsydia great?",
>>>         2: "Is Neuropsydia not great?",
>>>         3: "Is Python great?",
>>>         4: "Is Python not great?"
>>> }
```
• `questions_list_key_name (str)` – Key name of the sub-dict containing the items.
• `background (str)` – Background color.
• `size (int)` – Question’s size.
• `show_page_number (bool)` – Show page number.
• `randomize (bool)` – Randomize question presentation.
• `reverse (bool)` – Needs a “Reverse” sub-dict with booleans showing which questions are reversed.
• `results_save (bool)` – Save the results.
• `results_name (str)` – Filename.
• `results_path (str)` – Path where to save.
• `participant_id (str)` – Append the participant’s name in the filename.
• `dimensions_mean (bool)` – Compute the mean by dimension. Needs a “Dimension” sub-dict.
• `dimensions_key_name (str)` – Key name of the sub-dict containing dimensions.
• `style (str)` – style, check `neuropsydia.scale_styles()` function to see what’s available.
• `x (float)` – Horizontal position of the center (from -10 (left) to 10 (right)).
• `y (float)` – Vertical position of the center (from -10 (bottom) to 10 (top)).
• `anchors (list of two str)` – a list of two propositions to be displayed on the sides of the scale (e.g., [not at all, very much]).
• `anchors_space (float)` – spacing between the edge and the anchors.
• `anchors_size (float)` – size of the anchors’ font.
• `edges (list)` – the underlying numerical edges of the scale.
• `validation (bool)` – confirm the response with a second left click or withdraw with a right click.
• `analog (bool)` – continuous (discrete) scale.
• `step (int)` – if analog is True, what are the step to go between the edges (determine the number of points on the scale).
• `labels (str or list)` – “num”, “numeric” or “numbers” or list of actual text labels (e.g., ["not at all", “a bit”, “very much"]).
• `labels_size (float)` – Size of labels.
• `labels_rotation (float)` – Labels rotation angle.
• `labels_space (float)` – Space between scale and labels.
• `labels_x (float)` – Horizontal dodging position.
• `line_thickness (float)` – Scale line thickness.
• `line_length (float)` – Scale line length.
• `line_color (str)` – Scale line color.
• **background** – Scale background color.

• **title**(str) – Scale title/question to ask.

• **title_style**(str) – ‘body’, ‘psychometry’, ‘psychometry_bold’, ‘light’, ‘bold’. You can also insert the name of a system font, or the path to a specific font.

• **title_size**(float) – Title size.

• **title_space**(float) – Space between scale and title.

• **point_center**(bool) – Place a point at the center.

• **point_edges**(bool) – Place points at the edges.

• **reverse** – The result is scored as inverse.

• **force_separation**(int) – Creates visual separations with points.

• **separation_labels**(list) – Place labels corresponding to the separations.

• **separation_labels_size**(float) – Separation labels size.

• **separation_labels_rotate**(float) – Separation labels rotation angle.

• **separation_labels_space**(float) – Space between scale and separation labels.

• **show_result**(bool) – Add a marker to show the value on scale.

• **show_result_shape**(str) – Shape of this marker. “circle” or “rectangle”.

• **show_result_shape_fill_color**(str) – Fill color of the marker.

• **show_result_shape_line_color**(str) – Line color of the marker.

• **show_result_shape_size**(float) – Marker’s shape size.

• **show_result_space**(float) – Space between scale and marker.

• **show_result_size**(float) – Results text size.

• **show_result_color**(str) – Results text color.

• **show_result_decimals**(int) – How many decimals to show.

• **cursor_size**(float) – Size of the circle cursor.

**Returns** df – A pandas’ dataframe containing the data.

**Return type** pandas.DataFrame

**Example**

```python
>>> import neuropsydia as n
>>> questions_dictionary = {
>>>     "Item": {
>>>         1: "Is Neuropsydia great?",
>>>         2: "Is Neuropsydia not great?",
>>>         3: "Is Python great?",
>>>         4: "Is Python not great?"
>>>     },
>>>     "Reverse": {
>>>         1: False,
>>>         2: True,
>>>         3: False,
>>> ```
```python
>>> 4: True
>>> },
>>> "Dimension": {
>>> 1: "Neuropsydia",
>>> 2: "Neuropsydia",
>>> 3: "Python",
>>> 4: "Python"
>>> }
>>> };
>>> n.start()
>>> n.questionnaire(questions_dictionary, anchors=["No", "Yes"], results_→save=True, dimensions_mean=True)
>>> n.close()
```

Notes

Authors

- Dominique Makowski (https://github.com/DominiqueMakowski)

Dependencies

- pygame
- pandas

instructions

```
neuropsydia.instructions(text, background='white', color='black', size=1.0, title='INSTRUCTIONS', title_color='black', subtitle=None, subtitle_color='black', end_text='Appuyez sur ENTRER pour commencer.', top_space=5)
```

Help incomplete, sorry.

Parameters NA –

Returns

Return type NA

Example

NA

Dominique Makowski

- pygame 1.9.2

Miscellaneous

scale_styles

```
neuropsydia.scale_styles()
```

Returns available scale styles.
Example

```python
>>> import neuropsydia as n
>>> n.start()
>>> print(n.scale_styles())
>>> n.close()
```

Notes

Authors

• Dominique Makowski (https://github.com/DominiqueMakowski)
Installation

Installation can be easily done using pip:

```
pip install https://github.com/neuropsychology/Neuropsydia.py/zipball/master
```

**Note:** A fully working python distribution (including Neuropsydia) can be downloaded [here](https://github.com/neuropsychology/Neuropsydia.py/zipball/master).
Questions? Help? Movie Recommendations?

Do it on the dedicated chat!
CHAPTER 6

Index

- genindex
- modindex
- search
Index

A
ask() (in module neuropsydia), 25

C
choice() (in module neuropsydia), 29
circle() (in module neuropsydia), 23

I
image() (in module neuropsydia), 21
instructions() (in module neuropsydia), 33

L
line() (in module neuropsydia), 23

N
newpage() (in module neuropsydia), 19

Q
questionnaire() (in module neuropsydia), 30

R
rectangle() (in module neuropsydia), 24
refresh() (in module neuropsydia), 20
response() (in module neuropsydia), 25

S
scale() (in module neuropsydia), 27
scale_styles() (in module neuropsydia), 33

W
write() (in module neuropsydia), 20