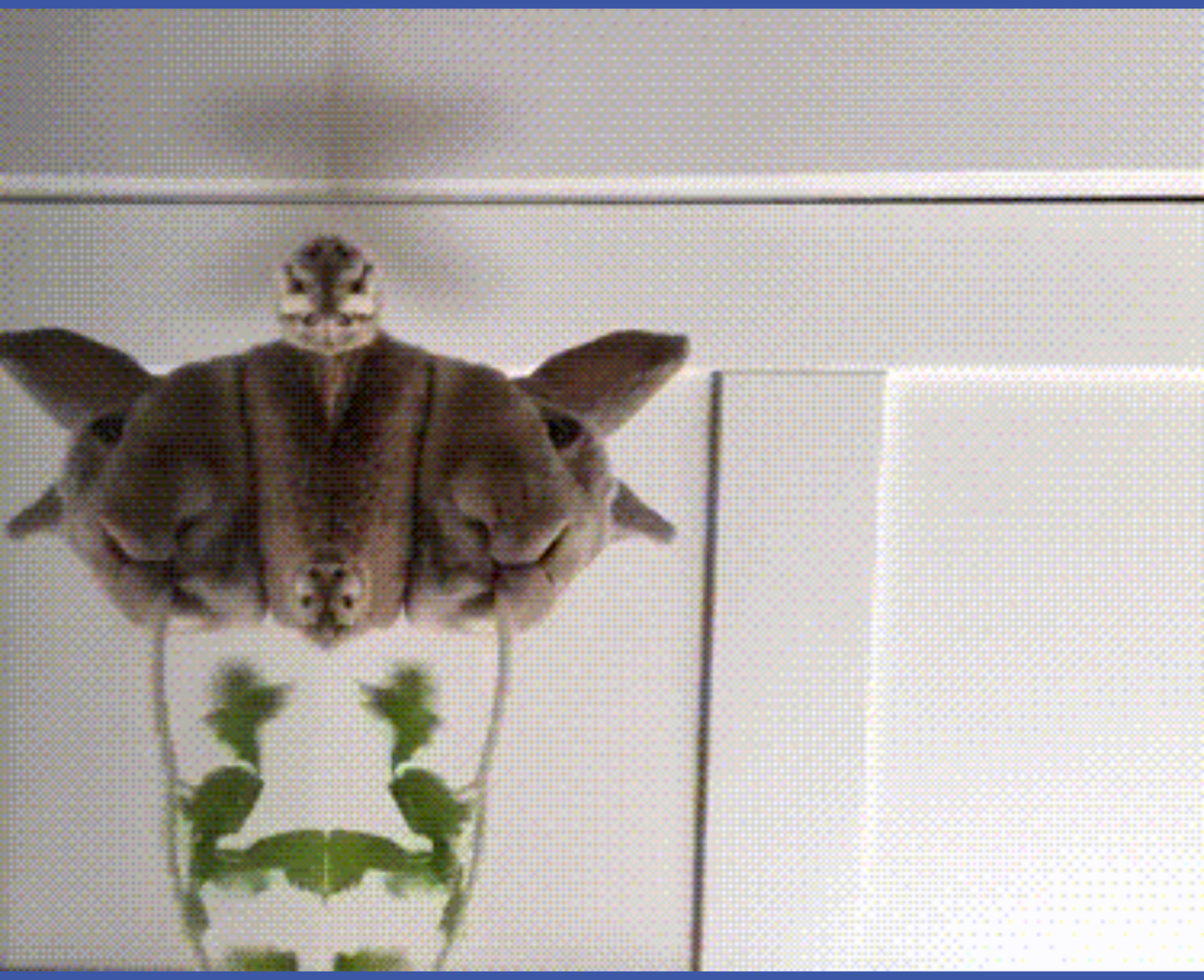
# Data science fundamentals
# Lesson 3

Hay Kranen

Monday October 1st, 2018



# Today's programme

Recap
Variables and memory in Python
Break
Lists
Exercise
Advanced use of lists
Exercise
Lunch break

# RECAP

github.com

Features Business Explore Marketplace Pricing Search Sign in or Sign up

hay / hu-dsf

Watch 3 Star 4 Fork 7

Code Issues 0 Pull requests 0 Projects 0 Insights

Course materials for the HU course Data Science Fundamentals

31 commits 1 branch 0 releases 2 contributors

Branch: master New pull request Find file Clone or download

hay Merge pull request #2 from yvonnebutselaar/master ... Latest commit 56050cb a minute ago

| | | |
|---|---|---|
| assets | Completely updating README and adding a fancy image | 2 days ago |
| examples | Moving notebooks to examples, adding exercises and slides | 7 days ago |
| exercises | Adding more exercises by students | 7 minutes ago |
| slides | Adding week 1, lesson 2 slides | 2 days ago |
| .gitignore | Moving notebooks to examples, adding exercises and slides | 7 days ago |
| README.md | Changing 'week' to 'lesson' | 4 minutes ago |
| cheatsheet.md | Adding more terminal things to the cheatsheet | 2 days ago |
| students.md | Update students.md | a day ago |

README.md

| Real weeks | Master weeks | ~~DSF weeks~~ | Lesson |
|---|---|---|---|
| Week 40 | Week 5 | ~~Week 2~~ | Lesson 3 |

## Examples

For every week there is a Jupyter Notebook containing examples relating to the subjects of that week

- Lessons 1 & 2: Math, variables, `print()`, `input()`, comparisons, types, `if`, string methods
- Lessons 3 & 4: Lists, `for`, `while`, files, `csv`
- Lessons 5 & 6: dicts, `json`, f-strings, HTTP api's
- Lessons 7 & 8: Reddit / `praw`, Pandas
- Lessons 9 & 10: Web scraping

## Exercises

These are optional exercises you can make during the lesson to test your knowledge. You **don't** need to submit these with the final assignment.

- Lesson 1: Myfitnesspal
- Lesson 2: Lovetest

## Slides

These are PDF versions of the slides i use during classes.

- Lesson 1
- Lesson 2
- Lesson 3
- Lesson 4
- Lesson 5
- Lesson 6
- Lesson 7
- Lesson 8
- Lesson 9
- Lesson 19

hay / hu-dsf



Branch: master hu-dsf / students.md



yvonnebutselaar Update students.md 5a5c502 a day ago

3 contributors

31 lines (22 sloc) 913 Bytes



This is a list of students and teachers of the HU master Data Driven Design with their Github accounts

# Teachers

- Aletta
- Erik
- Hay
- Jonas

# Students

- Alex
- Arjen
- Bente
- Bob
- Chrisje
- Denilo
- Jorn
- Koen
- Kolien
- Leon

| Exercises | Assignment |
| --- | --- |
| For practice | To pass the course |
| Optional | Mandatory |
| Work on in class | Work on in your own time |
| Share during class on Slack | Deliver all before deadline |

# Build your own love test

```python
#asking for the names
name_1 = input("What is your name?")
name_2 = input("What is your lovers name?")

#make the names all lowercase
name_1 = name_1.lower()
name_2 = name_2.lower()

#delete spaces around the names
name_1 = name_1.strip()
name_2 = name_2.strip()
```

```python
name1 = input("What is your name?").lower().strip()
name2 = input("What the name of the second person?").lower().strip()
```

```
#the comments below is the code for the basic assignment
#if name_1 == name_2:
#       print("You are in love with yourself")
#elif name_1 > name_2:
#       print("It's a match!")
#else:
#       print("You are not a match, go dating again")
```

```
# this is the basic assignment
"""
if name_1 == name_2:
    print("Match made in heaven")

elif name_1 > name_2:
    print("Matchiematchie")

else:
    print ("Neup, download tinder")
"""
```

95 lines (94 sloc) | 2 KB

Raw Blame History


```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 62,
   "metadata": {},
   "outputs": [],
   "source": [
    "# variable names\n",
    "name1 = input(\"First name \")\n",
    "name2 = input(\"Second name \")"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 63,
   "metadata": {},
   "outputs": [
    {
```

```python
#Giving the difference a percentage outcome
if positive_difference == 0:
    print("You are a 100% match made in heaven!")

elif positive_difference == 1:
    print("You are a 90% match, you will be a good couple!")

elif positive_difference == 2:
    print("You are a 80% match, you will be a good couple!")
```

```python
if count_dif == 0:
        print("You are a 100% match!")
elif count_dif == 1:
        print("You are a 80% match")
elif count_dif == 2:
        print("You are a 60% match")
```

```python
if difference_two < 1:
        print("Perfect match, 100% match")

elif difference_two < 2:
        print("Almost a perfect match, keep working on your relation, 80% match")

elif difference_two < 3:
        print("You have to work harder to get a good relationship, 60% match")

elif difference_two < 4:
```

```python
#function to compare the names
def compare_names():
    name_length = len(name)
    lover_name_length = len(lover_name)
    count = 0

    if name_length == lover_name_length:
        while count < (name_length * lover_name_length) + 2:
            print(count)
            count += 1
    elif name_length != lover_name_length:
        while count < (name_length * lover_name_length) * 2:
            print(count)
            count += 1
    else:
        print("0...")
```

```python
def print(text):
    # Print the text to the screen

print("Hello, world")

def say_hello():
    print("Hello!")

say_hello()
```

```
#Count the letters
count_name_1 = len(name_1)
count_name_2 = len(name_2)
count_dif = abs(count_name_1 - count_name_2)
```

| Function | Description | Example |
|---|---|---|
| `abs()` | Returns the absolute value: makes any number positive | `abs(-20) # 20` |
| `len()` | Returns the length of a string | `len("Hello") # 5` |
| `round()` | Rounds up to the nearest integer | `round(1.2) # 1`<br>`round(1.5) # 2` |
| `max()` | Returns the largest number from a series | `max(4, 20, 1) # 20` |
| `min()` | Returns the lowest number from a series | `max(-20, 1, -3) # -20` |

```python
#counting the characters in both names
count_name_1 = len(name_1)
count_name_2 = len(name_2)

#because the difference between the amount of cha
count_dif = abs(count_name_1 - count_name_2)

#if the difference between the amount of characte
if count_dif == 0:
        print("You are a 100% match!")
elif count_dif == 1:
        print("You are a 80% match")
elif count_dif == 2:
        print("You are a 60% match")
elif count_dif == 3:
        print("You are a 40% match")
elif count_dif == 4:
        print("You are a 20% match")
elif count_dif == 5:
        print("You are a 10% match")
else:
        print("You are a 0% match")
```

```python
#Count the letters
count_name_1 = len(name_1)
count_name_2 = len(name_2)
count_dif = abs(count_name_1 - count_name_2)

#turning it into %
if count_dif == 0:
    print("You're a 100% match")
elif count_dif == 1:
    print("You're a 80% match")
elif count_dif == 2:
    print("You're a 60% match")
elif count_dif == 3:
    print("You're a 40% match")
elif count_dif == 4:
    print("You're a 20% match")
elif count_dif == 5:
    print("You're a 10% match")
else:
    print("Download Tinder,Grindr or She")
```

# VARIABLES AND MEMORY IN PYTHON

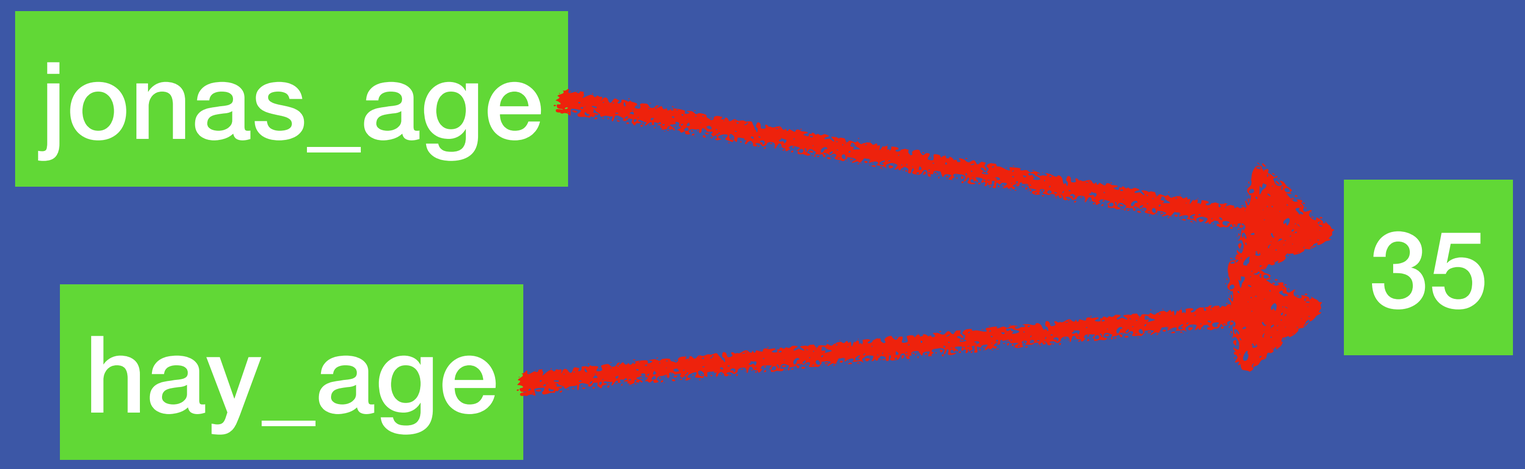
```python
name = "Barrie"

name.upper()
print(name) # "Barrie"

print(name.upper()) # "BARRIE"

name = name.upper()
print(name) # "BARRIE"
```

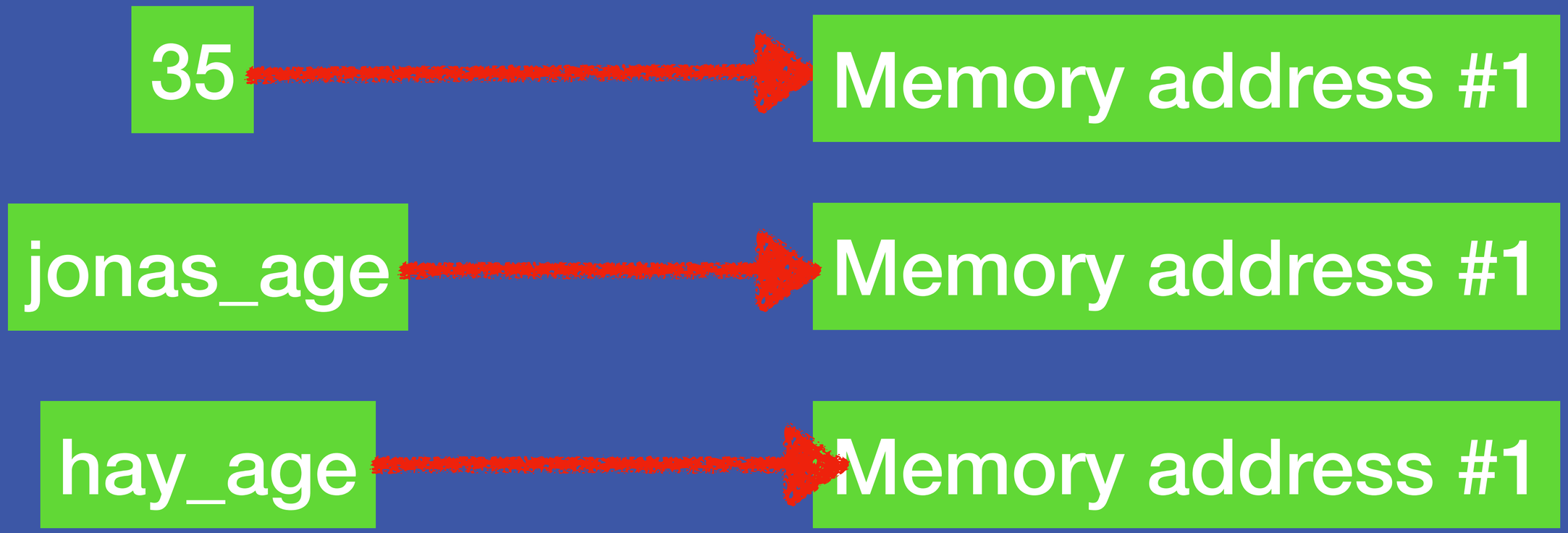
```python
hay_name = "Hay Kranen"
hay_age = 35
hay_is_male = True

jonas_name = "Jonas Moons"
jonas_age = 35
jonas_is_male = True
```

| Variable name | Value |
|---|---|
| hay_name | "Hay Kranen" |
| hay_age | 35 |
| hay_is_male | True |

| Variable name | Value |
|---|---|
| jonas_name | "Jonas Moons" |
| jonas_age | 35 |
| jonas_is_male | True |

How many variables?
How many values?


jonas_age
hay_age
35


35
Memory address #1
jonas_age
Memory address #1
hay_age
Memory address #1

| Variable name | Value |
| --- | --- |
| hay_name | "Hay Kranen" |
| hay_age | 35 |
| hay_is_male | True |

| Variable name | Value |
| --- | --- |
| jonas_name | "Jonas Moons" |
| jonas_age | 35 |
| jonas_is_male | True |

| Variable name | Address |
| --- | --- |
| hay_name | #2 |
| hay_age | #3 |
| hay_is_male | #4 |
| jonas_name | #1 |
| jonas_age | #3 |
| jonas_is_male | #4 |

| Address | Value |
| --- | --- |
| #1 | "Jonas Moons" |
| #2 | "Hay Kranen” |
| #3 | 35 |
| #4 | True |

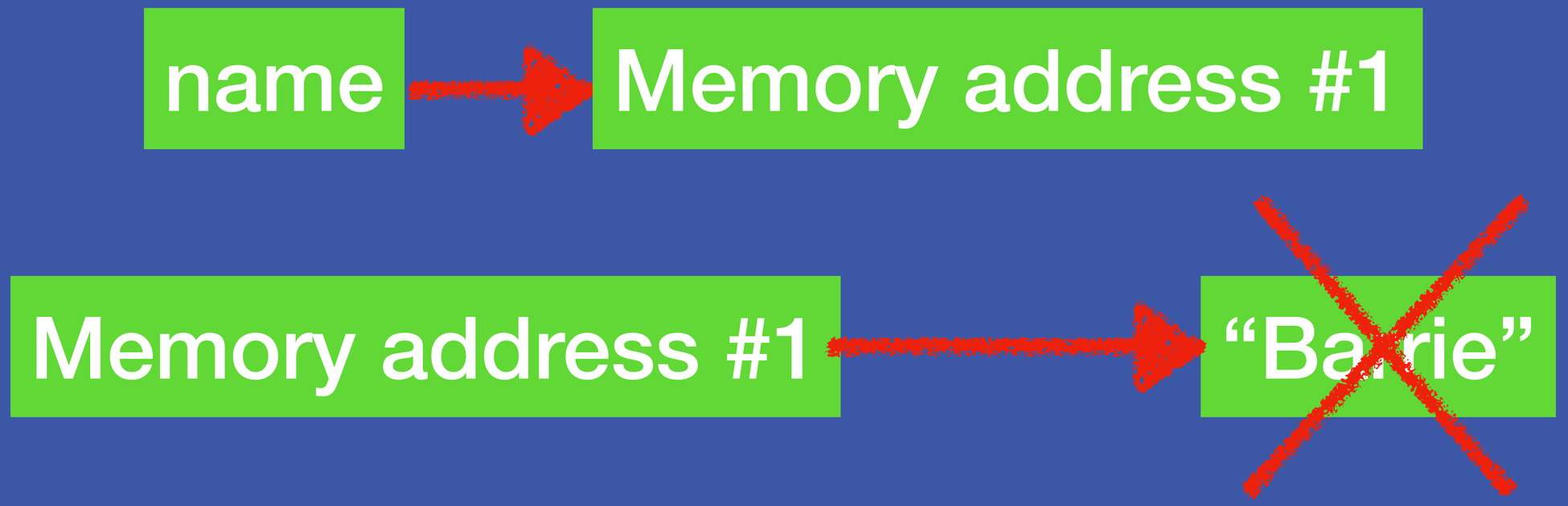
```
hay_age = 35
jonas_age = hay_age
hay_age = hay_age + 1
```

```
hay_age = 35
jonas_age = 35
hay_age = hay_age + 1
```

```
jonas_age = 35
hay_age = jonas_age
hay_age = 36
```

```
Put integer 35 in memory location #1
Point variable 'hay_age' to memory location #1
Point variable 'jonas_age' to memory location #1
Put integer 36 in memory location #2
Point variable 'hay_age' to memory location #2
```

```python
name = "Barrie"

name.upper()
print(name) # "Barrie"

print(name.upper()) # "BARRIE"

name = name.upper()
print(name) # "BARRIE"
```


name
Memory address #1
Memory address #1
“Barrie”

# *In Python, everything is an object*

| | |
|---|---|
| value | "Barrie" |
| lower() | "barrie" |
| upper() | "BARRIE" |
| length | 6 |
| size in memory | 55 |
| type | str |

name → Memory address #1

Memory address #1 →

| | |
|---|---|
| value | "Barrie" |
| lower() | "barrie" |
| upper() | "BARRIE" |
| length | 6 |
| size in memory | 55 |
| type | str |

# *Lets write this out*

```
name = "Barrie"
name = name.upper()
print(name)
```

# LISTS

| Address | Value |
| --- | --- |
| #1 | "Jonas Moons" |
| #2 | "Hay Kranen” |
| #3 | 35 |
| #4 | True |

**Reference**
Use the **examples-2 notebook** from the Github repo and try out everything until you get to “the ‘in’ operator”

**Basic**
Write a program that loops through the names of three friends you predefine in a list. Print out the names and the length of the name.

**Advanced**
Extend the program so that after printing the name the program asks for their favourite snack. After completing the loop, loop through the names again and print the name and their favourite snack.

```
names = ["Barrie", "Tinus", "Hans"]

index = 0
for name in names:
    print(name)
    print(index)
    index = index + 1
```

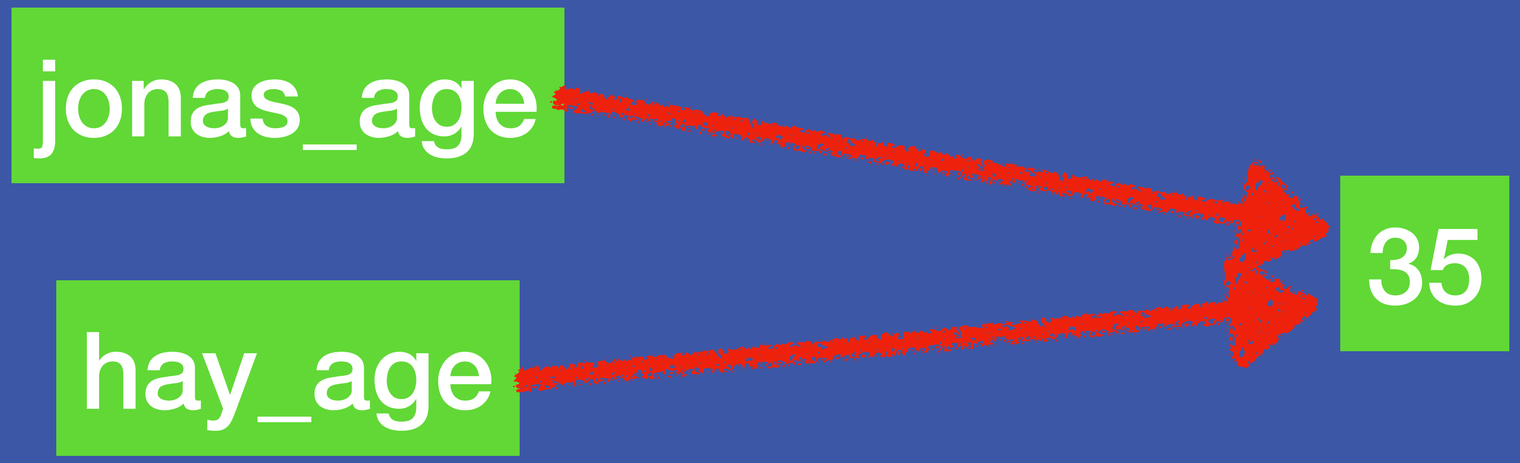
```python
name = "Barrie"
name.upper()
print(name) # "Barrie"

names = []
names.append("Barrie")
print(names) # ["Barrie"]
```

| Type | Storage | Updates |
| --- | --- | --- |
| `int` | `literal` | `immutable` |
| `str` | `literal` | `immutable` |
| **`list`** | **`container`** | **`mutable`** |

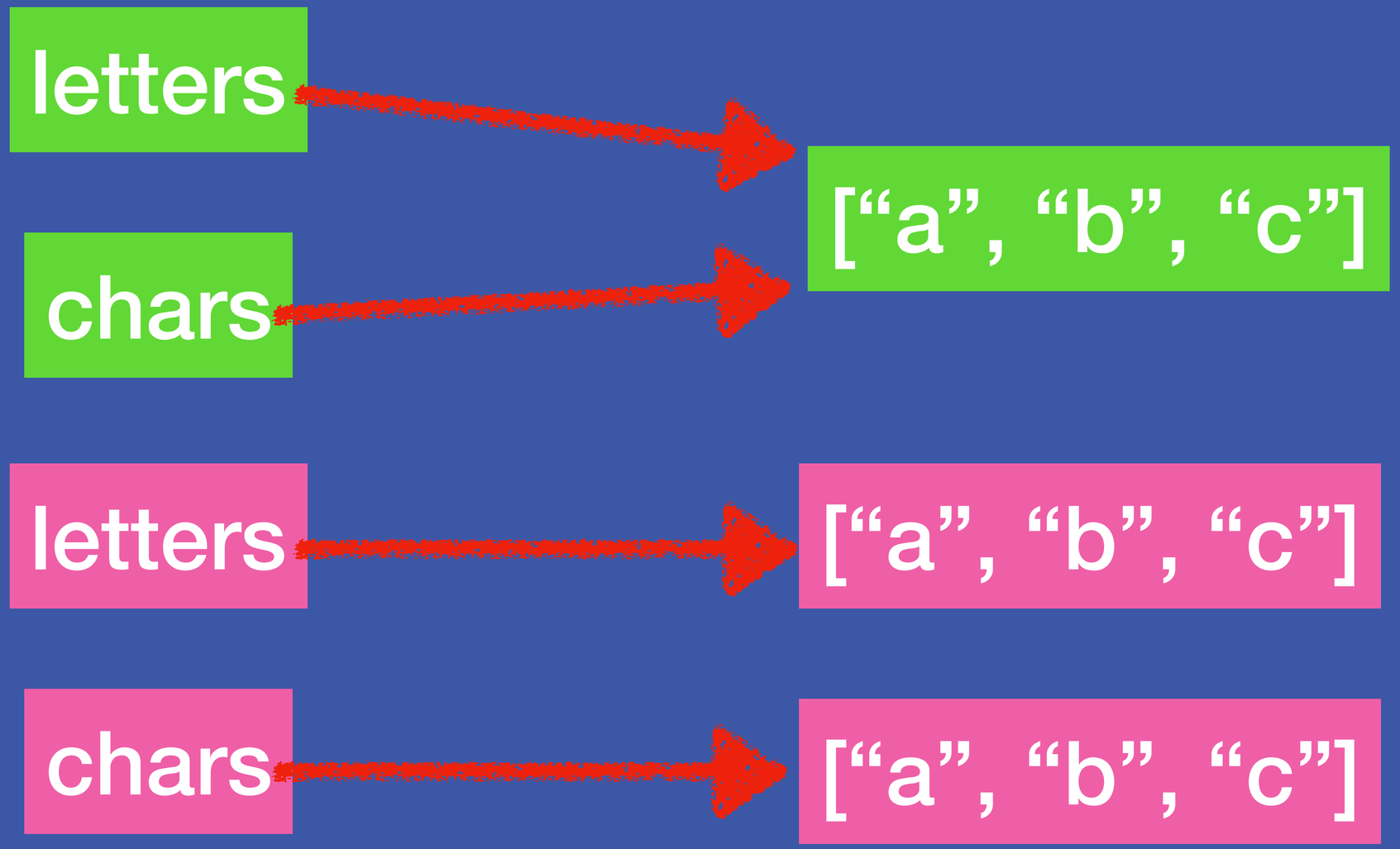
| Variable name | Address |
| --- | --- |
| hay_name | #2 |
| hay_age | #3 |
| hay_is_male | #4 |
| jonas_name | #1 |
| jonas_age | #3 |
| jonas_is_male | #4 |

| Address | Value |
| --- | --- |
| #1 | "Jonas Moons" |
| #2 | "Hay Kranen” |
| #3 | 35 |
| #4 | True |

```
jonas_age = 35
hay_age = 35
```



```
letters = ["a", "b", "c"]
chars = ["a", "b", "c"]
```



# ADVANCED LISTS

# *Spell checker*

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

**Reference**
Use the **examples-2 notebook** from the Github repo and try out everything you haven't done until 'Day 2'

**Basic**
Write a program that has a short list of words (around 5 to 10) that are considered 'wrong'. Ask the user for a sentence. Loop through all the words in the sentence, check if the word is 'wrong', and if so, display an error.

1 Define a **list** of words
2 Ask for a sentence using **input()**
3 **split()** that sentence in a new list of words
4 Loop through the new sentence list using the **for** statement
5 In the for loop, check if your word is **in** the list you defined in step 1
6 If that is the case, **print** a warning

**Advanced**
Apart from warning the user when looping through the sentence print back a 'corrected' sentence at the end where you replace all the 'wrong' words with 'corrected' words.