Getting Started with Programming: python for Psychologists

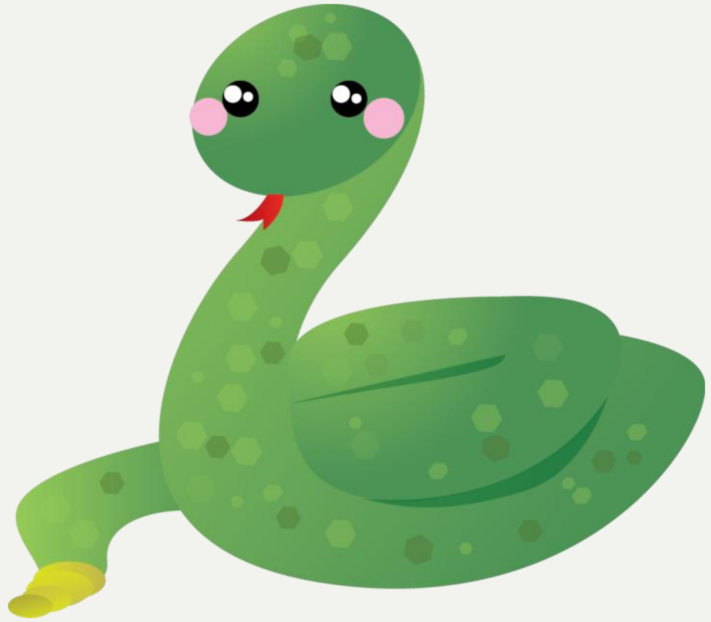
A Guided Tour and Walkthrough to get you started!

Overview and Scope

- Aim today is to cover:
  1. What and why
  2. Secrets of Success
  3. Language Fundamentals
  4. Demonstration
  5. Tips for Future Learning
- Targeted at beginners
- However, aim is not to cover all the language from ground up
  - More important you learn the concepts and problems first

Don’t Run – Python Isn’t Scary!

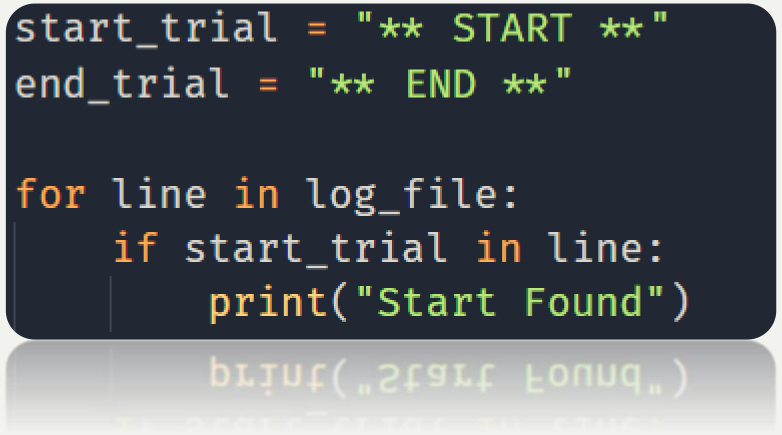
Initial Expectations of Python


Python after a few years

Why it’s actually called Python

What is Python?

- Open source (free) general-purpose programming language.
  - *An ecosystem and language for telling a computer what to do*
- Available for download from http://www.python.org
- Developed by Guido van Rossum in the early 1990s



```
start_trial = "** START **"
end_trial = "** END **"

for line in log_file:
    if start_trial in line:
        print("Start Found")
```

Why Learn Python (or any PL)?

- Computers are integral part of our work (no sign of stopping)
- Learning even the basics of Python will:
  - Improve your ability to work with computers, data, and information
  - Make using computers more enjoyable and less frustrating
  - Save you time and human error
  - Make you more employable, both inside and outside of research.
- You can grapple with more complex questions
- Learning one of the scripting languages is a life-long skill

Why Start with Python?

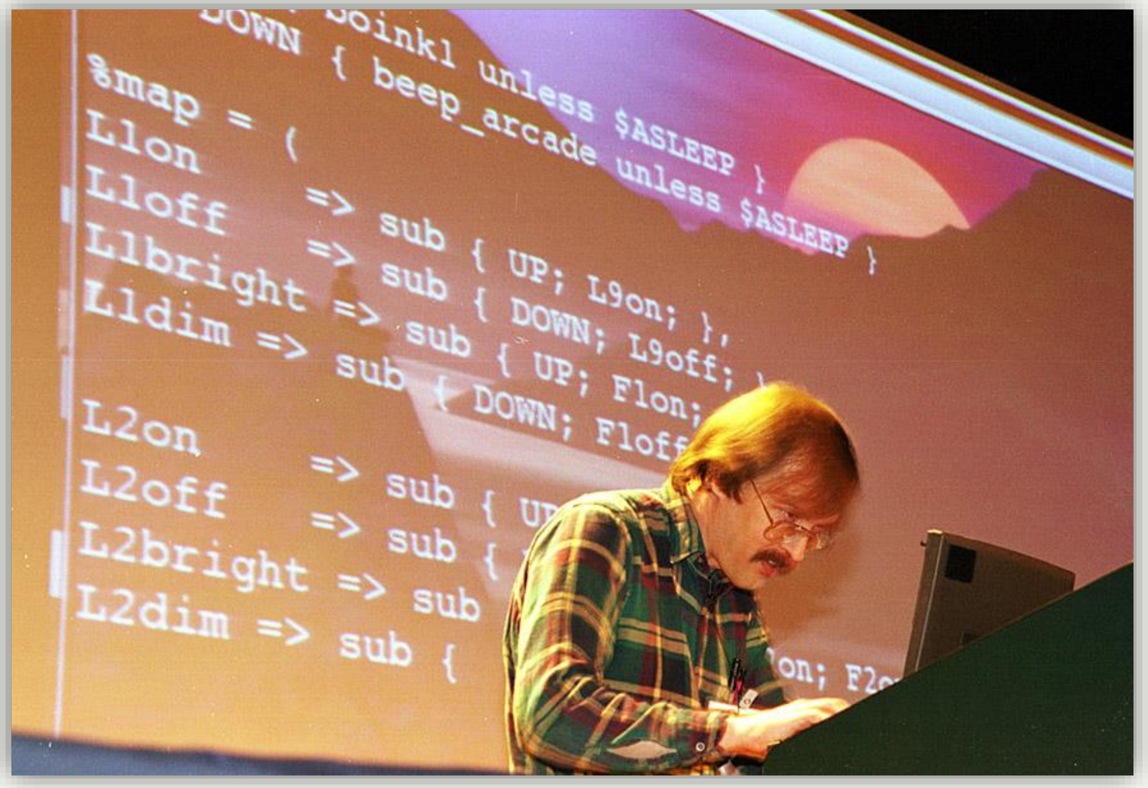
- Python has one of the largest software communities
- Actively developed and thriving ecosystem
- It is the swiss army knife of computational problems
- Enforces a strict style, yet incredibly learnable.
- “Simple is better than complex. Complex is better than complicated.” *https://www.python.org/dev/peps/pep-0020/*
- It will make it easier to move onto other environments like R, Bash, Ruby, Golang, XML, etc…

Use Cases for Social Scientists

- Automating simple computing tasks
  - **Moving, renaming, and sorting large amounts of files and folders**
  - **Text/string processing (e.g., finding words in larger corpus/dataset)**
- Data processing, analysis and organisation
  - **Aggregating and cleaning data sets**
  - Transforming CSV’s to different formats (wide to long, anyone)
  - Statistics, sorting, and classifying text and numbers
- New methods and analytical techniques
  - **Web scraping**, machine learning, computational modelling
  - Reproducibility in Data Analysis and Processing
- Experimental design (psychopy; custom built)

What’s the secret to becoming a good programmer? What should I do?

The 3 Qualities of an Amazing Programmer...



```
DOWN    boinkl unless $ASLEEP }
        { beep_arcade unless $ASLEEP }
%map = (
L1on        => sub { UP; L9on; },
L1off       => sub { DOWN; L9off;
L1bright => sub { UP; Flon;
L1dim => sub { DOWN; Floff

L2on        => sub { U
L2off       => sub {
L2bright => sub
L2dim => sub {
                    on; F2
```

Lary Wall

1. Laziness

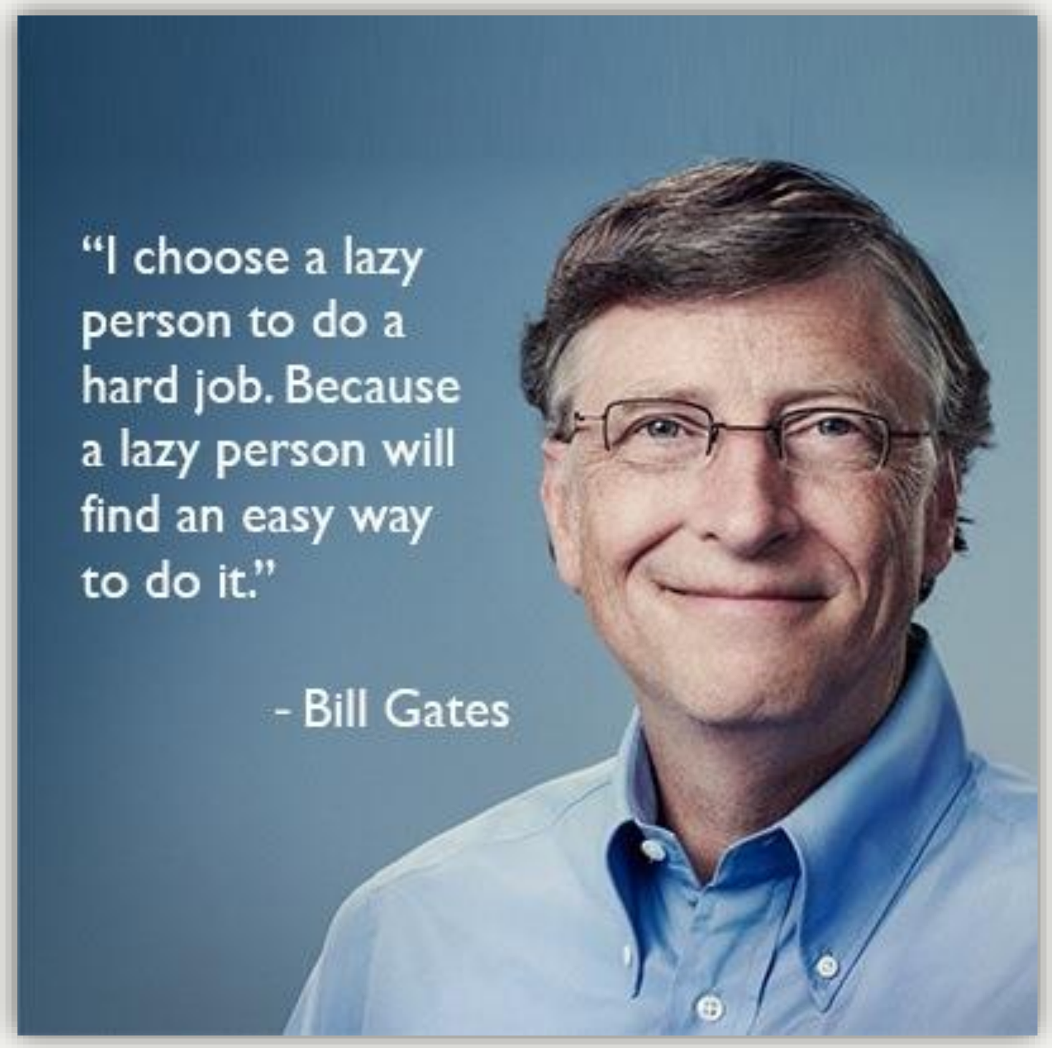
The quality that makes you go to great effort to reduce overall energy expenditure. It makes you write labor-saving programs that other people will find useful, and document what you wrote so you don't have to answer so many questions about it.

2. Impatience

The anger you feel when the computer is being lazy. This makes you write programs that don't just react to your needs, but actually anticipate them. Or at least pretend to.

3. Hubris

Excessive pride, the sort of thing Zeus zaps you for. Also the quality that makes you write (and maintain) programs that other people won't want to say bad things about.


"I choose a lazy person to do a hard job. Because a lazy person will find an easy way to do it."
- Bill Gates

Michael’s Twenty Minute Rule to Success

Never do a repetitive computer-based task manually for longer than 20 minutes unless you are **absolutely sure** there’s no better way.

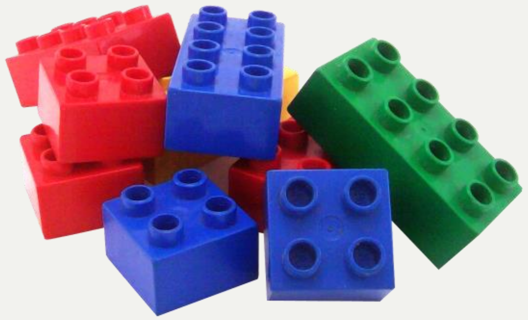
- In almost every case, it will be quicker to learn how to automate it
- When it’s not, think whether this is something you might do again…
- If you don’t think you have the skills to do the task - at least Google it and see just how ~~im~~possible it really is!
- If you have to learn off-the-clock, that’s a perfect learning tool!
- A tale…
- As we go through the exercises think about the 20 minute rule, and the qualities of a programmer.

Guided Tutorial

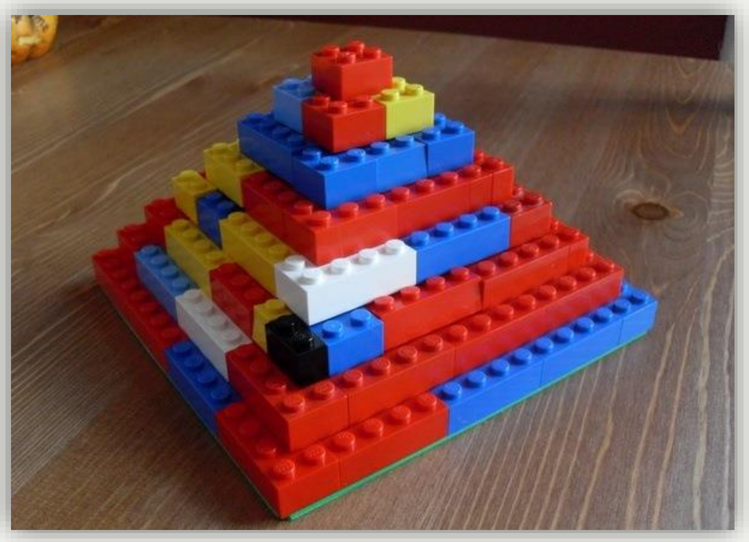
Let’s get the basics and up going!

What did I Learn? Building Blocks

- You learnt the traits and strategies of a good programmer
- We started with the fundamental building blocks of all programs
  - Variables and types
  - Conditionals (if, else)
  - For Loops and flow control
  - Functions
- We saw how to combine basic elements to make quite complex and practical programs.



Basics everyone starts with



What you built

What you were told to build
…by last week

Finding the Answer on Google

Common Questions

- How do I get started when my goal is large?
- What should I do after this class?
- How do I get proficient?
- Books, tutorials, exercises?

How to start something big

- Break down what you are doing step by step into a series of instructions (just like a manual)
- Google how to do each step with ‘in Python’
  - “How do I open CSV in Python”
  - “How do I change CSV row in Python”
- Translate your instructions into code

Next Steps

- If you want to continue using Python, you'll need to:
  - Install Python 3 from python.org and install
  - Install a (good) Code Editor
  - Can demonstrate this to interested people afterwards
- Learn the basic syntax and building blocks
  - Online tutorials, combined with Youtube (be wary of generalisation)
  - https://github.com/jorgegonzalez/beginner-projects
- Then, start applying these to solve data with your own work

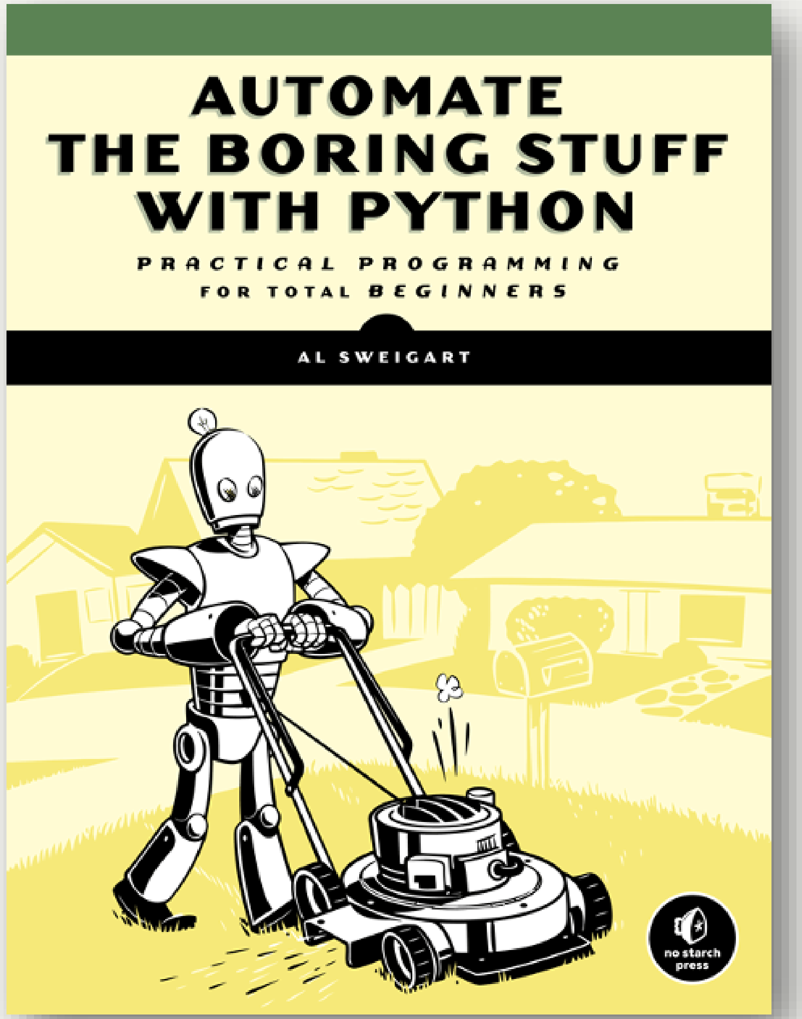
Getting Proficient

- Follow the 20 minute rule and be lazy
- Work on real problems rather than toy ones ASAP
- Be disciplined and persistent if that works for you, but in the long run, try and find the fun in it!

- Keep trying when errors happen
- If you can’t find the right ‘brick’: Ask friends to help you (even if they are just Googling as well).

How I use Resources: Books

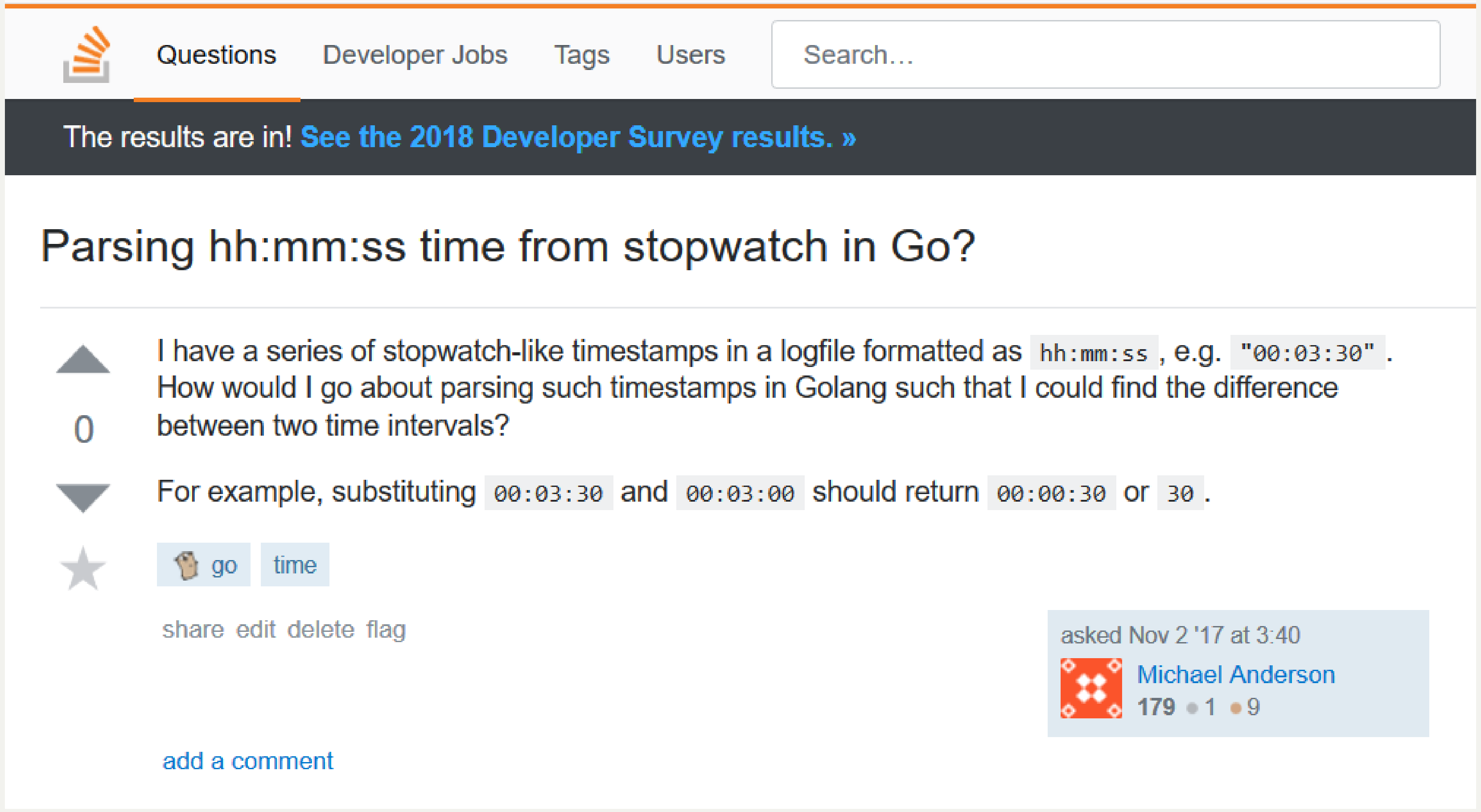
- I don’t tend to actually work through books cover to cover.
- I sometimes use for concept clarification
- I tend to find them more useful as resources to skim through and get inspiration or a more holistic idea of what a particular language can do.


AUTOMATE THE BORING STUFF WITH PYTHON
PRACTICAL PROGRAMMING FOR TOTAL BEGINNERS
AL SWEIGART
no starch press

How I use Resources: Documentation/Google

- Keep using Google
- As soon as you get to the point that you can find what you are looking for in the official documentation and understand ~15% of it – use that as a comparison resource.
  - https://docs.python.org/3/library/csv.html
  - I know it’s dry, but once you get used to the logic it’ll all click

Stack Overflow



The results are in! See the 2018 Developer Survey results. »

Parsing hh:mm:ss time from stopwatch in Go?

0

I have a series of stopwatch-like timestamps in a logfile formatted as `hh:mm:ss`, e.g. `"00:03:30"`. How would I go about parsing such timestamps in Golang such that I could find the difference between two time intervals?

For example, substituting `00:03:30` and `00:03:00` should return `00:00:30` or `30`.

go time

share edit delete flag

asked Nov 2 '17 at 3:40
Michael Anderson
179 ● 1 ● 9

add a comment

4

✓

You may subtract one time from another.

```
package main

import (
    "fmt"
    "time"
)

func main() {

    t1, err := time.Parse("15:04:05", "03:00:30")
    t2, err := time.Parse("15:04:05", "03:00:00")

    fmt.Println(err, t1.Sub(t2), t1.Sub(t2).Seconds())
}
```

Working example - https://play.golang.org/p/VegXLBvfnM

UPDATE: How to cope with durations over `24:00:00`? This way they can be parsed manually:

```
func Parse(st string) (int, error) {
    var h, m, s int
    n, err := fmt.Sscanf(st, "%d:%d:%d", &h, &m, &s)
    fmt.Print(n, err)
    if err != nil || n != 3 {
        return 0, err
    }
    return h*3600 + m*60 + s, nil
}
```

https://play.golang.org/p/HHSRHzaGqT

share edit flag

edited Nov 2 '17 at 4:42

answered Nov 2 '17 at 3:58

Eugene Lisitsky

6,483 ● 1 ● 14 ● 39

Tutorials

- **https://nealcaren.github.io/python-tutorials/**
- https://www.codecademy.com/learn/python
- http://www.learnpython.org/
- https://developers.google.com/edu/python/
- http://www.tutorialspoint.com/python/
- https://www.python.org/about/gettingstarted/
- http://www.data-analysis-in-python.org/
- https://swcarpentry.github.io/python-novice-inflammation/