Kickstart-kursus i programmering 23 dag 2

Daniel Spikol
ds@di.ku.dk

DIKU
Københavns Universitet

15 august 2023

Recap from Yesterday

- Marshmallow Challenge - constant prototyping as a problem-solving method
- Coordinate System
- Drawing with Processing
- Variables
- Functions maybe...

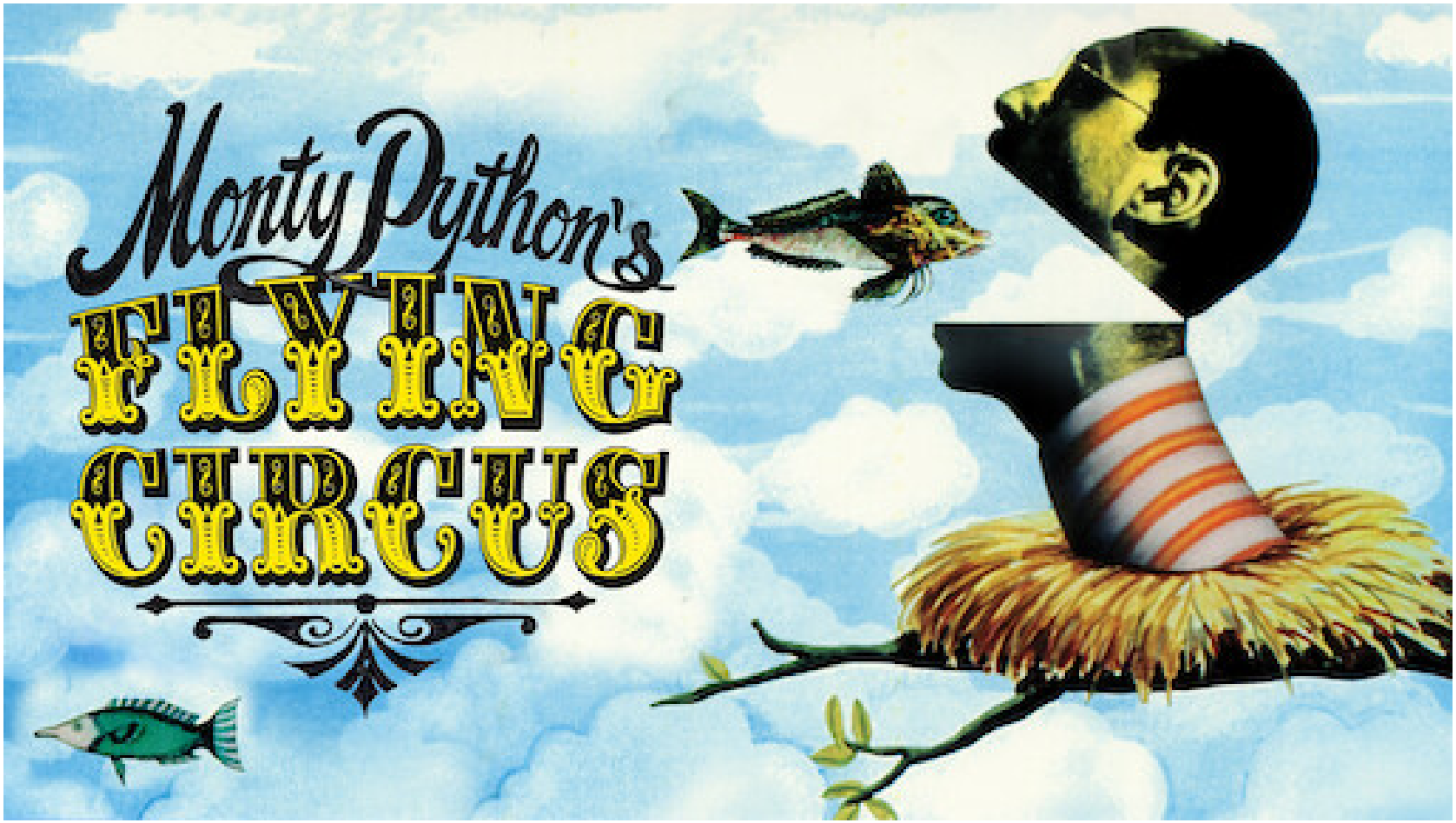
KØBENHAVNS UNIVERSITET

Todays Plan - IFOs

- Let's dig into basic Python and Functions
- Making some Animations
- Exploring Pair Programming

KØBENHAVNS
UNIVERSITET

The Python of Python



What the heck is Python

- A high-level, interpreted programming language known for its simplicity and readability.
- Supports multiple programming paradigms: procedural, object-oriented, and functional.
- Extensive standard libraries and vast ecosystem of third-party packages.
- Used for web development, data analysis, artificial intelligence, scientific computing, and more.
- Guiding principle: "There should be one—and preferably only one—obvious way to do it." (From the Zen of Python)

KØBENHAVNS
UNIVERSITET

Brief History of Python

- Created by Guido van Rossum and first released in 1991.
- Named not after the snake, but after the British comedy series "Monty Python's Flying Circus", which Guido enjoyed.
- Python 2.0 (2000) introduced new features like garbage collection and Unicode support.
- Python 3.0 (2008) was a major overhaul fixing inconsistencies, focusing on removing duplicate constructs and modules.
- Python's popularity Continues growing due to its ease of learning and versatility, with a strong community support and a wealth of libraries and frameworks.

KØBENHAVNS
UNIVERSITET

What are Functions

Basic

- Writing a program is like writing a recipe.
- Similar to how a recipe is a set of steps another person follows, a program is a set of steps the computer follows.
- A single recipe step might be “preheat the oven to 350 degrees” or “add 2 cups of flour”, and you might write each step on its line.
- The other person then follows those steps in order, one after the other, to bake a cake.

```
1 preheat oven to 350 degrees
2 get a large bowl
3 add 2 cups of flour
4 add 1 cup of sugar
5 ...
```

KØBENHAVNS
UNIVERSITET

Functions in Programming

- This is similar to how a computer program works. A program is a set of instructions that tells the computer to follow a series of steps.
- Each step is written on its line, and the computer follows the instructions individually.
- A function is one of those steps. Calling a function gives the computer a single instruction that tells it to do one thing.

KØBENHAVNS
UNIVERSITET

Py. Processing

Examples

```
1  def drawFish(fishX, fishY, eyeSize):
2      # Krop
3      fill(255, 157, 0)
4      ellipse(fishX, fishY, 120, 75)
5
6      # Finner
7      fill(247, 222, 0)
8      triangle(fishX-60, fishY, fishX-90, fishY-30,
           fishX-90, fishY+30)
9      triangle(fishX+10, fishY+10, fishX-20, fishY,
           fishX-20, fishY+25)
10
11     # Eye
12     fill(0, 0, 0)
13     ellipse(fishX + 30, fishY-10, eyeSize, eyeSize)
14     fill(255, 255, 255)
15     ellipse(fishX + 32, fishY-10, 5, 5)
```

KØBENHAVNS UNIVERSITET

Python Syntax

Python is a programming language that follows a simple, clean, and readable format. Here's a basic breakdown:

- **Indentation:** Instead of using braces {} like many other languages, Python uses indentation (spaces or tabs) to denote code blocks. The amount of indentation should be consistent throughout a block.

```
1 def say_hello():
2     print("Hello!")
```

- **Colons:** Statements that introduce a new block, like if, for, def (for defining functions), end with a colon :

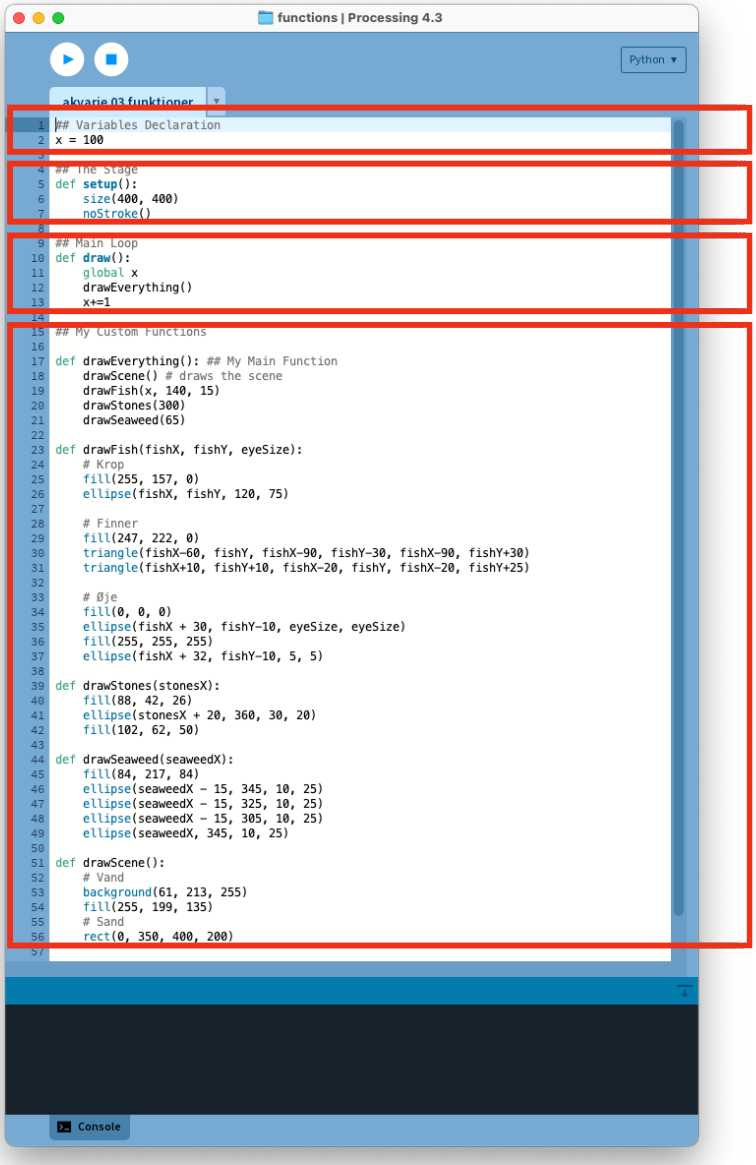
```
1 def drawStones(stonesX):
2     fill(88, 42, 26)
```

- **Comments:** Any line that starts with a # is a comment, meaning Python ignores it. It's for the programmer to write notes.

```
1 # This is a comment
2 print("This is not a comment")
```

KØBENHAVNS UNIVERSITET

Basic Structure for Processing



Declarations e.g. Variables

Canvas, the stage

Main Loop (60fps)

My Functions

KØBENHAVNS UNIVERSITET

Debugging Strategies

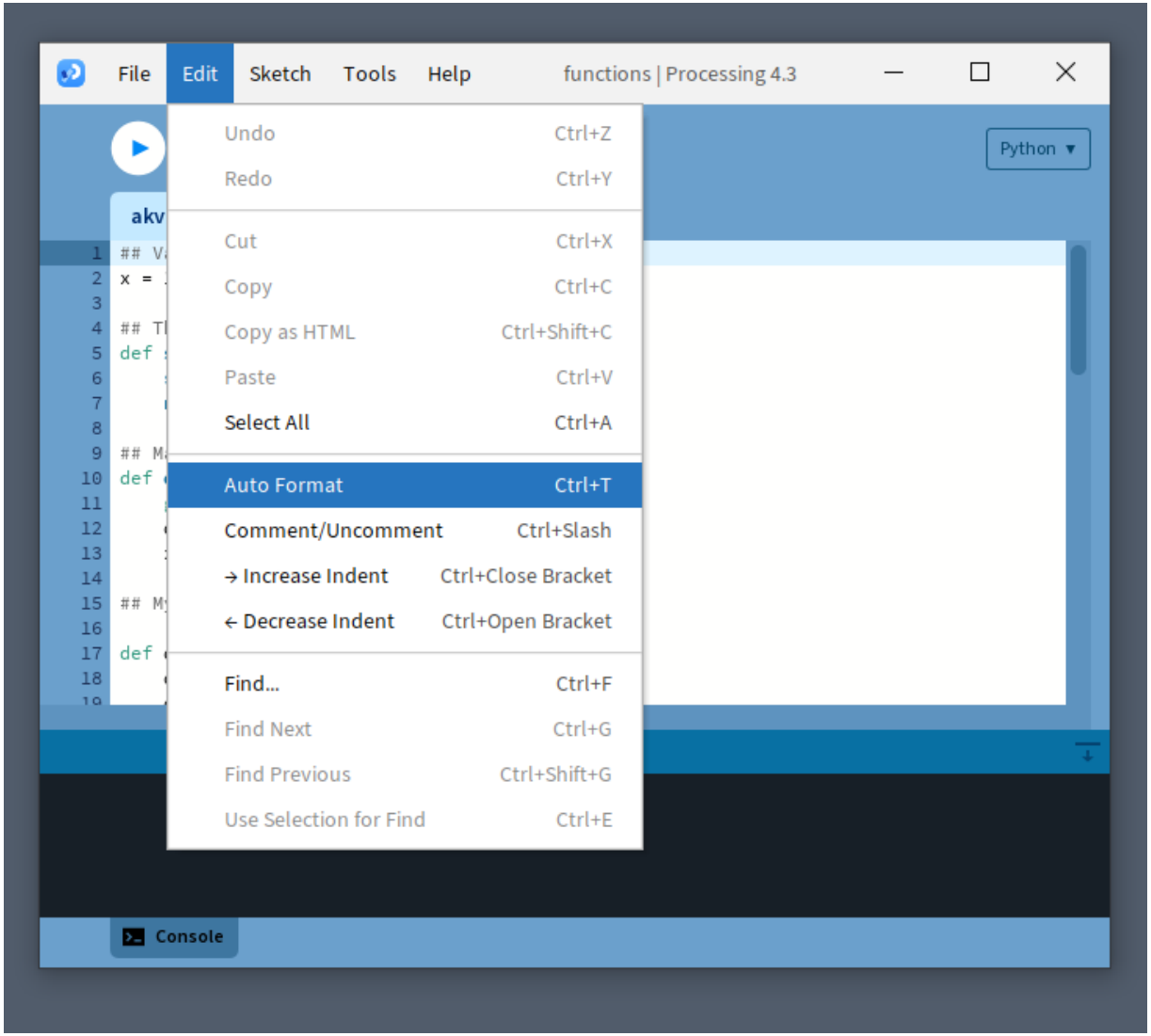
1. **Print Statements**:
   - Sometimes, the simplest methods are the most effective. Use `print()` to display the values of variables, the flow of the program, or to check if a specific part of the code is being executed. This can quickly help you locate where things might be going wrong.
2. **Error Messages**:
   - Always read error messages in the Processing console. They can give you precise information on what went wrong and where. For instance, a `NullPointerException` might indicate you're trying to use an object that hasn't been initialized.
3. **Commenting Out**:
   - If you're unsure which part of your code is causing the problem, try commenting out sections of it to isolate the problematic area. You can gradually uncomment sections to narrow down the issue.
4. **Visual Feedback**:
   - Since Processing is a graphical environment, use visual feedback to your advantage. For example, change colors, draw borders, or use simple shapes to visually represent the flow of logic or the state of specific variables.

Meta Debugging Strategies

1. **Break Down the Problem**:
   - If you have a complex piece of code that isn't working, break it down into smaller, more manageable chunks. Test each chunk independently to ensure it works as expected before integrating it back into the larger codebase.
2. **Check for Common Mistakes**:
   - In the context of Processing, this might mean:
     - Make sure `setup()` and `draw()` functions are correctly defined.
     - Ensuring that you're using the right coordinates or dimensions for shapes.
     - Verifying that image or data files are in the correct directory and are being loaded properly.

KØBENHAVNS
UNIVERSITET

Some Toolsl in the PDE



Pair Programming - Hello Friend!

Two heads are better than one

Pair programming

- Couple of programmers develop higher-quality software
- Slightly longer development time, but much less time spent on find errors afterwards.
- Cozier and more fun than working alone.
- Improved knowledge sharing between the programmers, less time spent on coordination
- Increased learning (and that's why you're here, isn't it?), you take turns a way to teach each other
- Otherwise, unspoken knowledge and habits are exchanged

KØBENHAVNS
UNIVERSITET

Pair programming in practice

Two programmers, one computer

- Driver - has hands on the steering wheel (keyboard) and eyes on the road (screen)
- Navigator - focuses on the destination and how we get there

Rules

- You must not be bossy towards your partner
- The navigator must not touch the mouse or keyboard
- The driver must not ignore the navigator
- You must switch roles often (e.g. every 20 minutes)
- Try to keep a conversation going

KØBENHAVNS
UNIVERSITET

Conversations between driver and navigator

- Good pair programming is not without communication and talk.
- Talk together as a pair about what is happening on the screen.
- Reflect on what you have done and where you are going.
- The driver tells what they are doing and what is happening.
- The navigator makes comments, ensures they are running with the right thing and tells what they have to do now and later.

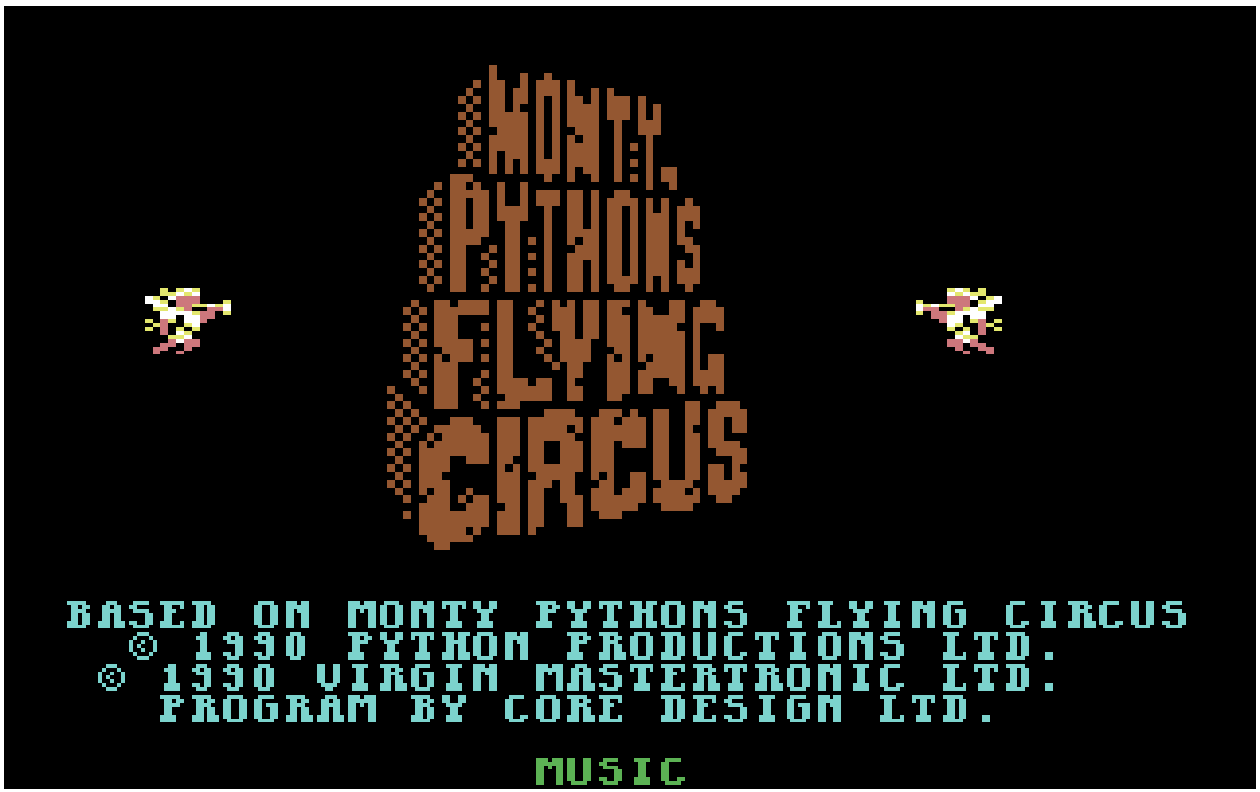
Examples:

- Driver: "Now we create a new function to draw a sunflower."
- Driver: "Now we just test whether XYZ works before we continue."
- Navigator: "How can you do that?"
- Navigator: "Can you explain what you're doing?"

KØBENHAVNS UNIVERSITET

Recap

- Python Syntax and Structure
- Functions and Animations
- Debugging
- Pair Programming



KØBENHAVNS UNIVERSITET

Wednesday

- Conditionals
- more input with keys and mouse
- Finite State Machines?
- Brainstorming Ideas for your Projects!