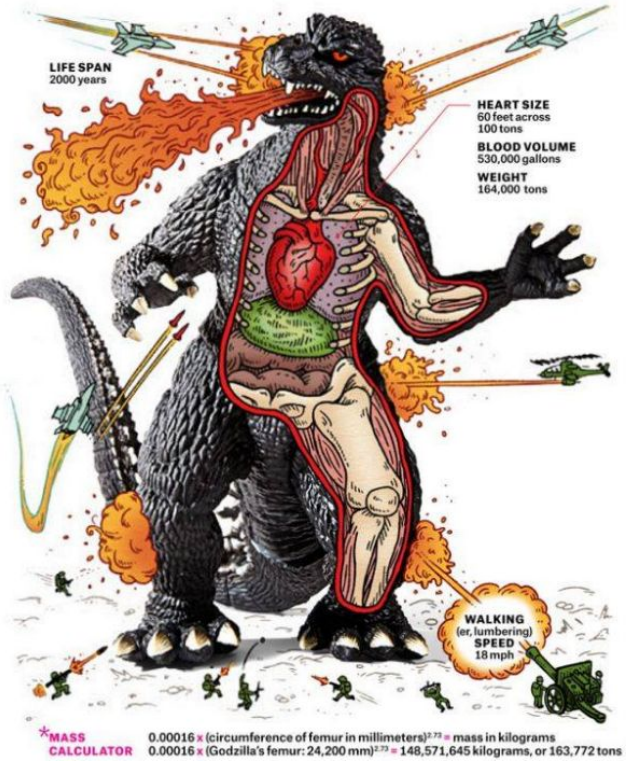
Kickstart-kursus i programmering 23 dag 3



Daniel Spikol
ds@di.ku.dk

DIKU
Københavns Universitet

16 august 2023

Recap from Tuesday

- Python and Functions
- Animations
- Pair Programming
- Mouse and Key Input

KØBENHAVNS
UNIVERSITET

Wednesday IFOs

- Variable Scoping
- Draw() function
- Conditionals
- Ideation

KØBENHAVNS
UNIVERSITET

Variable Scoping Py.Processing

- The principles of variable scoping in Processing.py largely follow those of Python but with some considerations given the environment of Processing's draw loop and function setup.
- Scoping in programming refers to the region or portion of the code where a variable or function is defined and can be accessed or modified.
- The concept of scoping is crucial for understanding variable lifetimes, visibility, and potential naming conflicts in your programs. It helps manage and organize data and functionality, enabling modular and maintainable code designs.

KØBENHAVNS
UNIVERSITET

Variable Scoping Py.Processing

Global Variables

Global Variables: Variables declared outside of any function are global to the sketch. They can be accessed and modified from any function, but if you want to modify them inside a function, you must declare them as `global` within that function.

```
1  #global in action
2  x = 10
3
4  def changeX():
5      global x
6      x = 20
```

KØBENHAVNS UNIVERSITET

Variable Scoping Py.Processing

Local Variables

Local Variables: Variables declared inside a function are local to that function. They cannot be accessed outside of the function, and their memory is reclaimed once the function execution is complete.

```
1  def showValue():
2      y = 15
3      print(y)  # This will print 15
4
5  showValue()
6  # print(y)  # This would be an error because y is not defined
       outside of the function.
```

KØBENHAVNS UNIVERSITET

Variable Scoping Py.Processing

The setup() and draw() Functions: In Processing.py, `setup()` is called once at the beginning of the sketch, and `draw()` is called repeatedly, producing frames.

```
1  x = 0
2
3  def setup():
4      global x
5      size(400, 400)
6      x = width / 2  # Initializing x based on the canvas width
7
8  def draw():
9      global x
10     background(240)
11     ellipse(x, height/2, 50, 50)
12     x += 1
```

Variables declared in `setup()` are local to `setup()`. Still, often you want to declare global variables at the top level of your sketch and then initialise or modify them in `setup()`.

KØBENHAVNS UNIVERSITET

Conditionals in Computer Science

- Conditionals in computer science refer to constructs that allow for decision-making in code.
- They determine the flow of execution based on whether a given condition is true or false.
- Depending on the outcome of the condition, different blocks of code will be executed.
- Specifically, conditionals perform different computations or actions depending on whether a programmer-defined boolean condition evaluates to true or false.

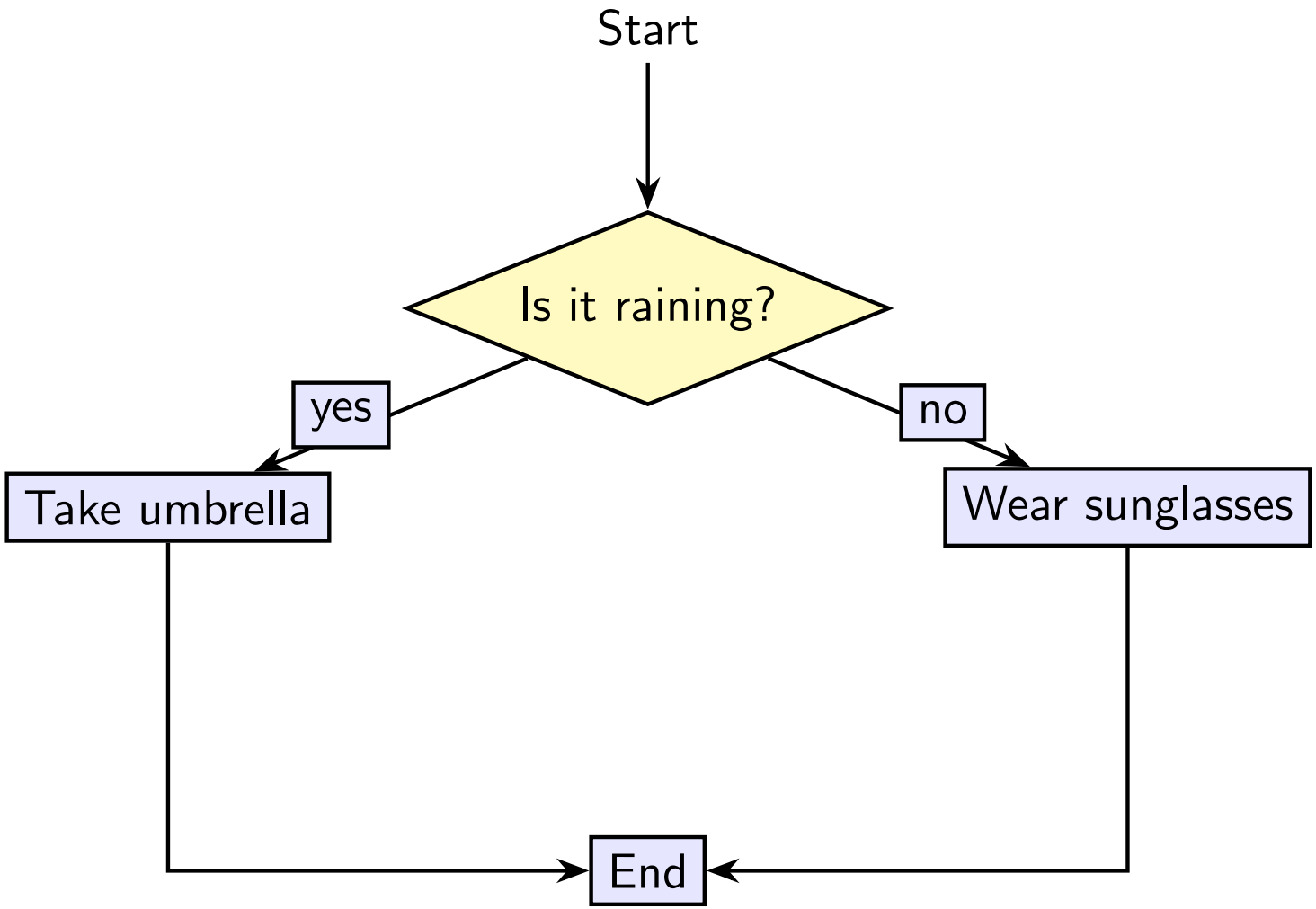
KØBENHAVNS
UNIVERSITET

Conditionals Conceptually

- **Basic Conditional (IF)** *Concept*: If a specific condition is true, do something. *Example*: "If it's raining, take an umbrella."
- **Alternative Path (ELSE)** *Concept*: If the first condition isn't met, do something else instead. *Example*: "If it's raining, take an umbrella. Otherwise, wear sunglasses."
- **Multiple Conditions (ELSE IF or ELIF)** *Concept*: Check multiple conditions in sequence, and do the first thing that's true. *Example*: "If it's raining, take an umbrella. If it's sunny, wear sunglasses. Otherwise, just go outside as usual."
- **Combining Conditions** *Concept*: You can use logical operators (AND, OR, NOT) to combine conditions. *Example*: "If it's a weekend AND the weather is good, go hiking."

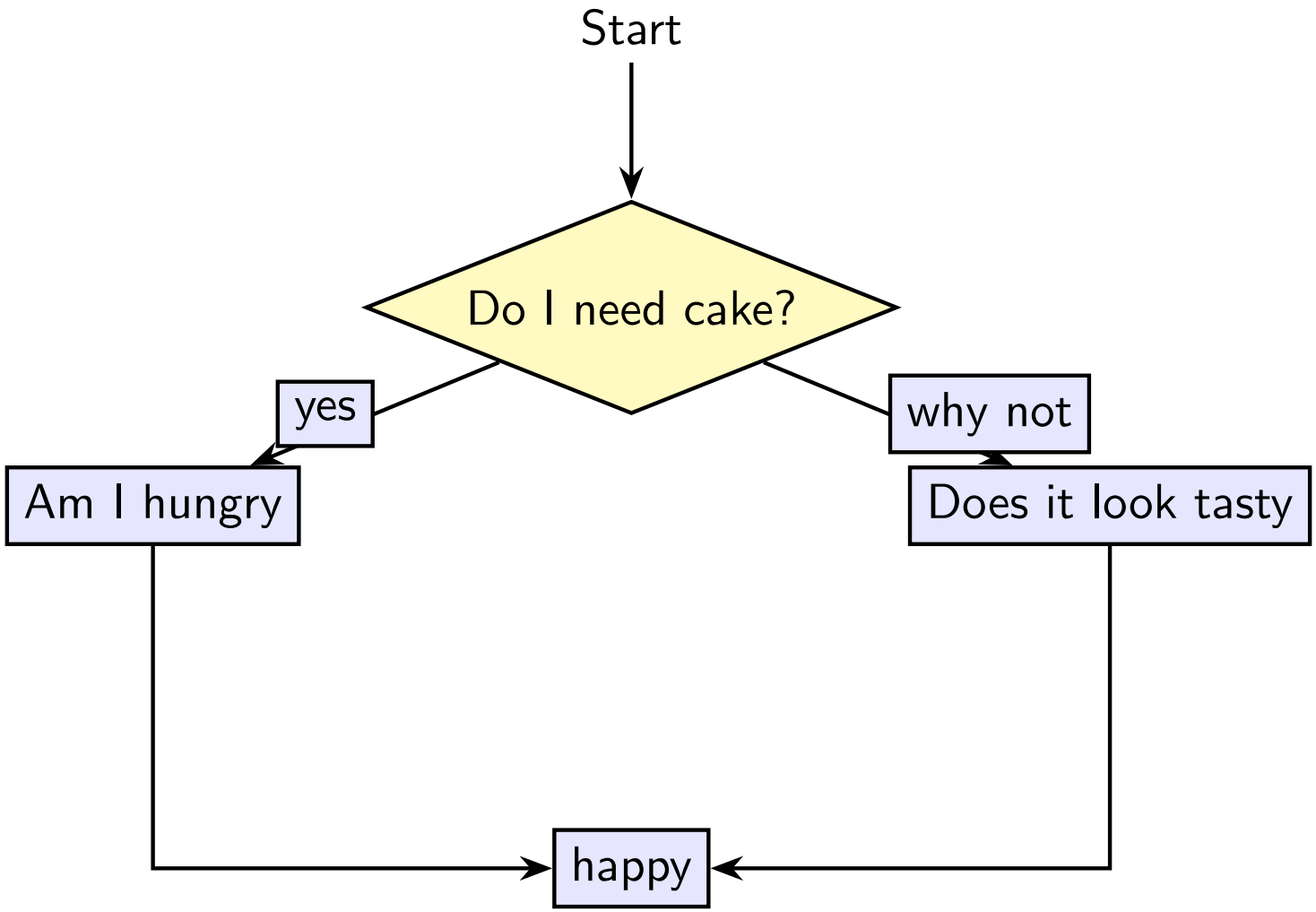
KØBENHAVNS
UNIVERSITET

UML Condition



KØBENHAVNS UNIVERSITET

Cake Condition



Conditionals

- **If Statement:** Executes a code block if a specified condition is true.
- **Else Statement:** Used in conjunction with an if statement, it specifies a block of code to be executed if the condition in the if statement is false.
- **Else If Statement:** Used to specify a new condition to test if the first condition is false.
- **Switch or Case Statement:** Allows a variable to be tested for equality against a list of values.

KØBENHAVNS
UNIVERSITET

Conditionals in Python

Python supports the usual logical conditions from mathematics:

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

KØBENHAVNS
UNIVERSITET

Conditionals in Python

`if:`

```
if x > 10:
    print("x is greater than 10")
```

`elif (else if):`

```
if x > 10:
    print("x is greater than 10")
elif x == 10:
    print("x is 10")
```

`else:`

```
if x > 10:
    print("x is greater than 10")
else:
    print("x is 10 or less")
```

You can also combine conditions using logical operators (`and`, `or`, `not`):

```
if x > 10 and y < 5:
    print("x is greater than 10 and y is less than 5")
```

KØBENHAVNS UNIVERSITET

Conditionals in Py.Processing

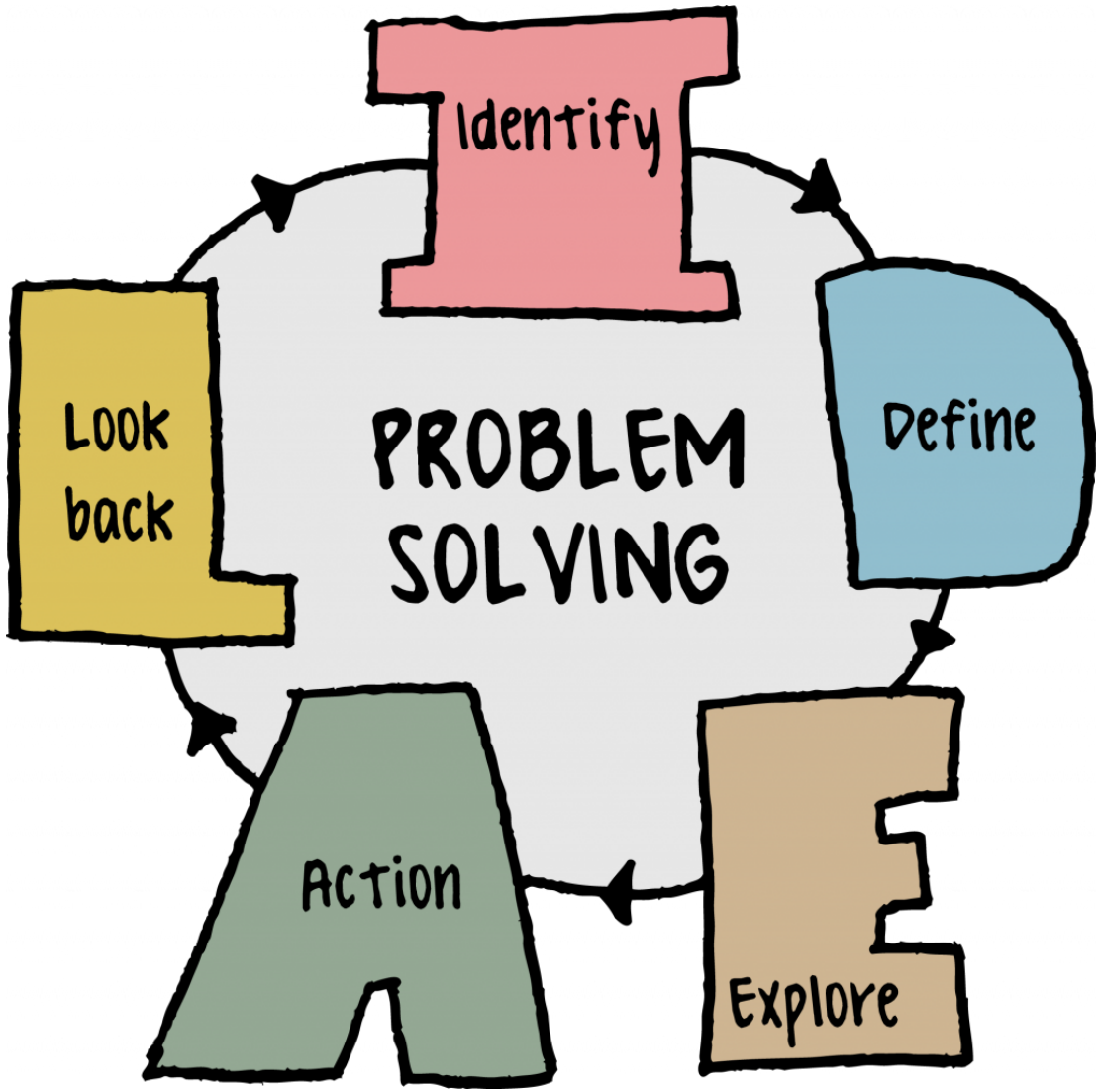
For instance, to animate a circle moving across the screen and to make it wrap around when it reaches the edge:

```
x_pos = 0

def setup():
    size(400, 400)

def draw():
    global x_pos
    background(240)
    ellipse(x_pos, height/2, 50, 50)
    x_pos += 2

    # Conditional to check if the circle is out of bounds
    if x_pos > width:
        x_pos = 0
```

In the code above, the conditional if `if x_pos > width:` checks if the circle has moved outside the canvas. If true, it resets its position.

KØBENHAVNS UNIVERSITET

IDEAL Problem Solving



KØBENHAVNS UNIVERSITET

Today' Recap

- Conditionals
- Better Animations
- Pair Programming
- Ideas for Projects

KØBENHAVNS
UNIVERSITET

Tomorrow

- Finite State Machines
- Project Work
- Coding

KØBENHAVNS
UNIVERSITET