Lecture 1
Introduction to Variables and Control Statements

Jan Tekülve
jan.tekuelve@ini.rub.de

Computer Science and Mathematics
Preparatory Course

19.09.2019
Course Formalities

Goals:

▶ Learning basic programming with Python
▶ Refreshing elementary mathematical concepts

Concept:

▶ Each lecture will usually be split into a theoretical explanation and a programming session
▶ On the last day (02.10.) there will be an ungraded “test”
Overview

1. Motivation

2. Programming
   - First Steps
   - Variables
   - Control Statements
   - Utilities

3. Tasks
Motivation: Modeling a cognitive agent

Braitenberg Vehicles

[Braitenberg, 1986]
Motivation: Modeling a cognitive agent

Environmental Factors (Numbers)

Braitenberg Vehicles

[Braitenberg, 1986]
Motivation: Modeling a cognitive agent

Environmental Factors (Numbers)

Braitenberg Vehicles

[Braitenberg, 1986]
Motivation: Modeling a cognitive agent

Environmental Factors
(Numbers)

Distance and Orientation
(Trigonometry)

Relationships
(Functions)

Braitenberg Vehicles

[Braitenberg, 1986]
Motivation: Modeling a cognitive agent

Environmental Factors (Numbers)
Relationships (Functions)
Distance and Orientation (Trigonometry)
Velocity and Position (Differentiation and Integration)

Braitenberg Vehicles

[19.09.2019 4 / 28]
Motivation: Modeling a cognitive agent

Environmental Factors
(Numbers)
Relationships
(Functions)
α
Distance and
Orientation
(Trigonometry)
Velocity and
Position
(Differentiation
and Integration)
Behavior (Differential Equations)

Braitenberg Vehicles

[Braitenberg, 1986]
Motivation: Modeling a cognitive agent

Environmental Factors (Numbers)

Velocity and Position (Differentiation and Integration)

Distance and Orientation (Trigonometry)

Connections (Matrices)

Relationships (Functions)

Behavior (Differential Equations)

Braitenberg Vehicles

[Braitenberg, 1986]
Programming Goal
# Course Structure

<table>
<thead>
<tr>
<th>#</th>
<th>Date</th>
<th>Title</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19.09.</td>
<td>Variables and Control Statements</td>
<td>Data Types, Control Statements</td>
</tr>
<tr>
<td>2</td>
<td>20.09.</td>
<td>Functions in Math and Programming</td>
<td>Function Types and Properties, Plotting Functions, Lists</td>
</tr>
<tr>
<td>3</td>
<td>23.09.</td>
<td>Full-Time Programming Session</td>
<td>Deepen Programming Skills</td>
</tr>
</tbody>
</table>
# Course Structure

<table>
<thead>
<tr>
<th>#</th>
<th>Date</th>
<th>Title</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>26.09.</td>
<td>Integration</td>
<td>Geometrical Definition, Calculating Integrals, Numerical Integration</td>
</tr>
<tr>
<td>7</td>
<td>27.09.</td>
<td>Differential Equations</td>
<td>Properties of Differential Equations, Euler Approximation, Braitenberg Vehicle</td>
</tr>
<tr>
<td>8</td>
<td>30.09.</td>
<td>Matrices</td>
<td>Matrix Addition, Matrix Multiplication, Basic Neural Networks</td>
</tr>
<tr>
<td>9</td>
<td>01.10.</td>
<td>“Make a wish Lecture”</td>
<td>Individual Wishes, e.g. Object-Oriented Programming</td>
</tr>
<tr>
<td>10</td>
<td>02.10.</td>
<td>“Test”</td>
<td>Repetition of Core Concepts and Ungraded Test</td>
</tr>
</tbody>
</table>
1. Motivation

2. Programming
   - First Steps
   - Variables
   - Control Statements
   - Utilities

3. Tasks
The Python Programming Language

Why Python?

- It is simple but high level
- It is interpreted “on the fly”
- It is the state of the art scripting language

Helpful Resources

- The Anaconda Distribution contains all necessary software: https://www.anaconda.com/distribution/
- You can find helpful documentation here: https://docs.python.org/3/
Setting Up

- Open the *Spyder* IDE (Integrated Development Environment)

- Choose a working directory
  - Select the folder symbol in the top right corner
  - Navigate to your desired directory
  - *(Recommended)* Create a new folder for your python projects

- Create your first python script file
  - Close the default temporary file
  - Go to *File → Save as* …
  - Choose the name *helloworld*

- You are set up to write your first Python script!
**Hello World**

▶ Write the following line into the file:

```python
print("Hello World!")
```

▶ Press the green *Play* button in the toolbar to execute the script

▶ Observe the output in the console on the right
Hello World

- Write the following line into the file:
  ```python
  print("Hello World!")
  ```
- Press the green *Play* button in the toolbar to execute the script
- Observe the output in the console on the right
- The `print()` function writes its argument to the console
Script: A series of commands

- Code is executed from top to bottom - one line after each other

```python
print("Hello There!")
print("Haven't seen you in a while.")
print("How are you?")
```

You can write comments in your code using the # character

```python
print("Hello!") #This is a comment
print("How are you?") #print("I am bored") This line is ignored
```
Script: A series of commands

- Code is executed from top to bottom - one line after each other

```python
print("Hello There!")
print("Haven't seen you in a while.")
print("How are you?")
```

- You can write comments in your code using the # character

```python
print("Hello!") # This is a comment
# Lines that start with # are ignored
print("How are you?")
# print("I am bored") This line is ignored
```
Variables

Variables are the elementary building block of every program

```python
>>> greeting = "Hello, Hello!"
>>> print(greeting) #prints "Hello, Hello!"
```

Variables are assigned via `=`

```python
>>> var1 = "Hello" #variable names may be chosen arbitrarily
>>> long_variable_name5 = "Hi"
>>> 
>>> #letters, numbers and underscores may make up a name

>>> print(greeting) #prints "Hello, Hello!"
>>> greeting = "Hey!" #variables may be overwritten
>>> print(greeting) #prints "Hey!"
```
Variables

- Variables are the elementary building block of every program

```python
# greeting is assigned
print(greeting)  # prints "Hello, Hello!"
```

- Variables are assigned via `=`

```python
# variable names may be chosen arbitrarily
var1 = "Hello"  # variable names may be chosen arbitrarily
long_variable_name5 = "Hi"

# letters, numbers and underscores may make up a name
```
Variables

- Variables are the elementary building block of every program

```python
# greeting variable assignment
greeting = "Hello, Hello!"
print(greeting) # prints "Hello, Hello!"
```

- Variables are assigned via ‘=’

```python
# variable assignment example
var1 = "Hello" # variable names may be chosen arbitrarily
long_variable_name5 = "Hi"
# letters, numbers and underscores may make up a name
```

- Assigned variables are available for code following the assignment

```python
# code following the variable assignment
print(greeting) # prints "Hello, Hello!"
greeting = "Hey!" # variables may be overwritten
print(greeting) # prints "Hey!"
```
Data Types and Operations

- Variables store information of various type:

  farewell = "Bye, Bye!" # String Type
  num1 = 5 # Integer Type
  num2 = 3.0 # Float Type

Operations may be performed using variables:

  print(num1 + num2) # prints 8.0

Results may again be stored in variables:

  num3 = num1 + num2 # num3 is now 8.0
  print(num3) # prints 8.0
  num3 = num3 + 1 # num3 updates based on its current value
  print(num3) # prints 9.0
Data Types and Operations

- Variables store information of various type:

```python
farewell = "Bye, Bye!"  # String Type
num1 = 5  # Integer Type
num2 = 3.0  # Float Type
```

- Operations may be performed using variables

```python
print(num1+num2)  # prints 8.0
```
Data Types and Operations

- Variables store information of various type:
  - `farewell = "Bye, Bye!"` #String Type
  - `num1 = 5` # Integer Type
  - `num2 = 3.0` # Float Type

- Operations may be performed using variables
  - `print(num1+num2)` #prints 8.0

- Results may again be stored in variables
  - `num3 = num1+num2` #num3 is now 8.0
  - `print(num3)` #prints 8.0
  - `num3 = num3+1` #num3 updates based on its current value
  - `print(num3)` #prints 9.0
Excursion: The Spyder Debugger

A debugger allows a look under the ‘hood’ of a program

These are the Debug Controls

- Start Debugging
- Execute Line by Line
- Stop Debugging

Click here to display the current variables
Useful Operations on Data Types

Operations on Numbers

2+2 #4
50-5*6 #20
(50-5*6)/4 #5.0
8/5 #1.6
17/3 #5.666666666666667
17//3 #5 Integer Division
17%3 #2 Rest of the Division

Operations on Strings

'Wo' + 'rd' #'Word' or "Word"
'Isn't' # This results in an error!
'Isn\'t' #'Isn't' Use \ to escape characters
Control Statements

▶ if-Statement

```python
x = 3.5
if x > 0 : #Indentation organizes blocks
    print("x is positive!") #Indent with 4 spaces
print("Program is finished!")
```

▶ else-statement

```python
x = 3.5
if x > 0 : #Indentation organizes blocks
    print("x is positive!") #Indent with 4 spaces
else :
    print("x is not positive!")
print("Program is finished!")
```
Control Statements

▶ if-Statement

```python
x = 3.5
if x > 0:  # Indentation organizes blocks
    print("x is positive!")  # Indent with 4 spaces
print("Program is finished!")
```

▶ else-statement

```python
x = 3.5
if x > 0:  # Indentation organizes blocks
    print("x is positive!")  # Indent with 4 spaces
else:
    print("x is not positive!")
print("Program is finished!")
```
Control Statements

- else if-statement

```python
x = 3.5
if x > 0:  # Indentation organizes blocks
    print("x is positive!")  # Indent with 4 spaces
elif x < 0:
    print("x is negative!")
else:
    print("x is zero!")
print("Program is finished!")
```

```
Variable Scope

- Python code is organized in blocks by indentation (4 spaces)

```python
a = 3
b = 4
if a > 2:
c = a + b
    b = 1
if c > 5:
    print(a)
else:
    print(a)
    print(c)
print(b)
```
Variable Scope

- Python code is organized in blocks by indentation (4 spaces)

- Variables defined in the global scope are available at all positions in the code below its definition

```python
# Global
a = 3
b = 4
if a > 2:
c = a + b
b = 1
if c > 5:
    print(a)
else:
    print(a)
    print(c)
print(b)
```
Variable Scope

- Python code is organized in blocks by indentation (4 spaces)
- Variables defined in the global scope are available at all positions in the code below its definition
- Variables defined in a block are available in the block and all blocks inside it
Variable Scope

Example

```python
a = 3  # Global Scope
b = 4
if a > 2 :
    c = a + b  # Block 1
    b = 1
    if c > 5:
        print(a)  # Block 2
else :  # Global
    print(a)  # Block 3
    print(c)  # If a <= 2 this will result in an error
print(b)  # '1' or '4' if a <= 2
```
While Loops

- Print the numbers from 1 to 10

```python
a = 0
while a < 10 :
    a = a +1 # Increase a by 1
    print(a)
```
**While Loops**

- Print the numbers from 1 to 10

```python
a = 0
while a < 10 :
    a = a +1  # Increase a by 1
    print(a)
```

- Be careful with the exit condition

```python
a = 0
while a < 10 :
    print(a)  # Prints 0 until the end of time
```

You can kill the running program by pressing the red terminate button.
Boolean Statements

Examples

3 > 2 # True, greater than
3 < 3 # False, less than
3 <= 3 # True, equal or less than
4 == 5 # False, == checks equality
4 != 5 # True, != is the opposite of ==
"ello" in "Hello" # True, only works for sequence types
"hel" not in "Hello" # True, "in" is case sensitive
Boolean Statements

Examples

3 > 2 # True, greater than
3 < 3 # False, less than
3 <= 3 # True, equal or less than
4 == 5 # False, == checks equality
4 != 5 # True, != is the opposite of ==
"ello" in "Hello" # True, only works for sequence types
"hel" not in "Hello" # True, "in" is case sensitive

Boolean Variables

test = 7
isGreaterThanOne = test > 1
if isGreaterThanOne:
    print("The number is Greater than 1!")
User Input

▶ Use input to prompt the user

```python
person = input('Enter your name: ')
print('Hello ' + person)
```
User Input

▶ Use input to prompt the user

```python
person = input('Enter your name: ')
print('Hello ' + person)
```

▶ Invalid Data Types

```python
inputValue = input('Please enter a number: ')
result = 5 + inputValue # This results in an error!
```
User Input

Use input to prompt the user

```python
person = input('Enter your name: ')
print('Hello ' + person)
```

Invalid Data Types

```python
inputValue = input('Please enter a number: ')
result = 5 + inputValue # This results in an error!
```

Variables might need to be type casted

```python
result = 5 + float(inputValue)
#This works if an actual number was typed
```
Type Casting

▶ Implicit Typecast

a = 1.0 #float
b = 2 #int
c = a + b #3.0 float
Type Casting

- **Implicit Typecast**

  ```
  a = 1.0 #float
  b = 2 #int
  c = a + b #3.0 float
  ```

- **Explicit Typecasts**

  ```
  d = float(b) #2.0
  e = 3.7
  f = int(3.7) #3 Any floating point is cut off
  g = str(e) #String '3.7'
  h = int(g) # This results in an error!
  i = float(g) # 3.7
  print('Variable i is: ' + str(i)) #Print expects strings
  ```
Useful built-in Functions

▶ Rounding and Absolute Value

\[
a = 3.898987897897 \\
b = \text{round}(a,3) \ #3.899 \\
c = \text{abs}(-3.2) \ #|-3.2| = 3.2 \\
t = \text{type}(c) \ #t \text{ is <class 'float'>} \\
test = t \text{ is float} \ # True
\]

▶ The math module

```python
import math \ #Import makes a module available
squareTwo = math.sqrt(2) \ #\sqrt{2} \\
power = math.pow(3,4) \ # 3^4 \\
exponential = math.exp(4) \ #e^4 \\
piNumber = math.pi \ #3.14159265359
```
Lecture Slides/Material

Use the following URL to access the lecture slides:

https://www.ini.rub.de/teaching/courses/c_science_math_2019
Tasks

1. Write a script that takes a percentage and prints out the corresponding verbal grade.
   ▶ Define a variable `perc` and assign it a number between 1 and 100.
   ▶ Use If and Else to print out the correct grade depending on the value of `perc`.

<table>
<thead>
<tr>
<th>%</th>
<th>Grade</th>
<th>%</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>86-100</td>
<td>A</td>
<td>40-55</td>
<td>D</td>
</tr>
<tr>
<td>71-85</td>
<td>B</td>
<td>25-39</td>
<td>E</td>
</tr>
<tr>
<td>56-70</td>
<td>C</td>
<td>1-24</td>
<td>F</td>
</tr>
</tbody>
</table>

2. Write a script that asks the user for two different inputs and prints their sum
   ▶ Define a variable `num1` and assign it a value using the `input()` function
   ▶ Repeat the above step for a second variable `num2`
   ▶ Add `num1` and `num2` together in a third variable `sum` and print it (Do not forget to typecast `num1` and `num2`)
Tasks Continued

3. Write a script that asks the user for number input until the sum of the inputs is greater than 20.
   - Start with a variable `sum` that is initialized with the value 0.
   - Create a while-loop that ends when `sum` is greater than 20.
   - Inside the while-loop ask the user for input and add the input to `sum`.
     (Do not forget to typecast the input)

4*. Write a script that finds the maximum number out of 3 numbers.
   - Example:
     You choose the three numbers to be 13, 16 and 5.
     The program should print: “The highest number is 16”.
   - Define three variables each containing a different number.
   - Use If and Else statements to find the highest of the three numbers.
   - Print the number to the console.
   - The script should work for any three numbers.
References