Modul - Introduction to AI (AI1)

Bachelor Programme AAI

03 - Python (PART II)

Prof. Dr. Marcel Tilly

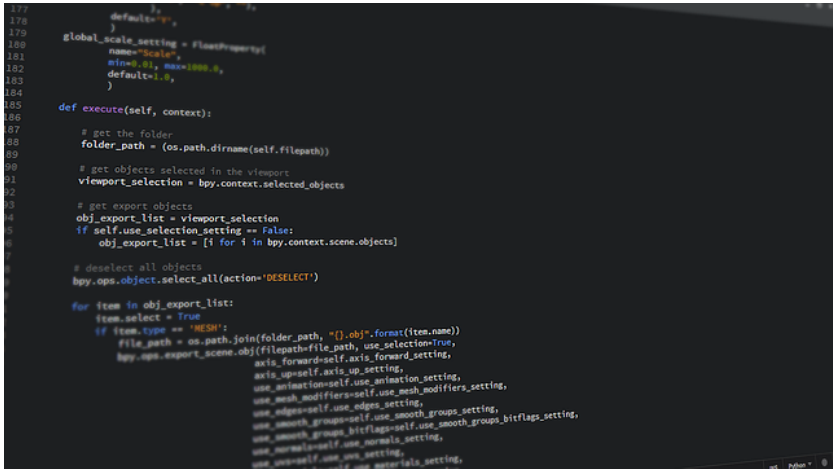
Faculty of Computer Science, Cloud Computing

Vers. 8.11.2021

Agenda

On the menu for today:

- Python
  - Sequences, lists, dict
  - files
  - functions
  - comprehensions



```
                default='Y',
                )
    global_scale_setting = FloatProperty(
                name="Scale",
                min=0.01, max=1000.0,
                default=1.0,
                )

    def execute(self, context):

        # get the folder
        folder_path = (os.path.dirname(self.filepath))

        # get objects selected in the viewport
        viewport_selection = bpy.context.selected_objects

        # get export objects
        obj_export_list = viewport_selection
        if self.use_selection_setting == False:
            obj_export_list = [i for i in bpy.context.scene.objects]

        # deselect all objects
        bpy.ops.object.select_all(action='DESELECT')

        for item in obj_export_list:
            item.select = True
            if item.type == 'MESH':
                file_path = os.path.join(folder_path, "{}.obj".format(item.name))
                bpy.ops.export_scene.obj(filepath=file_path, use_selection=True,
                                        axis_forward=self.axis_forward_setting,
                                        axis_up=self.axis_up_setting,
                                        use_animation=self.use_animation_setting,
                                        use_mesh_modifiers=self.use_mesh_modifiers_setting,
                                        use_edges=self.use_edges_setting,
                                        use_smooth_groups=self.use_smooth_groups_setting,
                                        use_smooth_groups_bitflags=self.use_smooth_groups_bitflags_setting,
                                        use_normals=self.use_normals_setting,
                                        use_uvs=self.use_uvs_setting,
                                        [illegible]materials_setting,
```

Sequence types

```
# list  is  mutable
l = list()
print(l)
l = [1, "2", list()]
print(l)
l[0] = 9
l.append(10)
print(l)

#  tuple  ist  immutable
t = tuple()
t = ("value", 1)
print(t)
t[0] = t[1]
# error

#  range  ist  immutable
r = range(0,4,1)
# das  selbe  wie  range (4)
print(r)
print(list(r))
```

https://repl.it/@marceltilly/TH-Rosenheim#lecture/03-sequences.py

Assignments

No copies are made for assignments!

```
l = [0, 4, 42]
k = l
k[1] = 7
print(k)
print(l)

# int , float , str  sind immutable!
i = 42
j = i
j += 1
#  creates  '43' ad an object  and makes the assigment for j
print(j)
print(i)
```

https://repl.it/@marceltilly/TH-Rosenheim#lecture/04-zuweisungen.py

Set type und dictionary type

- **set** : Sets with unique elements
- **dict** : Key-Value Pair Store

```
# set
s = set()
print(s)
s = set([1 ,2 ,3])
print(s)
print(s == set([1,2,2,1,3]))

#  dictionary
d = dict()
d = {'key': 'value'}
d = {'Paris': 2.5, 'Berlin': 3.4}
print ('Inhabitants  Paris:', d['Paris'], 'Mio')
print ('Inhabitants  Berlin:', d['Berlin'], 'Mio')
```

https://repl.it/@marceltilly/TH-Rosenheim#lecture/05-set.py

Control Structures 101

Technische Hochschule Rosenheim

IF

```python
#  if
condition = "" or (3 - 3)
if condition:
    print("Condition  is true!")
elif 1 == 2:
    print("1 is  equal to 2!")
else:
    print("Nothing  true  here :(")
```

https://repl.it/@marceltilly/TH-Rosenheim#lecture/06-if.py

Control Structures 101

Loops

```python
# while -Schleife
a = 0
while a < 5:
    a+=1
    print(a)

# for -Schleife
for i in range(0,4,1):
    print(i)
```

https://repl.it/@marceltilly/TH-Rosenheim#lecture/07-while.py

Control Structures 101

- Iterator:

```
for <variable> in <iterable object>:
    <operations>
```

In a for loop, it is possible to iterate over *sequence types* and *dictionary types*, for example.

The statements or statement blocks that logically belong together must be indented to the same depth.

Examples

```
for i in range(5):
    print(i)
```

Control Structures 101

```
for i in range(5):
    if i == 2:
        continue     # next iteration
    print(i)
    if i == 3:
        break       #  interrupt loop

# enumerate  all elements  of an iterables
for i, value in enumerate(range(0 ,10 ,2)):
    print('the', i, 'th number is', value)

# loop of  key/value-pairs  of dicts
dic = { 'Paris ': 2.5, 'Berlin ': 3.7, 'Moskau ': 11.5 }
for key , value in dic.items ():
    print(key + ':', value , 'Mio  inhabitants ')
```

https://repl.it/@marceltilly/TH-Rosenheim#lecture/08-for.py

Reading and writing files

In Python, files can be easily opened, written and read. Python provides the following functions provides the following functions:

- **open( )**: Open a file
- **write( )**: Write individual strings to file
- **writelines( )**: Write list of strings to file
- **readlines( )**: Reads lines of a file into list of strings
- **readline( )**: Reads single line of file into string
- **read( )**: Reads all lines of a file into a string

Reading and writing files

Furthermore, an additional attribute can be passed to the open( ) command that regulates the type of access to a file:

- **r**: Open for reading (default)
- **w**: Open for writing - implies overwriting
- **a**: Open to write at the end of the file
- **r+**: Open to read and write at the beginning of the file
- **w+**: Open for reading and writing, file contents previously deleted.
- **a+**: Open for reading and writing at the end of the file

Reading and writing files

```python
#  open file
d = open("sample.txt","r+")
#  read
contents = d.read()
if contents  != "":
    print(contents)
else:
    print("File is empty!\n\n")

#  user input
text = input("Input at the end: ")

#  write to file with line feed
d.write(text+"\n")
#  do not forget to close file
d.close()
```

https://repl.it/@marceltilly/TH-Rosenheim#lecture/09.read.py

Functions

Technische Hochschule Rosenheim

- Functions are used to structure statements, which can then be conveniently executed any number of times by calling the function. The function can receive input arguments and return objects itself.
- Functions in Python are called with the keyword `def`, the function name and the passed parameter list as follows:

```python
def function_name(a,b,...):
    <operations>

#  Define a function to sum numbers
def sum(a, b):
    return a+b

result = sum (1,2)
print(result)
```

https://repl.it/@marceltilly/TH-Rosenheim#lecture/10-functions.py

Exercise 2

Write your second Python programm

Given two numbers, write a Python function which gets 2 numbers and outputs the result:

Examples:

```
Input: a = 2,b = 4

Output: b is greater

Input: a = -1,b = -4

Output: a is greater
```

Summary

Lessons learned today:

- Python Basics
  - lists, dics
  - etc.
- more to come...

Homework

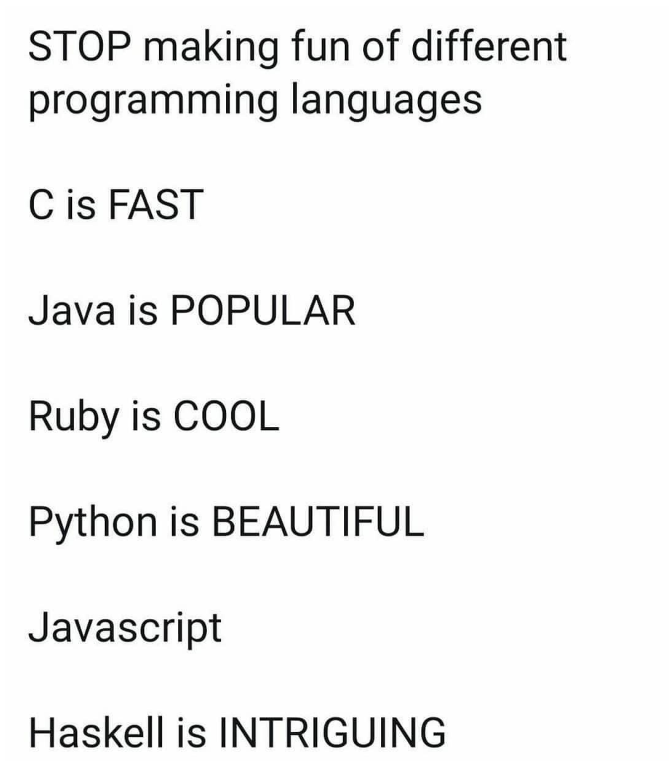
Solve the Padlock challenges 1-3:

1) https://www.101computing.net/padlock-code-challenge-1/

2) https://www.101computing.net/padlock-code-challenge-2/

3) https://www.101computing.net/padlock-code-challenge-3/

Final remark



STOP making fun of different programming languages

C is FAST

Java is POPULAR

Ruby is COOL

Python is BEAUTIFUL

Javascript

Haskell is INTRIGUING