2 Multimedia Programming with Python and Pygame

2.1 Introduction to Python

2.2 Pygame:
Multimedia/Game Frameworks for Python

2.3 SDL: Background of Pygame

Literature:
G. van Rossum and F. L. Drake, Jr., An Introduction to Python - The Python Tutorial (version 3.2), Network Theory 2011
• Guido van Rossum, 1991, CWI Amsterdam
• Now open source, current main versions:
  – 2.7.11 and 3.5.1
• Targeted at programming novices
• Characteristics:
  – Interpreted scripting language
  – Compiled to intermediate byte code (similar to Java)
  – Multi-paradigm language:
    imperative/structured, object-oriented, functional, aspect-oriented
  – Dynamic typing
  – Automatic garbage collection

• Do you really understand all these terms?

QUIZ:
How is the foot related to Python?
Java to Python: Imperative Example (Java)

```java
public class Main {

    public static int sequentialSearch(int q, int[] a) {
        for(int i = 0; i < a.length; i++) {
            if(a[i]==q) {
                return i;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        int[] a = {11, 22, 33, 44, 55, 66};
        System.out.println("Array a: "+a);
        System.out.println("Search for 55: "+sequentialSearch(55,a));
        System.out.println("Search for 23: "+sequentialSearch(23,a));
    }
}
```
Java to Python: Imperative Example

(Python)

def sequentialSearch (q, a):
  for i in range(0,len(a)):
    if a[i]==q:
      return i
  return -1

a = [11, 22, 33, 44, 55, 66]
print("Array a: ", a)
print("Search for 55: ",sequentialSearch(55,a))
print("Search for 23: ",sequentialSearch(23,a))

QUIZ:
What are the differences to Java?
First Observations on Python

- Very compact code
- Data types are not specified
- Powerful but simple built-in list datatype
- Indentation (white space) is important for program semantics !!!
  - Block levels given by indentation
  - What is done in Java with {} brackets, is done here with indentation
- Example: A different (wrong!) algorithm:

```python
def sequentialSearch (q, a):
    for i in range(0, len(a)):
        if a[i] == q:
            return i
    return -1
```
Scripting Language

• Traditionally:
  A \textit{scripting language} is a programming language that is used to control
  some application software
  – Command languages for operating systems
  – Task automatization in user interfaces
  – Scripts for Web browsers, word processors, spreadsheet software, …

• Historically, considered slow in execution and limited in program size

• Modern general-purpose scripting languages
  – Have inherited many features from traditional scripting languages
  – Are considered as full application programming languages:
    – Examples: Rexx, Perl, \textbf{Python}, Ruby
Compilation, Interpretation and Others

Compilation
(Example: C)

Program → Compiler → Machine code → Output

Input → Machine code → Output

Program

Interpretation
(Example: Python)

Input → Interpreter → Output

Variations:
• Compilation to intermediate code (Java)
• Just-in-time compilation
• Internal compilation to intermediate code in interpreter (Python/ CPython)
Interactive Interpreter

- Interpreted languages can easily be executed line-by-line
- Interactive execution is helpful for understanding
  - See BASIC, Logo etc.
Static and Dynamic Typing

• Type checking:
  – Simple, automatically executable form of proof for program correctness (in certain limited respects)
  – Avoids operations to be applied to unsuitable arguments

• **Static** typing:
  – Type information is checked **before execution** of program (at compile time)
  – Program code has to specify (explicitly or implicitly) types for all variables
  – Examples: Java, Pascal, C, Standard ML

• **Dynamic** typing:
  – Type information is checked **during execution** of program (at run time)
  – Type information for variables only exists after value assignment
  – Examples: Smalltalk, Python, JavaScript

• In practice, static and dynamic tying are sometimes mixed:
  – See the dynamic type check for *downcast* operations in Java!
Strong and Weak Typing

• Surprisingly ill-defined terms!
  – Do not take this classification too serious!

• **Strong** typing:
  – Basic idea: “Strong” typing provides no (or only very limited) possibility to evade the restrictions of the type system
  – Examples of strongly typed languages:
    - Java, Pascal, Standard ML, **Python**

• **Weak** typing:
  – Implicit type conversions
  – Type conversions with undefined result
  – Examples of weakly typed languages:
    - Visual Basic, C, JavaScript
Duck Typing

“When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck."
James Whitcomb Riley

- The type of an object is determined only by the fact whether it has the features required from it.
- Appropriate for object-oriented programming languages with dynamic types - like Python.
String Operations in Python

Operations valid for all sequence types:

- **Indexing:** `str[5]`  
  *(str is the string object)*

- **Negative indexing:** `str[-5]`  
  *(counting from the end)*

- **Slicing:** `str[2:5], str[:5], str[2:6:2], str[::-1]`
  - Omitted index is begin or end, third value is step size (covers reversion)

- **Operations:**
  - `len(str)`, `min(str)`, `max(str)`, `x in str`

Numerous methods specific for strings like:

- `capitalize()`
- `count(substr)`
- `find(substr)`
- `isalpha()`
- `partition(sep)`
- `replace`
- `split(sep)`
- `upper()`
- `title()`
Lists in Python

- List: Sequential collection of objects (of arbitrary, also varying type)
- Can be easily used as stack or queue data structures
- Flexible creation of lists e.g. by list comprehension:
  \[ l = [3*x \text{ for } x \text{ in range}(1,4)] \]
- Lists are mutable (can be even changed through slices)
- List methods:
  - append
  - count
  - extend
  - index
  - insert
  - pop
  - remove
  - reverse
  - sort
Sets in Python

• Set: Unordered collection without duplicates
• Constructor
  – `set` builds a set from a list
• Basic mathematical operations for sets:
  – Union (|)
  – Intersection (&)
  – Difference (-)
  – Symmetric difference (^)
• Example:
  ```python
  set('multimedia') & set('programming')
  ```
Java to Python: Imperative Example (Python)

def sequentialSearch (q, a):
    return q in a

a = [11, 22, 33, 44, 55, 66]
print(a)
print("Array a: ", a)
print("Search for 55: ",{sequentialSearch(55,a)})
print("Search for 23: ",{sequentialSearch(23,a)})
Tuples and Dictionaries in Python

- Tuple: immutable collection of objects (of arbitrary type)
  
  \[ N = ('\text{max}', '\text{muster}') \]
  
  Strange: One-element tuple written as 'max',

- Easy unpacking of tuples:
  
  \[ \text{vorname, nachname} = ('\text{max}', '\text{muster}') \]

- Dictionary: Mutable collection of object maps (of arbitrary type)
  
  \[ \text{age} = \{ '\text{anna'} : 23, '\text{max'} : 22 \} \]

  - Key entries can only be of immutable type (strings, numbers, tuples)
  
  - Key entries must be \textit{hashable}
  
  - Main purpose: indexed access \texttt{age['anna']}

- Constructor accepts lists or \textit{generator expressions}:
  
  \[
  \text{dict}((x, x*x) \text{ for } x \text{ in range(0,5)})
  \]
Java to Python: Object-Oriented Example (Java)

```java
public class Counter {

    private int k = 0;

    public void count () {
        k++;
    }

    public void reset () {
        k = 0;
    }

    public int getValue () {
        return k;
    }

}
```
Java to Python: Object-Oriented Example (Python)

class Counter:

    def __init__(self):
        self.k = 0

    def count(self):
        self.k += 1

    def reset(self):
        self.k = 0

    def getValue(self):
        return self.k

Initialization (constructor)

Instance variable k

“Self” parameter is implicit in method calls but explicitly mentioned in declaration
Constructing Objects, Invoking Methods

• Example:
  
  ```python
  c = Counter()
  print(c.getValue())
  c.count()
  c.count()
  c.count()
  c.count()
  print(c.getValue())
  ```
Inheritance in Python

class LimitCounter(Counter):

    def __init__(self, limit):
        self.k = 0
        self.limit = limit

    def count(self):
        if self.k != self.limit:
            self.k += 1

In contrast to Java, Python allows *multiple inheritance*!
Python Modules

• Module: A file containing Python definitions and statements
  – File name is module name with suffix `.py`
  – Module name is available as global variable `__name__`
  – Statements in a module are executed when the module is imported (initialization)

• Importing a module `m`:
  ```python
  import m
  ```
  – Accessing a definition `f()` in `m`:
    ```python
    m.f()
    ```
  ```python
  from m import *
  ```
  – Accessing a definition `f()` in `m`:
    ```python
    f()
    ```
Why Python in This Lecture?

Python is **not** a specific multimedia language!

We will use a simple Python-binding for a multimedia/gaming framework…

Generally, knowing Python is a good thing – to get programming tasks done easily.