A hands-on introduction to Python programming

BIM309 – Artificial Intelligence

Python Series (1 of 4)

Scope

- Prerequisites:
  - Downloading and installation of Python and PyCharm (IDE)

Part 1	Variables and simple data types Introducting lists Working with lists
Part 2	if statements Dictionaries User input and while loops
Part 3	Functions Classes
Part 4	Files and exceptions Testing

Variables and Simple Data Types

Variables

- Dynamically-typed (no declaration of data type)
- Naming conventions:
  - Letters, numbers, and underscores (numbers cannot be at the beginning)
  - Case sensitive
  - Spaces are not allowed.
  - Python keywords (int, float, etc.) cannot be used.
  - Short and descriptive
- type()
- isinstance()

Strings

- Series of characters
- Single or double quotes
  - msg1 = ‘Hello, students’
  - msg2 = “Hello, world”
- + to concatenate strings
- rstrip(), lstrip(), strip()
- title(), upper(), lower()

Numbers

- Integers, floats, and complex numbers
- Arithmetic ops:

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
**	Exponentiation
//	Floor divison

Introducing Lists

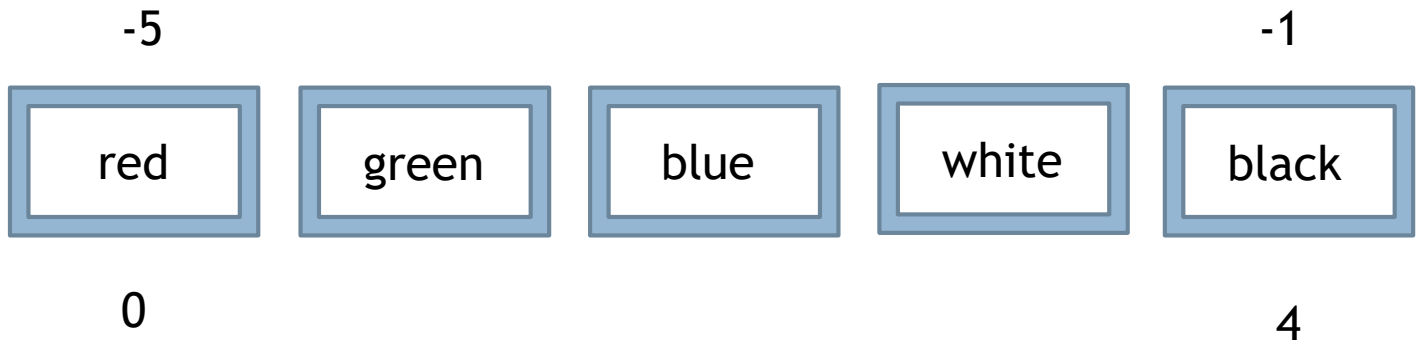
What is a List?

- A collection of items in a particular order
  - Analogous to ArrayList in Java
  - Data types can be arbitrary.
- A list of some colors:
  - colors = [“red”, “green”, “blue”, “white”, ”black”]
- A list of arbitrary values:
  - arby = [1, 2.5, “hello”, False, [4, 5]]

Accessing List Elements

- Index positions start at 0 (not 1, like in many of other languages)

- colors[0] will return the first element of colors (red)
- colors[-1] will return the last element of colors (black)
  - No need to know length of list to retrieve the last element



Changing, Adding, Removing List Elements

- Changing an element by index
  - colors[0] = ”yellow”
- Two ways of adding an element to a list:
  - colors.append(“red”) → adding to the end of list
  - colors.insert(0, “brown”) → positional insert
- Three ways of removing an element from a list:
  - del colors[1] → remove by del keyword and index
  - colors.pop() → retrieve and remove last element
  - colors.remove(“green”) → remove by value

Organizing a List

- Sorting a list permanently with sort() method:
  - colors.sort()
- Sorting a list temporarily with sorted() method:
  - sorted(colors)
- Reversing a list:
  - colors.reverse()
  - colors[::-1]
- len()

Working with Lists

Making Numerical Lists

- range() function (actually returns a range instance) can be used to create numerical lists.
  - range parameters (start is 0 by default, stop, step is 1 by default)

```
digits = list(range(10))
even_digits = list(range(0,10,2))
```

```
squares = []
for i in range(1,11):
    squares.append(i**2)
```

Can be shortened by list comprehensions

List Comprehensions

- An elegant way to create lists based on existing lists

- Comprehension syntax:
  - new_list = [expression for member in iterable (if conditional)]

```
squares = [i**2 for i in range(1,11)]
even_digits = [i for i in range(0,10) if i % 2 == 0]
vals = [abs(x) for x in [5, -4, 3, 0, -1]]
```

List Comprehensions

- Compact and faster (in general)
- Very long, complex comprehensions may be hard to read.
- Every list comprehension can be rewritten in for loop, but every for loop cannot be rewritten in the form of list comprehension.

Slicing a List

- Building a new list (sublist, shallow copy) from an existing list
- Full slice syntax → start:stop:step

- nums = [10, 20, 30, 40, 50, 60, 70, 80, 90]
  - nums[1:5]
  - nums[0:3] or nums[:3] n first elements
  - nums[-3:] n last elements
  - nums[-4:8]
  - nums[1:7:2]
  - nums[::-1] reverse
  - nums[-2:1:-2]

Tuples

- Immutable objects: the objects that cannot be changed after initialization (e.g., int, bool, string)
- A tuple is an immutable list.
  - x = (1,) → This comma is needed for the interpreter when tuple has only one item
  - y = (1, 2, 3)
- Indexing, slicing, and basic operations (length, concatenation, membership, iteration) are same as lists.