Introduction to Python

Lect. PhD. Arthur Molnar

Babes-Bolyai University

arthur@cs.ubbcluj.ro

Overview

1. Introduction to Python
   - Data in Python
   - Simple Data Types
   - Compound Data Types
   - Variables, expressions and statements

Hardware and software

- **Hardware** -*computers* (desktop, mobile, etc) and related *devices*
- **Software** -*programs* or *systems* which run on hardware
- **Programming language** - notation that defines syntax and semantics of programs

What computers do

- Storage and retrieval
    - Internal memory
    - Hard disk, memory stick
- Operations
    - Processor
- Communication
    - Keyboard, mouse, display
    - Network connector

Python 101

- **Python** - a high level programming language. It is a great language for beginner programmers!
- **Python interpreter** - a program which allow us to run/interpret new programs.
- **Python standard library**: built-in functions and types

Python 101

Python is:

- A modern programming language
- Simple to write and understand
- An **interpreted** language
- A **garbage collected** language
- A language that support multiple paradigms: structured, object-oriented, functional and aspect oriented programming are all on the menu!
- A language with great support and many available libraries

Python 101

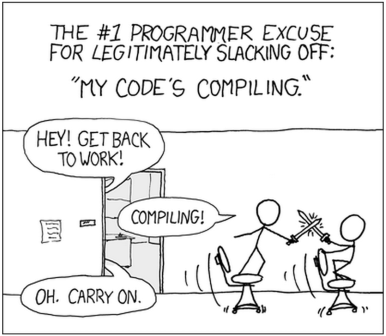
Python is...

Simple to write and understand

```
myList = []
while True:
    x = int(input("Enter item (-1 to finish):"))
    if x == -1:
        break
    myList.append(x)
return myList
```

Python is...

An **interpreted** language

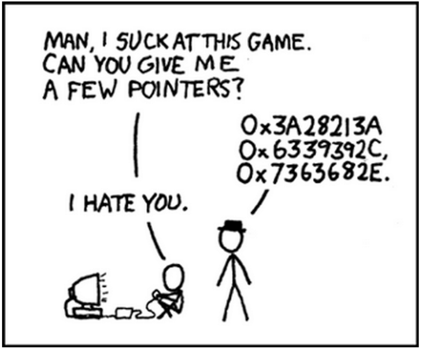


[1]

[1] https://xkcd.com/303/

Python is...

A **garbage collected** language


MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A 0x6339392C, 0x7363682E.
I HATE YOU.
[2]

[2] https://xkcd.com/138/

Python 101

Python mantra[3]:

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Flat is better than nested
- Sparse is better than dense
- Readability counts

[3] https://www.python.org/dev/peps/pep-0020/

What do you need?

1. Python 3
2. GitHub Desktop (**OR** use git in command line, **OR** git integration with IDEs)
3. An IDE (Eclipse + PyDev **OR** VS Code **OR** PyCharm)

OR

1. Use the PythonBox - a virtual machine we've prepared as a backup solution for the exam, which you can also use from home[4]

[4] http://www.cs.ubbcluj.ro/~arthur/PythonBox/

Basic elements of a Python program

- **Lexical elements** - a Python program is divided into a number of **lines**.
- **Comments** - start with a hash (#) character and ends at the end of the line.
- **Identifiers** (or **names**) - are sequences of characters which start with a letter (a..z, A..Z) or an underscore (_) followed by zero or more letters, underscores, and digits (0..9).
- **Literals** - are notations for constant values of some built-in types.

Demo

Basic elements of a Python program

ex01_BasicSyntax.py

Data vs. Information

- **Data** - collection of symbols stored in a computer (e.g. 123 decimal number or 'abc' string are stored using binary representations)
- **Information** - interpretation of data for human purposes (e.g. 123, 'abc')

Python data model

All data in Python programs is represented by objects, an **object** being Python's abstraction for data.
An **object** has:

- an **identity** - we may think of of it as the object's address in memory.
- a **type** - which determines the operations that the object supports and also defines the possible values.
- a **value**.

Types

- **Types** classify values. A type denotes a **domain** (a set of possible values) and **operations** on those values.
- **Numbers** - are immutable, so once created, their values cannot be changed.

Identity, value and type

Recall what is a *name* and an *object* (*identity*, *type*, *value*).

- mutable objects: lists, dictionaries, sets
- immutable: numbers, strings, tuples

We determine the identity and the type of an object using the built-in functions:

- **id**(object)
- **type(object)**, **isinstance(object, type)**

Standard types in Python (1/3)

int[5]:

- Represents the mathematical set of integers (positive and negative, unlimited precision)

bool:

- Represents the the truth values True and False.

float:

- Represents the mathematical set of double precision floating point numbers.

[5] https://docs.python.org/3/library/stdtypes.html

Standard types in Python (2/3)

Sequence types[6]

- Finite ordered sets indexed by non-negative numbers
- Let a be a sequence.
  - len(a) returns the number of items
  - a[0], a[1], . . . , a[len(a)-1] represent the set of items
- Examples: [1, 'a']

string

- A string is an immutable sequence
- The items of a string are Unicode characters

[6] https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range

Standard types in Python (3/3)

list[7]

- Mutable sequence of elements
- Typically used to store collections of homogeneous items
- Every item has a predecessor and successor

tuple[8]

- Immutable sequence
- Typically used to store collections of homogeneous items

dict[9]

- Mapping between unique keys and values

[7] https://docs.python.org/3/library/stdtypes.html#list

[8] https://docs.python.org/3/library/stdtypes.html#tuple

[9] https://docs.python.org/3/library/stdtypes.html#dict

Demo

Basic compound types

ex02_BasicCompoundTypes.py

List

Lists represent finite ordered sets indexed by non-negative numbers.

Operations:

- Creation
- Accessing values (index, len), changing values (**lists are mutable**)
- Removing items (pop), inserting items (insert)
- Slicing
- Nesting
- Generate list using range(), list in a for loop
- Lists as stacks (append, pop)

Tuple

Tuples are immutable sequences. A **tuple** consists of a number of values separated by commas.

Operations:

- Packing values (creation)
- Nesting
- Empty tuple
- Tuple with one item
- Sequence unpacking

Dictionary

A **dictionary** is an unordered set of (key, value) pairs with unique keys. The keys must be immutable.

Operations:

- Creation
- Getting the value associated to a given key
- Adding/updating a (key, value) pair
- Removing an existing (key, value) pair
- Checking whether a key exists

Variables and expressions

NB!

Variables are reserved memory locations to store values

- A **variable** has:
    - Name
    - Type
        - Domain
        - Operations

A variable is introduced in a program using a name binding operation - assignment.

Variables and expressions

- **Expression** - a combination of explicit *values*, *constants*, variables, *operators*, and *functions* that are interpreted according to the particular*rules of precedence*, which computes and then *produces/returns* another value.
- **Examples**:
    - numeric expression: $1 + 2$
    - boolean expression $1 < 2$
    - string expression: $'1' + '2'$

Statements

NB!

Statements are the basic operations of a program. A program is a sequence of statements

Statements

- **Assignment**
  - Assignments are used to (re)bind names to values
  - Bind name:
    - x = 1 *#is a variable (of type int)*
  - Rebind name:
    - x = x + 2 *#a new value is assigned to x*
  - Rebind name of mutable sequences:
    - y = [1, 2] *#mutable sequence*
    - y[0] = -1 *#the first item is bound to -1*
- **Block**
  - A block is a section of a program that is executed as a unit
  - A sequence of statements is a block
  - Blocks of code are denoted by line indentation

Demo

Controlling program flow

ex03_ProgramFlow.py