BABEŞ-BOLYAI UNIVERSITY
Faculty of Mathematics and Computer Science

Algorithms and Programming

Lecture 1: Introduction

Camelia Chira
http://www.cs.ubbcluj.ro/~cchira

Outline

- Course organization
  - Objectives
  - Content
  - Activities and evaluation

- Programming process
  - What is programming?
  - Basic elements of Python

Objectives

- Learning the most important concepts of programming
- Getting familiar with software engineering concepts (*architecture, implementation, maintainance*)
- Understanding the basic software elements
- Learning the Python programming language and using it to implement, run, test and debug programmes
- Learning and improving a programming style

Course content

- Introduction & Basic elements of Python
- Procedural programming
- Modular programming
- Abstract data types, exceptions, classes
- Software development principles
- Testing and debugging
- Recursion
- Complexity of algorithms
- Search and sorting algorithms
- Backtracking
- Recap

Course bibliography

1. The Python Programming Language - https://www.python.org/
2. The Python Standard Library - https://docs.python.org/3/library/index.html
3. The Python Tutorial - https://docs.python.org/3/tutorial/
4. M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006.
5. M.L. Hetland, Beginning Python: From Novice to Professional, Apress, 2005.
6. MIT OpenCourseWare, Introduction to Computer Science and Programming in Python, https://ocw.mit.edu, 2016.
7. J. Elkner, A.B. Downey, C. Meyers, How to Think Like a Computer Scientist: Learning with Python, Samurai Media Limited, 2016.
8. K. Beck, Test Driven Development: By Example. Addison-Wesley Longman, 2002. http://en.wikipedia.org/wiki/Test-driven_development
9. M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999. http://refactoring.com/catalog/index.html

Schedule

- Timetable
  - *Lectures*: 2 hours / week
  - *Seminars*: 2 hours / week
  - *Labs*: 2 hours / week

- Camelia Chira, cchira@cs.ubbcluj.ro
- Course page, http://www.cs.ubbcluj.ro/~cchira

Activities and evaluation

- All activities are mandatory
  - Laboratory attendance mandatory: 90%
  - Seminar attendance mandatory: 75%
- Lab grading
  - *Lab assignments* are graded from 0 to 10 and should respect the deadline given
  - *2 lab tests* – each receive a grade
  - Delays in presenting assignments are penalized with 2 points for each week passed
  - Assignments that have not been presented receive grade 0
- A maximum of 2 assignments can be submitted each lab
- SUBMIT YOUR OWN WORK: A copied assignment receives grade 0

Activities and evaluation

- **Lab activities – 30%**
  - Several assignments (work during the lab & homework)
  - All assignments graded & 2 tests grades
  - **Lab grade = 0.5 * Average_Assignments + 0.5 * Average_Tests**
- **Practical exam – 30 %**
  - Practical test in last week of semester – **grade must be at least 5**
- **Written exam – 40%**
  - Conditions
    - Practical exam grade should be at least 5
    - Minimum attendance at labs and seminars
  - Written exam grade **must be at least 5**
- **Final grade = 0.3 * Lab grade + 0.3 * Practical exam + 0.4 * Exam (>= 5)**
- **Seminar**
  - Seminar activities can increase the final grade by 0-1 points

Software development process

- What is programming?
- Basic elements of Python

Software development

- Hardware
  - Computers (desktops, laptops, etc) and related devices
- Software
  - Programs and systems that run on the hardware
- Programming language
  - Rules and notations to define the syntax and semantics of computer programs
- Python
  - High-level programming language
  - *Python Interpreter:* a program that allows running other programs
  - *Python Libraries:* built-in functions and types

What computers do

- Perform computations and remember results
- Store data and information in:
  - Internal memory
  - External memory (hard, memory stick, etc)
- Operate
  - With the help of the processor
- Communicate
  - Via keyboard, mouse, display
  - Network connections

Data and information

- Information – interpreting some data
  - The number 12
  - The string “abc”
- Data – a collection of symbols stored in the computer (using a certain representation)
  - 12 – 1100
  - “abc” – 97 98 99
- Processing data and information
  - Input devices transform information in data
  - Data are stored in memory
  - Output devices produce information from data
- Basic operations of processors
  - Binary representation
  - Ex. AND, OR, NOT, XOR, etc.

What is programming?

- Telling a computer what to do
  - You have to feed the computer an algorithm in some language it understands
  - Recipes and algorithms consist of ingredients (object, things) and instructions (statements)

- Creating recipes
  - a *programming language* provides a set of primitive operations
  - *expressions* are legal combinations of primitives in a programming language
  - expressions and computations have *values* and meanings

Programming languages

- Primitive constructs
  - English: words
  - Numbers, strings, simple operators
- Syntax
  - English: “Girls cat dog” vs. “Girl hugs dog”
  - 3*5 (syntactically valid)
  - “dog”5 *(not syntactically valid)*
- Semantics (which syntactically valid things have meaning)
  - English: “I are hungry”
  - 3+5
  - “dog”+5 *(semantic error)*

Where things can go wrong...

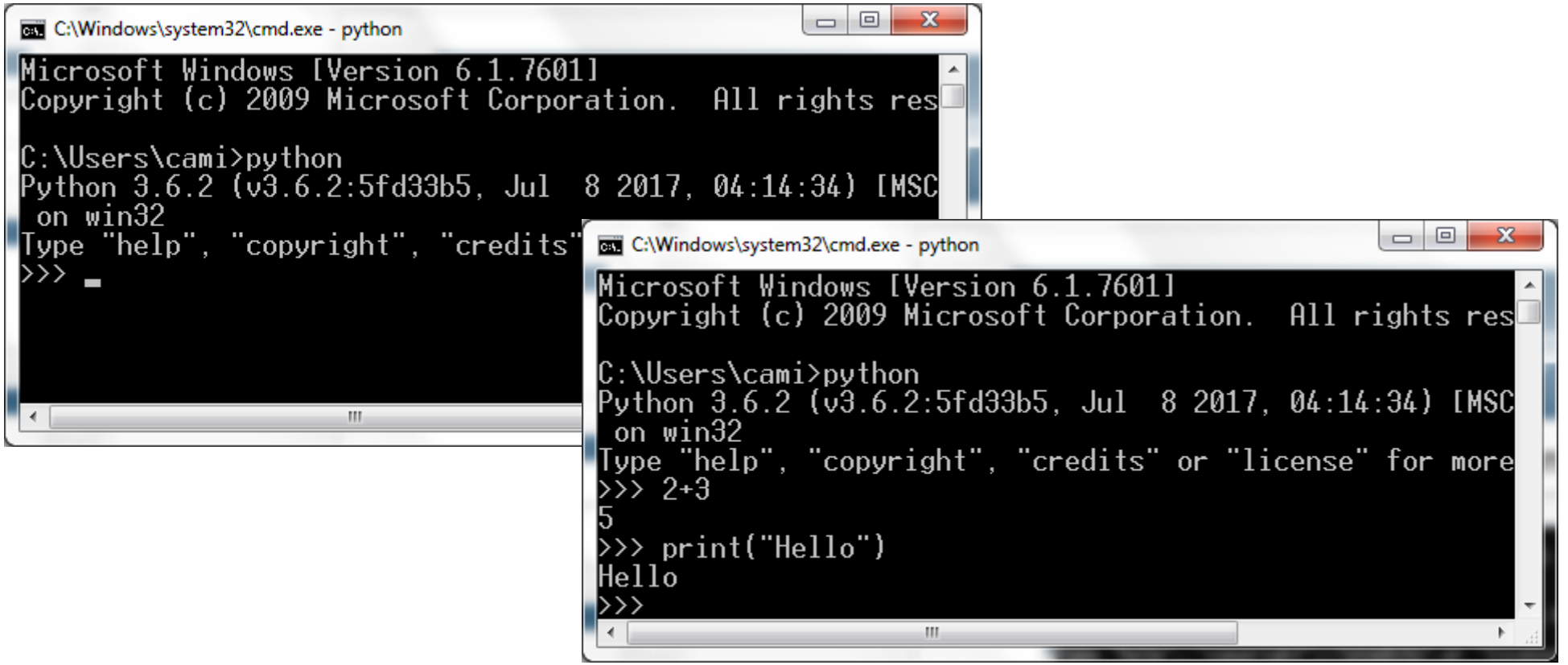
- Syntactic errors
  - Common but easy to identify and fix
- Runtime errors
  - Also called exceptions
- Semantic errors
  - Can sometimes cause unpredictable behavior
- Programming languages: a syntactically correct string of symbols has only one meaning but may not be what programmer intended
  - Different meaning than what the programmer intended
    - Program stops running (crashes)
    - Program runs forever
    - Program gives different answer than the expected one

Why Python?

- Python is a high-level programming language
  - **Interpreted**: processed at run time by the interpreter
  - **Interactive**: you can directly interact with the interpreter to write programs
  - Supports many paradigms e.g. structured, object-oriented, functional programming
  - Garbage collection
- Features
  - Easy to learn, easy to read, easy to maintain
  - Broad standard libraries
  - Portable, extendable, databases, GUI programming
- Who uses Python?
  - *Linux*: system administration tasks in several Linux distributions
  - *NASA*: as the standard scripting language in its Integrating Planning System
  - *Industrial Light & Magic*: production of special effects for large-budget feature films
  - *Google*: many componets of the Web crawler and search engine
  - Computer games and bioinformatics...etc.*who isn't using it?*

The Interactive Interpreter

- Shell mode (interactive programming)

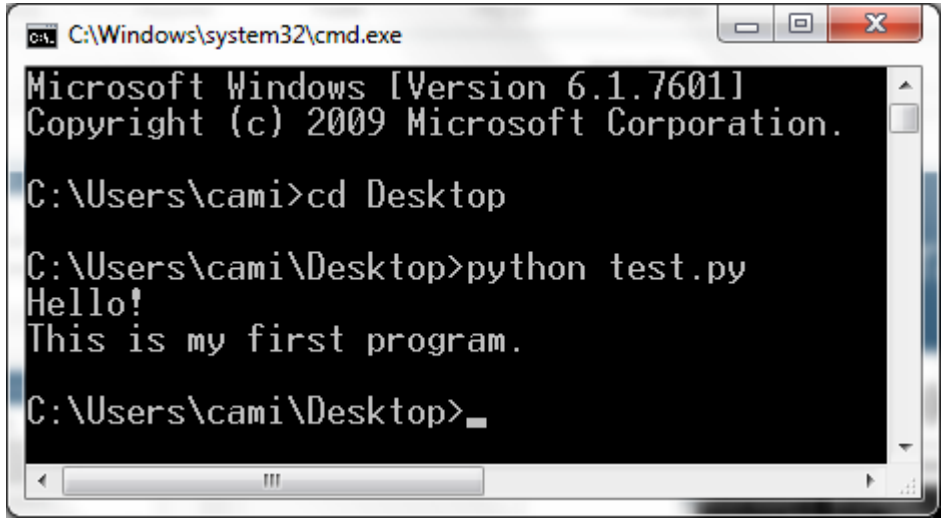


The Interactive Interpreter

- Script mode programming

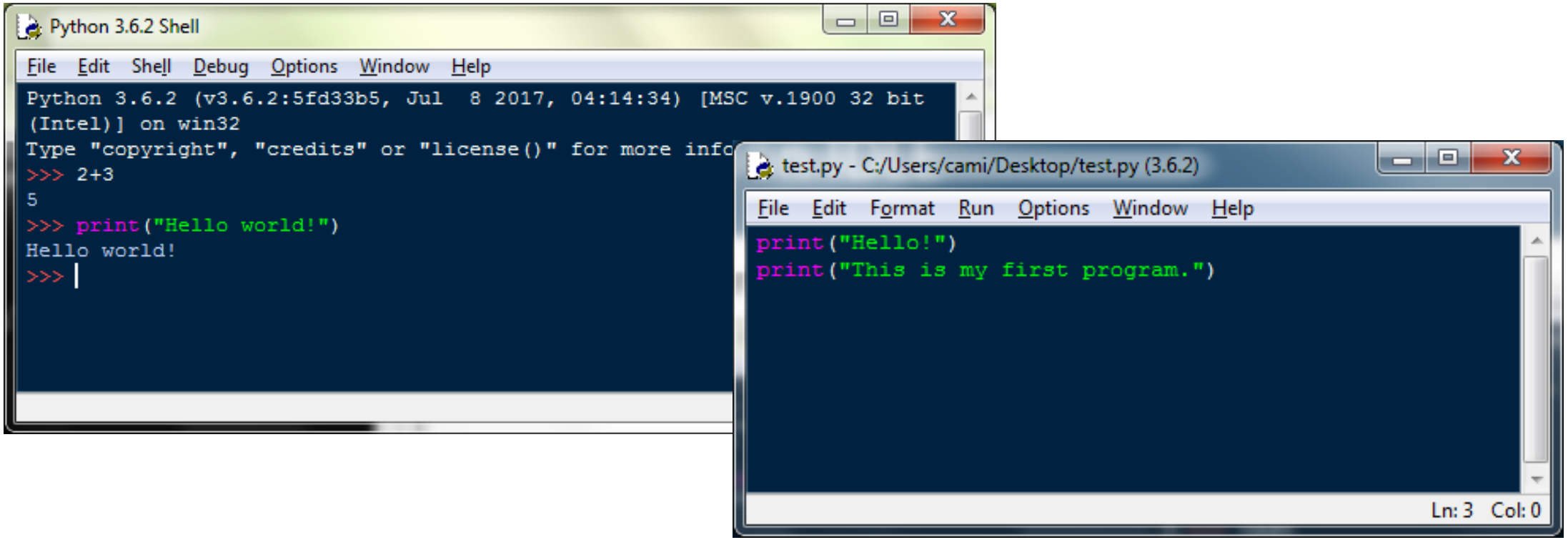
test.py

```
print("Hello!")
print("This is my first program.")
```



Python IDLE (Integrated DeveLopmentEnvironment)

- IDLE is the standard Python development environment
- Use interactive mode or script mode programming



Python programs

- A sequence of definitions and statements. Example:

```
# takes two integers and prints their sum
a = 3
b = 4
c = a + b
print("The sum of ", a, " and ", b, " is ", c)
```

- **Lexical elements** – a Python program can have several lines
- **Comments**
  - Start with # and last to the end of line
  - Start with ’’’ and last several lines until another ’’’
- **Identifiers**
  - Name used to identify a variable, function, class, module
  - Character sequences (letters, numbers, _) starting with a letter or _
- **Literals**
  - Notations for constant values or user-defined types

Python programs

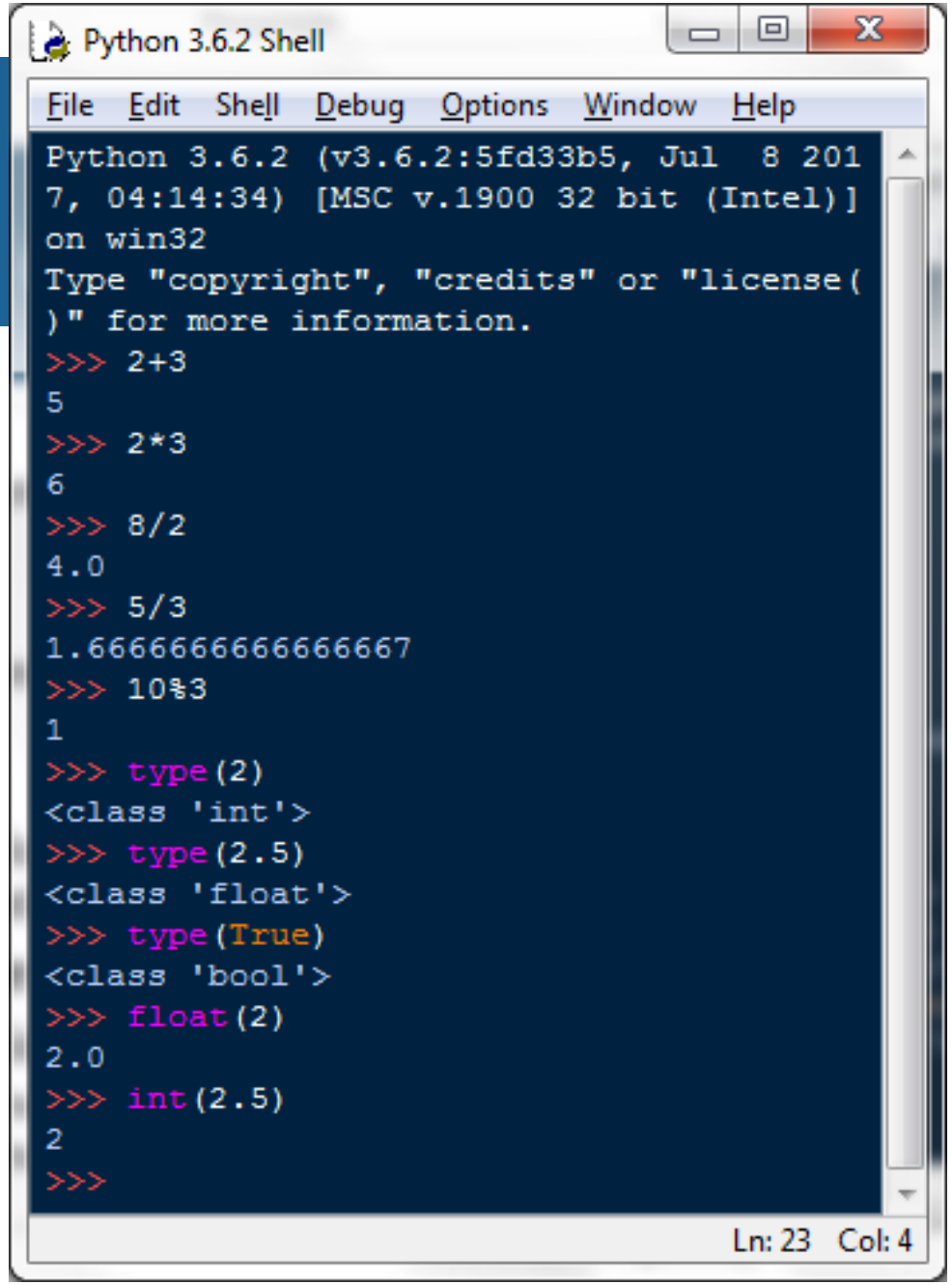
- Programs manipulate data objects
- Objects have:
  - An identity – address of the object in the memory
  - A type – determines the values the object can take and the operations possible on that object
  - A value
- Once created, the identity and type of the object can not be changed
- The value of some objects can be modified
  - Mutable objects
  - Immutable objects

Data types

- Domain – set of values
- Operations
- **Standard data types**
  - Number
  - String
  - List
  - Tuple
  - Dictionary
- **Taxonomy**
  - ***Numbers*** - immutable
  - ***Sequences*** – mutable and immutable
    - Let **s** be a sequence:
      - **len(s)** returns the number of elements in **s**
      - **s[0], s[1],...,s[len(s)-1]** are the elements of **s**
      - Example: **s=[1, 'a', 23, "abc"]**

Numeric data types

- **int**
  - represent integers ex. 1, 23
  - +, -, *, /
- **float**
  - represent real numbers ex. 3.27
  - +, -, *, /
- **bool**
  - represent Boolean values ex. True, False
  - Logic operations (and, or, not,...)
- type() - to see the type of an object
- Type conversions (cast)
  - float(2)
  - int(2.5)



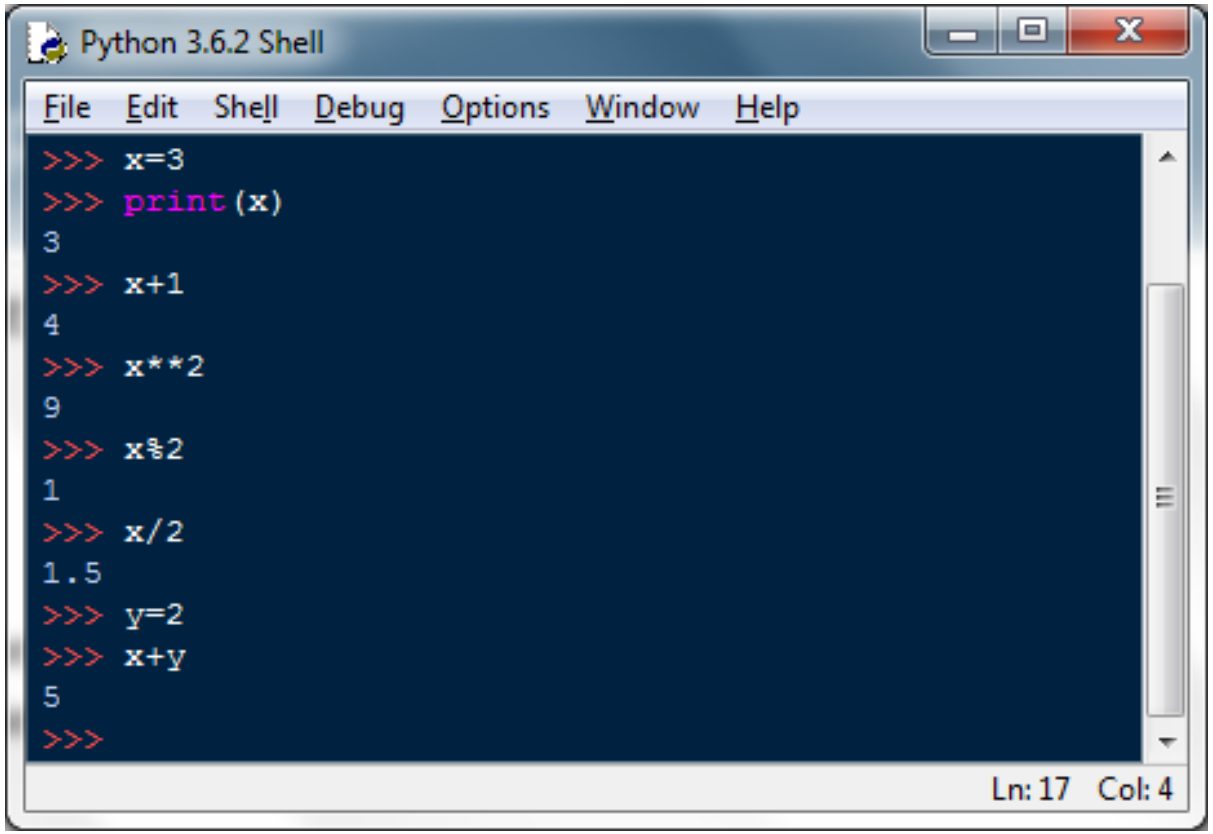
Basic elements of a Python program

- Variables
  - Locations in memory where data is stored
  - Have a name, a datatype and a value
  - Introducing a variable in a program – assignment
- Expressions
  - A combination of values, constants, variables, operators and functions which are interpreted according to precedence rules, computed and evaluated to a value
  - Examples
    - Numerical expression: `1 + 2`
    - Boolean expression : `1 < 2`
    - String expression : `“1” + ”2”`
- `Statements`

Variables and expressions

- A variable is a name that represents some value
- Assignment: x=3
- Expressions
  - Combine objects and operators
  - An expression has a value -> type
  - Ex. x+1, x**2

x+y	sum (result is int if both x and y are int, float if x or y is float)
x-y	Difference
x*y	Product
x/y	division (result is float)
x%y	remainder
xy**	power



Statements

- The basic operations of a program
- Taxonomy
  - Assignments
    - (Re-)binding variable names to values and changing the value of mutable objects
    - Binding: `x = 1, s = [1, 2]`
    - Re-binding: `x = x + 2, s[0] = 3`
  - Blocks
    - Part of a program executed as a unit
    - Sequence of statements
    - Identified using identation
  - Conditional statements
  - Loops

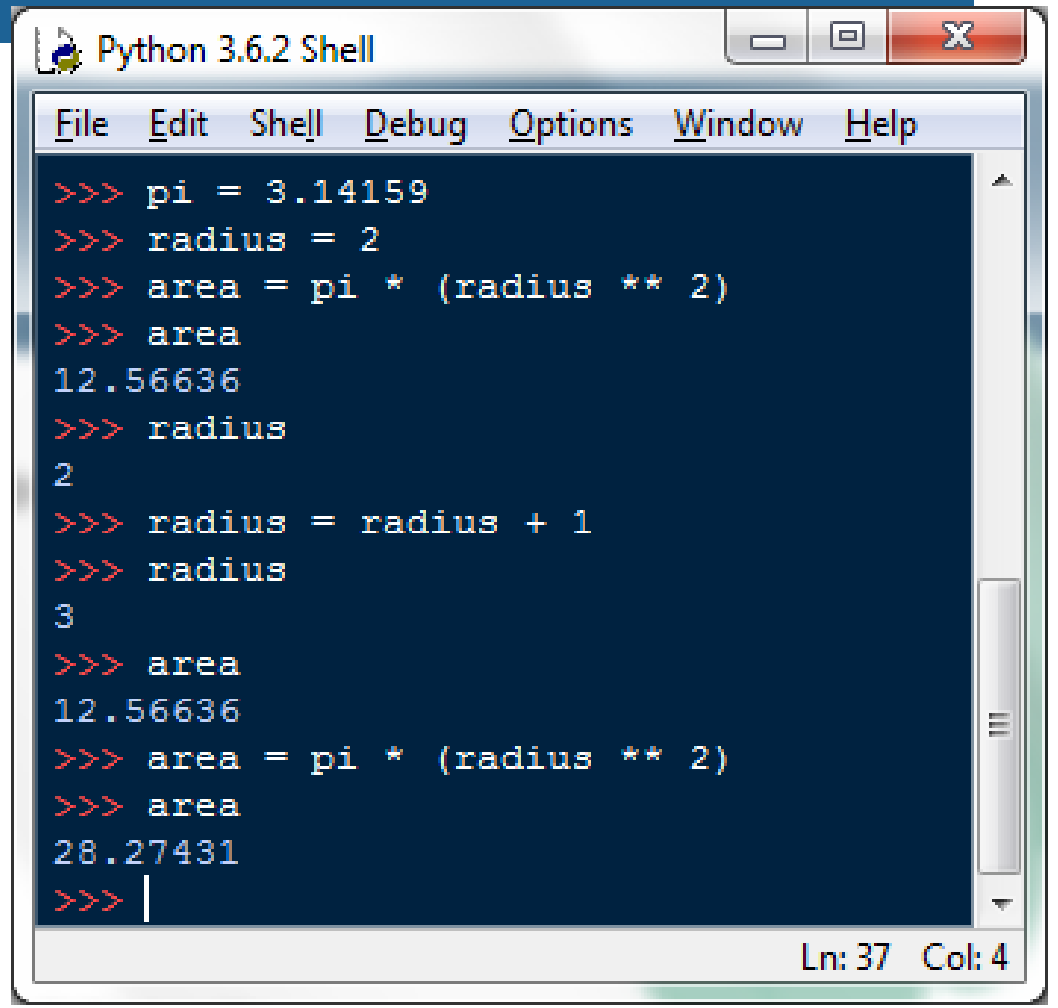
Assignments

- *On the left:* variable name
- *On the right:* expression, evaluated to a value

```
pi = 3.14159
radius = 2
# area of circle
area = pi * (radius ** 2)
radius = radius + 1
```

- Changing bindings:
  - Re-bind variables using new assignment statements
  - Previous value may still be stored – no handle to it
- Multiple assignments:

```
a = b = c = 1
a, b, c = 1, 2, "Zara"
```



Comparison and logic operators

- Comparison operators (int, float, string)
  - `a > b`
  - `a >= b`
  - `a < b`
  - `a <= b`
  - `a == b` (equality test, `True` if `a` is the same as `b`)
  - `a != b` (inequality test, `True` if `a` is not the same as `b`)
- Logic operators (bool)
  - **`not`** a (True if a is False, False if a is True)
  - a **`and`** b (True if both are True)
  - a **`or`** b (True if either or both are True)

```
my_age = 40
your_age = 20
print(my_age < your_age) # False

age = my_age >= 18 # True
license = False

b = age and license
print(b) # False
```

Conditional statements

```
if <condition>:
        <expression>
        <expression>
        ...
```

- Control flow – branching

```
# takes two integers and prints their max
a = 3
b = 4
if (a < b):
    c = b
else:
    c = a
print("The max of ", a, " and ", b, " is ", c)
```

```
if <condition>:
        <expression>
        ...
else:
        <expression>
        ...
```

```
if <condition>:
        <expression>
        ...
elif <condition>:
        <expression>
        ...
else:
        <expression>
        ...
```

Indentation

- Important in Python
- Blocks of code are identified using indentation

```
if a == b:
    print("a and b are equal")
    if b != 0:
        print(", meaning a/b =", a/b)
elif (a < b):
    print("a = ", a, " is smaller")
else:
    print("b = ", b, " is smaller")
print("The end")
```

Control flow: while and for Loops

- while

```
i = 0
while i < 10:
    print(i)
    i = i + 1
```

```
while <condition>:
        <expression>
        <expression>
        ...
```

- for

```
for i in range(10):
    print(i)
```

```
for <variable> in range(<some_num>):
        <expression>
        <expression>
        ...
```

range(start, stop, step)

- Starts with `value = start` (default `start = 0`)
- Each step, `value = value + step` (default `step = 1`)
- Loops until `value = stop - 1`

```
s = 0
for i in range(5):
    s += i
print(s)
```

```
s = 0
for i in range(1, 5, 2):
    s += i
print(s)
```

Example

```
# computes the gcd of two numbers
a = 42
b = 18
if a == 0:
    gcd = b
else:
    if b == 0:
        gcd = a
    else:
        while a != b:
            if a > b:
                a = a - b
            else:
                b = b - a
        gcd = a
print("gcd = ", gcd)
```

break Statement

- Exits a loop and skips the rest of the block

```
while <condition_1>:
        while <condition_2>:
                <expression_a>
                break
                <expression_b>
        <expression_c>
```

```
s = 0
for i in range(2, 10, 2):
    s += i
    if s == 2:
        break
        s = s + 1
    s += 10
```

Strings

- *Domain*: character sequence(letters, special characters, digits)
- *Operations*: concatenation, search
- Immutable
- Enclose in quotation marks or single quotes

```
s = "hello there"
```

```
>>> s = "hello there"
>>> s
'hello there'
```

- Concatenate strings

```
name= "Zara"
greeting1 = s + name
greeting2 = s + " " + name
greeting3 = s + " " + name * 3
```

Strings

- Indexing

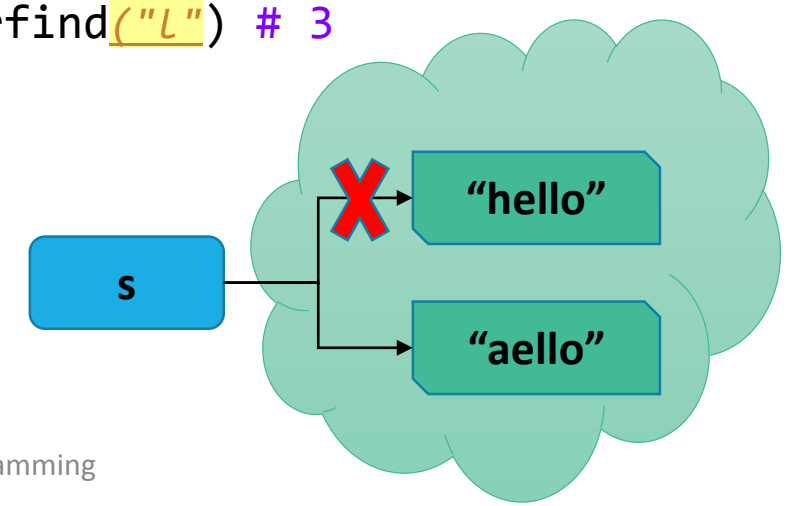
```
  s    = “abc”
index    0 1 2
index   -3 -2 -1

 s[0], s[-3] -> “a”
 s[1], s[-2] -> “b”
 s[2], s[-1] -> “c”
```

- Strings are immutable

```
 s    = “hello”
 s[0]= “a” # gives an error
 s    = “a” + s[1:] # ok (s is bound
                      to new object)
```

- Slicing and search

```
s = "hello there"
s[0]     # ’h’
s[2:5]   # ’llo’
s[2:]    # ’llo there’
s[:2]    # ’he’
s[-1]    # ’e’
s.find("l")  # 2
s.rfind("l") # 3
```



Input / Output

- print

```
x = 2
print(x)
x_str = str(x)
print("My number is", x, ".")
print("My number is " + x_str + ". ")
```

- input

```
age = input("Your age is:")
print(age)
```

input gives you a string

```
s = input("Your age is:")
age = int(s)
print("Age is ", age, ". Next year you will be ", age + 1, "...")
```

Recap today

- Programming process
  - What is programming?
  - Basic elements of Python

Next time

- More on Python basics

- Procedural programming
  - Functions
  - Variables
  - Parameters
  - Testing

Reading materials and useful links

1. The Python Programming Language - https://www.python.org/
2. The Python Standard Library - https://docs.python.org/3/library/index.html
3. The Python Tutorial - https://docs.python.org/3/tutorial/
4. M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006.
5. M.L. Hetland, Beginning Python: From Novice to Professional, Apress, 2005.
6. MIT OpenCourseWare, Introduction to Computer Science and Programming in Python, https://ocw.mit.edu, 2016.
7. J. Elkner, A.B. Downey, C. Meyers, How to Think Like a Computer Scientist: Learning with Python, Samurai Media Limited, 2016.

Bibliography

The content of this course has been prepared using the reading materials from previous slide, different sources from the Internet as well as lectures on Fundamentals of Programming held in previous years by:

- Prof. Dr. Laura Dioşan - www.cs.ubbcluj.ro/~lauras
- Conf. Dr. Istvan Czibula - www.cs.ubbcluj.ro/~istvanc
- Lect. Dr. Andreea Vescan - www.cs.ubbcluj.ro/~avescan