15-110 Principles of Computing – S21

Lecture 5:
An introduction to Python

Teacher:
Gianni A. Di Caro

So far

- Some training to think as a computer and act as a computer
- Writing algorithms use the English language → Natural language
- Looked at mathematical formalism to start writing algorithms in a more precise / compact way
  → ~ Formal language
- Introduced concepts that allow and favor abstraction: variables, indices, parametric procedures
- → Today: start up the transition to the use of a fully formal language to express, test, and use our algorithms: Python programming language!

A simple example from the quizzes

1. Start with the number 7

2. Multiply by the current month
3. Subtract 1

4. Multiply by 13

5. Add today's day

6. Add 3

7. Multiply by 11

8. Subtract the current month

9. Subtract the current day
10. Divide by 10
11. Add 11

12. Divide by 100

```
value = 7
value = value * 1
value = value - 1
value = 13 * value
value = value + 26
value = value + 3
value = value * 11
value = value - 1
value = value - 26
value = value / 10
value = value + 11
value = value / 100
```

Looks like simple math! ☺

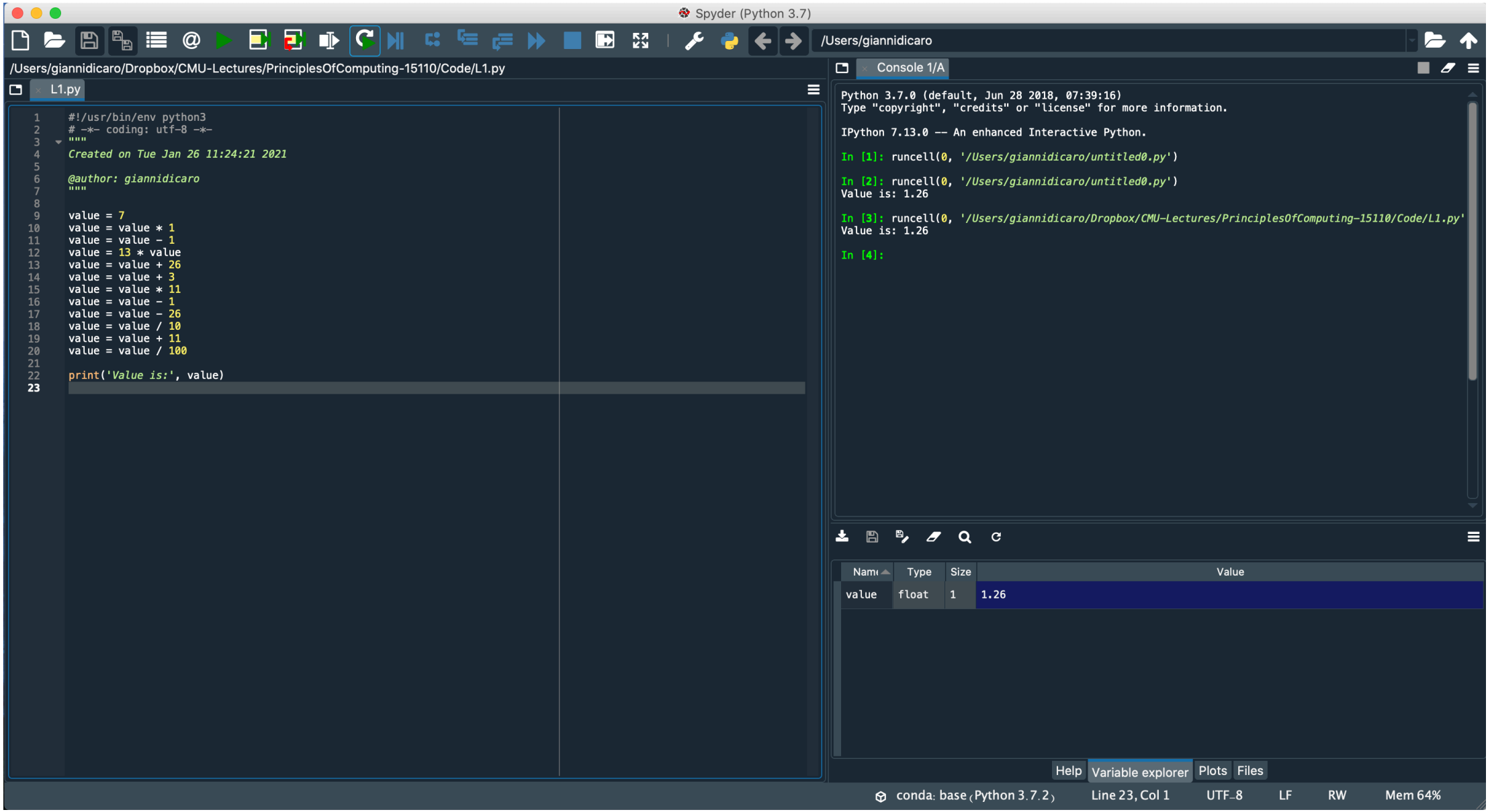
From the python code to its execution

- ✓ We have the algorithm in python, written *somewhere*
- ➢ How / where do we execute it on a computer?
- ➢ How do we get the output results?
- ➢ Can we pack it in a parametric procedure (a ***function***) to reuse it at different days / months?

```
value = 7
value = value * 1
value = value - 1
value = 13 * value
value = value + 26
value = value + 3
value = value * 11
value = value - 1
value = value - 26
value = value / 10
value = value + 11
value = value / 100

print('Value is:', value)
```

Spyder: Integrated Development Environment (IDE)



Basic elements in python code: comments and string descriptions

```
1   #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3   """
4   Created on Tue Jan 26 11:24:21 2021
5
6   @author: giannidicaro
7   """
8
9   # Code for question 1 of Quiz01
10  value = 7
11  value = value * 1
12  value = value - 1
13  value = 13 * value
14  value = value + 26
15  value = value + 3
16  value = value * 11
17  value = value - 1
18  value = value - 26
19  value = value / 10
20  value = value + 11
21  value = value / 100
22
23  print('Value is:', value)
24
```

A special comment line (a *shebang* line)

Triple quotes are used to write multi-line *descriptions* (technically these are not comments, but we can treat as they are for now ;-)

labels that line as a comment line

Comments are useful / necessary to understand what's going on!

Colors have no formal meaning they are there to **help you** identifying the different elements of the code

Let's move to Spyder!

Let's make an *abstraction* step: add variables for day, month

```
value = 7
value = value * 1
value = value - 1
value = 13 * value
value = value + 26
value = value + 3
value = value * 11
value = value - 1
value = value - 26
value = value / 10
value = value + 11
value = value / 100
```

→

```
day = 26
month = 1
value = 7
value = value * month
value = value - 1
value = 13 * value
value = value + day
value = value + 3
value = value * 11
value = value - month
value = value - 26
value = value / 10
value = value + 11
value = value / 100
```

Let's pack the code in a (parametric) *function*

- We need to give a **name to a function variable** such that we can flexibly reuse the code just referring to the function name
- The function **returns** the result, that can be manipulated further

```
day = 26
month = 1

value = 7
value = value * month
value = value - 1
value = 13 * value
value = value + day
value = value + 3
value = value * 11
value = value - month
value = value - 26
value = value / 10
value = value + 11
value = value / 100
```

Let's pack the code in a *function,* without parameters

```
day = 26
month = 1
value = 7
value = value * month
value = value - 1
value = 13 * value
value = value + day
value = value + 3
value = value * 11
value = value - month
value = value - 26
value = value / 10
value = value + 11
value = value / 100
```

→

```
def simple_computation():
    day = 26
    month = 1
    value = 7
    value = value * month
    value = value - 1
    value = 13 * value
    value = value + day
    value = value + 3
    value = value * 11
    value = value - month
    value = value - 26
    value = value / 10
    value = value + 11
    value = value / 100
    return value

v = simple_computation()
```

Let' see what happens in Spyder ...

Let's pack the code in a *parametric function*

```
def simple_computation(day, month):
    value = 7
    value = value * month
    value = value - 1
    value = 13 * value
    value = value + day
    value = value + 3
    value = value * 11
    value = value - month
    value = value - 26
    value = value / 10
    value = value + 11
    value = value / 100
    return value
```

- **Pass input arguments / parameters** (day and month) such that we can reuse the code for different days and months
- The function **returns** the result, that can be manipulated further

```
simple_computation(26, 1)

v = simple_computation(27, 1)
print('Value for 27/1 is:', v)
```

Let' see what happens in Spyder …

Homework: all functions to fill in!

```
def sum_of_two_numbers(x, y):
    # put your code here
    return s
```

Indentation spaces / tab!

The code is within the **scope** of the function definition

```
def sum_of_two_numbers(x, y):
    sum_two = x + y
    return sum_two
```

Basic tips

- Use Tab for autocompleting commands and variables names in the code
- Use F5 or Shift+Return for running the program
- Run the program often!
- Save your files continually
- Keep an eye on the console, it informs you which line is the error

Write a function!

Suppose you have invested a 100 Riyals on a fixed income fund that pays 0.2% per month. If you leave your money there for one month, how much do you have at the end? What about two months? What about three years?

Each time a month passes, you are paid 0.2% of the value in the fund. In the first month this value is 100, so at the end, you will have 100×1.002. In the second month, the amount in the fund is 100×1.002, so at the end of that month you will have $(100 \times 1.002) \times 1.002$. At the end of the third month, you will have $((100 \times 1.002) \times 1.002) \times 1.002$, and so on and so forth. You might have noticed the pattern now. At the end of the n-th month, you have 100×1.002^n.

In general, for an initial amount a in a fund paying $i\%$, at the end of month n, the amount will be $a \times (1 + i/100)^n$. Can you write a step by step instruction on how to calculate the final amount, given the initial amount, the interest, and the number of months? (You may assume all mathematical operations, including exponentiation, are available.)