

Review of Lecture 1

- **Programming language (Python)**
- **Why Python?**
 - Little code and easy to read
 - Programming concepts and problem solving
- **Installing Python**
 - Windows, Linux and macOS
 - Checking the Python installation
 - Windows**: `python --version`
 - Linux and macOS**: `python3 --version`
 - **The Python version**
 - Python 3.14.7

- **Ways to write Python code**
 - Text editors **Notepad**
 - Code editors **Visual Studio Code**
 - Integrated Development Environments
IDLE, Spyder, PyCharm
 - Notebook environments **Jupyter Notebook**
- **Python interpreter**
- **Interactive mode and script mode**

```
>>> 5 + 3
8
```

Program saved in a file
such as
`calculation.py`

Info 3

Introduction to Python

M. Bouzenita

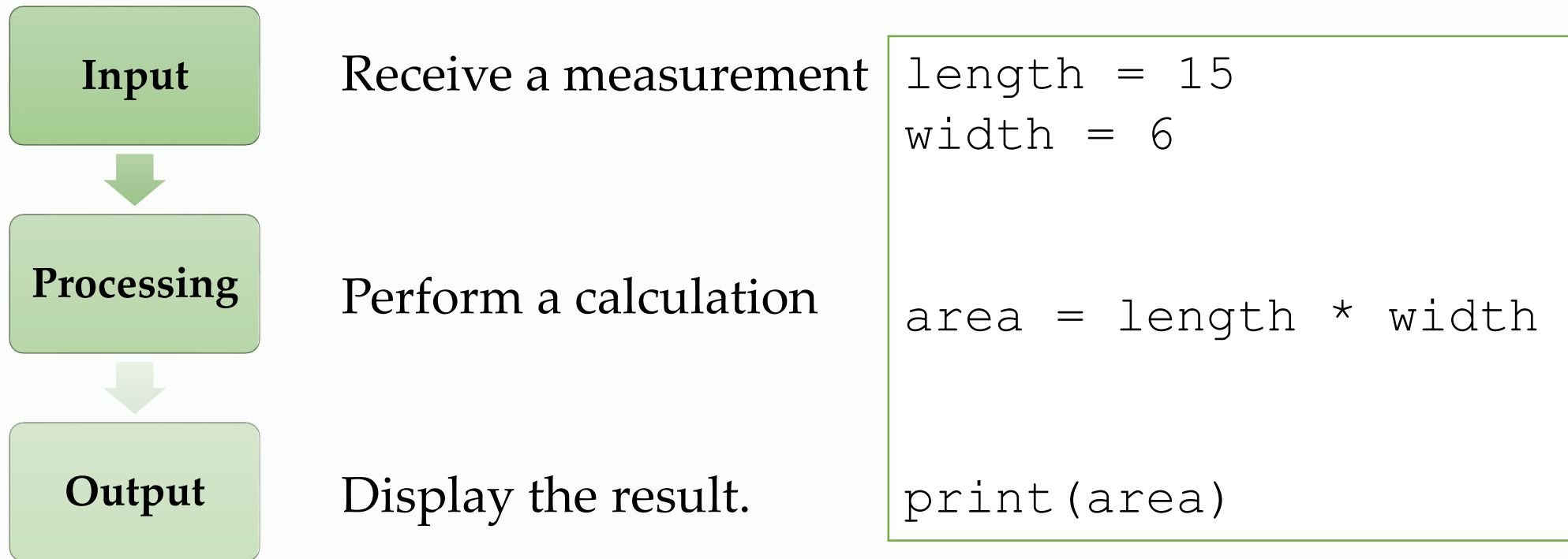
2nd year Engineer - University of Jijel

Lecture 2

Python Basics

1. Introduction

A program usually performs three fundamental tasks:



2. Basic arithmetic

Python uses the standard arithmetic operators:

- **Addition +**
- **Subtraction -**
- **Multiplication ***
- **Division /**
- **Exponentiation ****

```
>>> # Addition
>>> 2 + 3
5
>>> # Subtraction
>>> 2 - 3
-1
>>> # Multiplication
>>> 2*3
6
>>> # Division
>>> 3 / 2
1.5
>>> # Floor Division
>>> 3 // 2
1
>>> # Remainder
>>> 5 % 3
2
>>> # Exponentiation
>>> 3 ** 2
9
```

3. Priority of commands

The arithmetic operators have the following priority (PEMDAS):

- Parentheses first;
- Exponentiation next;
- Multiplication, Division (including floor division and remainder) next. Left to right in case of competition;
- Addition and Subtraction next. Left to right in case of competition.

```
>>> 1 + 4 / 2 * 3 ** 2
19.0
>>> (1 + 4 / 2 * 3) ** 2
49.0
>>> (1 + 4) / 2 * 3 ** 2
22.5
>>> (1 + 4) / (2 * 3) ** 2
0.13888888888888889
>>> (1 + 4 / 2) * 3 ** 2
27.0
```

4. Variables and assignment

Variables in Python are defined by an **identifier name**, which must respect the following rules:

1. The variable name must **begin with a letter** and can contain letters, digits and underscore character (_);
2. Python is **case sensitive**. Variable named **Var_1** is different from variable named **var_1**;
3. Python has reserved keywords, which cannot be used as an identifier name (**abs, max, min, print, input ...**).

5. Creating and modifying variables

A variable can be

- Created by assigning a value to it.
- Changed
- Used in expressions
- Printed

```
>>> length = 10
>>> width = 4

>>> width = 6

>>> area = length * width

>>> print(area)
```

7. Multiple assignment

Python allows several variables to be assigned in one statement.

For example:

```
length, width, height = 10, 5, 2
```

This is equivalent to:

```
length = 10
```

```
width = 5
```

```
height = 2
```

Python can also allow variables to receive the same value.

```
Length1 = length2 = length3 = 5
```

9. Augmented assignment

Suppose we have

```
speed = 100
```

and want to increase it by 10

We can write:

```
speed = speed + 10
```

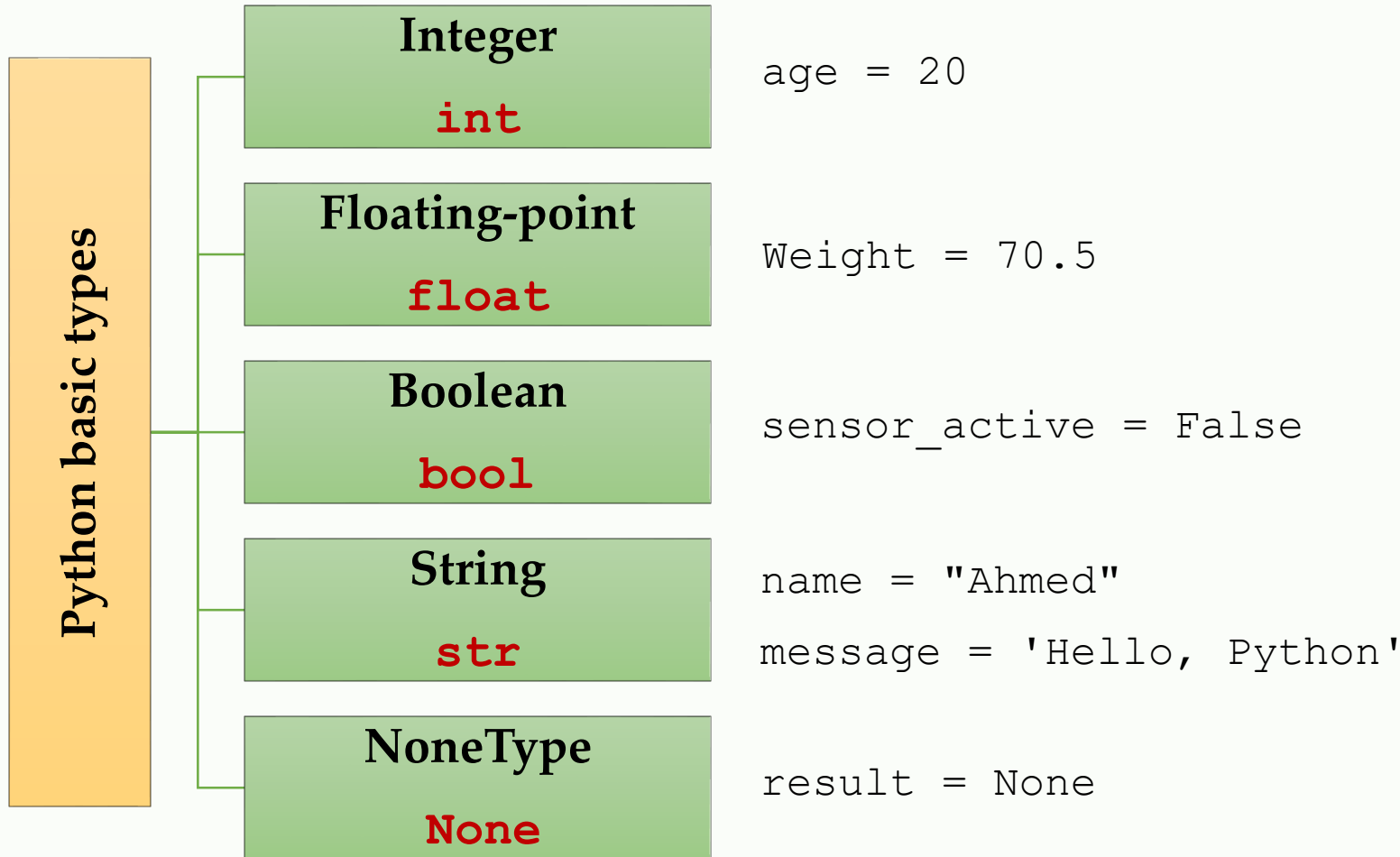
Python provides a shorter form:

```
speed += 10
```

```
>>> speed = 100
>>> speed += 10
>>> print(speed)
110
>>> speed *= 2
>>> print(speed)
220
>>> speed -= 200
>>> print(speed)
20
>>> speed /= 5
>>> print(speed)
4
```

10. Data types

In Python, the assigned data to a variable have different types.



Strings can contain **letters**, **numbers**, **spaces**, **punctuation**, and other characters.

```
code = "MEC@10-5"
```

The **+** operator can also be used to join strings

```
>>> "Ahmed " + "Ali"  
  
'Ahmed Ali'
```

11. Checking a data type with `type()`

The built-in function `type()` can be used to inspect the **type** of a value.

```
>>> type(100)
<class 'int'>
>>> type(3.5)
<class 'float'>
>>> type(3.0)
<class 'float'>
>>> type("Python")
<class 'str'>
>>> type(True)
<class 'bool'>
>>> type(None)
<class 'NoneType'>
```

12. Type Conversion

Python provides built-in functions to **convert a value from one type to another.**

The most important ones at this level are:

```
int()  
float()  
str()  
bool()
```

```
>>> # Converting to int  
>>> int(7.8)  
7  
>>> int("25")  
25  
>>> int(True)  
1  
>>> # Converting to float  
>>> float(5)  
5.0  
>>> float("3.14")  
3.14  
>>> float(True)  
1.0
```

12. Type Conversion

Python provides built-in functions to **convert a value from one type to another.**

The most important ones at this level are:

```
int()  
float()  
str()  
bool()
```

```
>>> # Converting to str  
>>> str(25)  
      '25'  
>>> str(3.14)  
      '3.14'  
>>> str(True)  
      'True'  
>>> # Converting to bool  
>>> bool(1)  
      True  
>>> bool(0)  
      False
```

12. Comparison expressions and Boolean results

The main **relational operators** in Python are:

>, <, >=, <=

== (equality)

!= (inequality)

The logical operators are:

and

or

not

```
>>> 5 < 8
True
>>> (5 >= 8) and (5 != 4)
False
>>> (5 >= 8) or (5 != 4)
True
>>> not (True)
False
>>> 5 == 5
True
>>> 5 = 5
SyntaxError: cannot assign to
literal here. Maybe you meant
'==' instead of '='?
```

13. Displaying information: `print()`

The `print()` function **displays** information on the screen.

```
>>> print ("Hello")
Hello
>>> print (25)
25
>>> print (3 + 2)
5
>>> length = 5
>>> width = 3
>>> print (length, width)
5 3
>>> print (length,          width)
5 3
>>> print (length * width)
15
>>> print ("length * width")
length * width
```

14. Formatted Output

Correct results are important, but **clear presentation** of the displayed outputs or results is also important.

Suppose:

```
>>> temperature = 20.57892
```

```
>>> print(temperature)
```

```
20.57392
```

What about:

```
20.57
```

```
20.574
```

```
Temperature = 20.57
```

```
Temperature = 20.57 C°
```

14. Formatted Output

1. `format()`

2. `str.format()`

3. **f-Strings** Modern method

```
>>> temperature = 20.57392
>>> print(temperature)
20.57392
>>> print(f" {temperature:.2f} ")
20.57
>>> print(f" {temperature:.3f} ")
20.574
>>> print(f" Temperature = {temperature:.2f} ")
Temperature = 20.57
>>> print(f"Temperature = {temperature:.2f} C°")
Temperature = 20.57 C°
```

16. Getting data with `input()`

The `input()` function allows a program to receive information from the user.

```
>>> name = input("Enter your name: ")
```

```
Enter your name: Mohammed
```

```
>>> print(name)
```

```
Mohammed
```

```
>>> print("Hello", name)
```

```
Hello Mohammed
```

**The value
entered by the
user is stored in**

name

Mohammed

16. Getting data with `input()`

The `input()` function allows a program to receive information from the user.

```
>>> age = input("Enter your age: ")
```

```
Enter your age: 20
```

```
>>> print(age)
```

```
20
```

```
>>> type(age)
```

```
<class 'str'>
```

**the value
returned by
`input()` is still
a string**

16. Getting data with input()

The `input()` function allows a program to receive information from the user.

If we need an integer we use:

```
age = int(input("Enter your age: "))
```

If we need an decimal number :

```
temperature = float(input("Enter the temperature: "))
```

17. Input → Processing → Output

Many simple programs follow this structure:

Input → Processing → Output

Consider a rectangle-area program.

Input

```
length = float(input("Enter the length: "))
```

```
width = float(input("Enter the width: "))
```

Processing

```
area = length * width
```

Output

```
print(area)
```

18. Input → Processing → Output

Many simple programs follow this structure:

Input → Processing → Output

Complete program:

```
length = float(input("Enter the length: "))  
width = float(input("Enter the width: "))  
  
area = length * width  
  
print(area)
```

19. Basic built-in functions

Python provides many **built-in functions** that are available without importing a module.

Some useful functions at this level are:

`abs()`, `max()`, `min()`, `pow()`, `round()`.

We have already encountered

`type()`, `int()`, `float()`, `str()`, `bool()`, `print()`, `input()`.

19. Basic built-in functions

1. **abs()** : returns the absolute value

2. **max()** and **min()** :

max() returns the largest value

min() returns the smallest.

1. **pow()** : calculates a power of a value

2. **round()** : used to round a number

```
>>> abs(-7)
7
>>> max(5, 3, 10, 2)
10
>>> min(5, 3, 10, 2)
2
>>> pow(3, 2)
9
>>> round(3.14159)
3
>>> round(3.14159, 2)
3.14
>>> round(3.14159, 3)
3.142
```

20. The math Module

Python provides many mathematical functions through the module **math**.

To use it, we write:

```
import math
```

We can then use functions provided by the module including:

```
sqrt(), sin(), cos(), tan(), log(), exp()
```

The general form is:

module_name.function_name()

`math.sqrt()`

20. The math Module

Mathematical Constants

`math.pi`

Square Root

`math.sqrt()`

Trigonometric Functions

`math.sin()`, `math.cos()`, `math.tan()`

Other Mathematical Functions

`math.log()`, `math.exp()`, `math.radians`

An important point is that Trigonometric functions use radians, not degrees.

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.sqrt(25)
5.0
>>> angle = math.radians(90)
>>> math.sin(angle)
1.0
```

21. Comments

When writing a script, it is meaningful to introduce some comments that **clarify how the task is accomplished** and **give the user some insights regarding the developed algorithm**.

The simple way to document the script is to use the **#** character before the comment line.

```
# Convert the temperature from Celsius to Kelvin
temperature_kelvin = temperature_celsius + 273.15

temperature = 25  # Temperature in degrees Celsius
```

The comments are ignored
when executing the script.

21. Comments

Program with comments

```
# Reading the length and the width
length = float(input("Enter the length: "))
width = float(input("Enter the width: "))

area = length * width # Rectangle area formula

# Printing the results
print(area)
```

22. Common Beginner Mistakes

1. Forgetting that `input()` returns a string

```
age = input("Enter your age: ")
```

this code stores a string. If numerical processing is required, we use:

```
age = int(input("Enter your age: "))
```

2. Using the wrong division operator

Remember that `7 / 2` produces `3.5` while `7 // 2` produces `3`.

3. Unexpected results from integer conversion

Consider `int(3.9)`, The result is `3` and it does not round to `4`.

If rounding is required, use `round(3.9)`.

4. Using unclear variable names

Using `temperature = 25` communicates more information compared to `x = 25`

Practice