

Prepared by Eng. Eman Hamdy

Introduction to Information Systems – IT111
Python Programming Language

Lecture10



Lec
10

What is Programming?

1. Just like we humans can understand a few languages (Arabic, English, etc.), so is the case with computers.
2. Computers understand instructions that are written in a specific syntactical form called a programming language.
3. Simply, programming is a way to tell computers to do a specific task.
4. Some of the popular Programming languages are Python, C, C++, Java, etc

Lec
10

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

- It is used for:
 1. web development (server-side),
 2. software development,
 3. system scripting.

Lec
10

Why Python?

1. Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
2. Python has a simple syntax similar to the English language.
3. Python has syntax that allows developers to write programs with fewer lines than some other programming languages.

Lec
10

Python Syntax compared to other programming languages

1. Python was designed for readability and has some similarities to the English language with influence from mathematics.
2. Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
3. Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Lec
10

High-Level Languages Python Success

- From IEEE spectrum **2017**: programming languages ranking

Language Rank	Types	Spectrum Ranking
1. Python	 	100.0
2. C	  	99.7
3. Java	  	99.5
4. C++	  	97.1
5. C#	  	87.7
6. R		87.7
7. JavaScript	 	85.6
8. PHP		81.2
9. Go	 	75.1
10. Swift	 	73.7

Lec
10

Variables

- In a Python program, **variables** are used to **store values**.
- A **variable** is a **storage location in a computer program**. Each variable has a **name** and holds a **value**
- A variable is similar to a parking space in a parking garage. The parking space has an identifier (as “J 053”), and it can hold a vehicle.



Like a variable in a computer program, a parking space has an identifier and a contents.

Javier Larrea/Agé Fotostock.

Lec
10

Variables

Variable Names

- A variable is defined when you give it a name that explains its purpose.
- Variable Name must follow a few simple rules:
 - Names must start with a letter or the underscore (`_`) character, and the remaining characters must be letters, numbers, or underscores
 - Cannot use other symbols such as `?` or `%`.
 - Spaces are not permitted inside names either.
- Names are **case sensitive**, that is, **canVolume** and **canvolume** are different names.
- You cannot use **reserved words** such as **if** or **class** as names; these words are reserved exclusively for their special Python meanings

Lec
10

Variables

- The left-hand side of an assignment statement consists of a variable.
- The right-hand side is an expression that has a value. That value is stored in the variable.
- Example: ***cansPerPack = 6*** here a variable name *canPerPack* will be equal to 6 after running this assignment statement
- **The first time a variable is assigned a value, the variable is created and initialized with that value.**

Lec
10

Variables

- After a variable has been defined, it can be used in other statements. E.g., **print(cansPerPack)**
- It will print the value stored in the variable cansPerPack
- If an existing variable is **assigned a new value**, that value **replaces** the previous contents of the variable. Example:

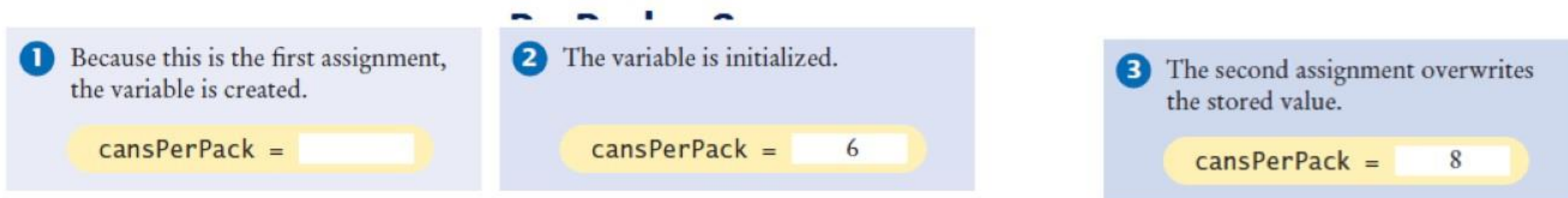


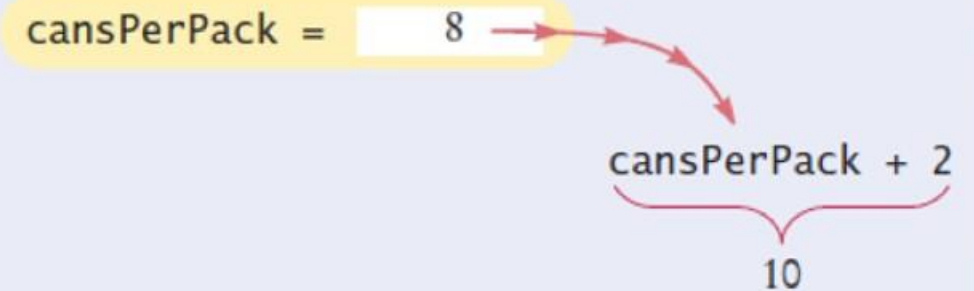
Figure 1 Executing Two Assignments

Lec
10

Variables

- Consider this example:
 $\text{cansPerPack} = \text{cansPerPack} + 2$
- This statement means to look up the value stored in the variable *cansPerPack*, add 2 to it, and place the result back into *cansPerPack*.

- 1 Compute the value of the right-hand side



- 2 Store the value in the variable

$\text{cansPerPack} = 10$

Lec
10

Variables

Assignment Statement for multiple variables:

```
a, b, c = 5, 3.2, "Hello"
```

```
>>> a
```

```
5
```

```
>>> b
```

```
3.2
```

```
>>> c
```

```
'Hello'
```

Lec
10

Data Types

- Computers manipulate data values that represent information, and these values can be of different types.
- Each value in a Python program is of a specific type.
- The **data type of a value determines how:**
 - **The data is represented in the computer**
 - **What** operations can be performed on that data.

Lec
10

Data Types

Integer Data Type (wrote as **int** in Python)

- Represents an **integral value** without a fractional part.
- For example, there must be an integer number of cans in any pack of cans—you cannot have a fraction of a can.

Floating Data Type (wrote as **float** in Python)

- Represents a **fractional part** is required.
- For example, number *0.355 is a floating number.*

String Data Type (wrote as **str** in Python)

- A sequence of characters enclosed in quotation marks “**Hello, World!**” is called a **string**. The message has to be enclosed by either by single (‘...’) or double (“...”)

Lec
10

Data Types

Table 1 Number Literals in Python

Number	Type	Comment
6	int	An integer has no fractional part.
-6	int	Integers can be negative.
0	int	Zero is an integer.
0.5	float	A number with a fractional part has type float.
1.0	float	An integer with a fractional part .0 has type float.
1E6	float	A number in exponential notation: 1×10^6 or 1000000. Numbers in exponential notation always have type float.
2.96E-2	float	Negative exponent: $2.96 \times 10^{-2} = 2.96 / 100 = 0.0296$
 100,000		Error: Do not use a comma as a decimal separator.
 3 1/2		Error: Do not use fractions; use decimal notation: 3.5.

Lec
10

Data Types

```
print("Hello")    # would print this message Hello
```

A **string** can be stored in a variable

```
greeting = "Hello"
```

```
print(greeting)  # would print this message Hello
```

Lec
10

Input and Output Variables

Input

- When a program asks for user input, it should first print a message that tells user which input is expected. Such a message is called a **prompt**.
- input function can only obtain string of text from user.
- **In Python, displaying** a prompt and reading the keyboard input is combined in one operation. For example:

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

```
Enter your first name: █
```

```
Enter your first name: Rodolfo█
```

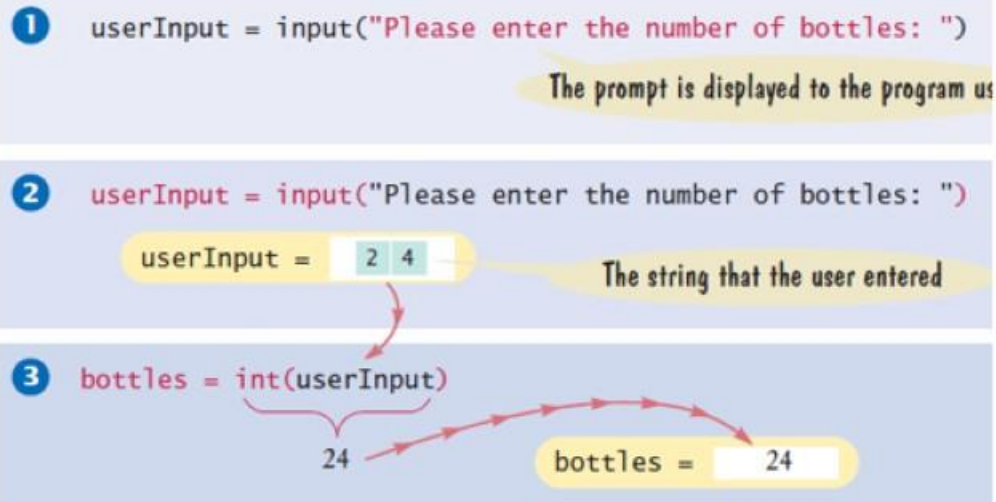
Lec
10

Input and Output Variables

- To read an integer value, first use the `input` function to obtain the data as a string, *then convert it to an integer using the `int` function*. For example:

```
userInput = input("Please enter the number of bottles: ")
```

```
bottles = int(userInput)
```



Lec
10

Input and Output Variables

- To read a floating-point value from the user, the same approach is used, except *the input string has to be converted to a float.*

```
userInput = input("Enter price per bottle: ")  
price = float(userInput)
```

- The previous examples could be written as

```
bottles = int( input("Please enter the number of bottles: ") )  
price  = float( input("Enter price per bottle: ") )
```

Lec
10

Input and Output Variables

Output

Is used to display messages and results to the user.

```
print("Hello, world")    # display  Hello, world
```

```
print(2+3)               # display  5
```

```
print("2+3=", 2+3)      #display   2+3= 5
```

Lec
10

Arithmetic operators

Basic Arithmetic Operators

- Addition $+$ $(5 + 2 = 7)$
- Subtraction $-$ $(5 - 2 = 3)$
- Multiplication $*$ $(5 * 2 = 10)$
- Float Division $/$ $(5 / 2 = 2.5)$
- Integer Division $//$ $(5 // 2 = 2)$ (floor division)
- Power $**$ $(5 ** 2 = 25)$
- Reminder $\%$ $(5 \% 2 = 1)$

Lec
10

Basic Arithmetic operators

$$b \times \left(1 + \frac{r}{100}\right)^n \rightarrow b * (1 + r / 100) ** n$$

Table 3 Floor Division and Remainder

Expression (where $n = 1729$)	Value	Comment
$n \% 10$	9	For any positive integer n , $n \% 10$ is the last digit of n .
$n // 10$	172	This is n without the last digit.
$n \% 100$	29	The last two digits of n .
$n \% 2$	1	$n \% 2$ is 0 if n is even, 1 if n is odd (provided n is not negative)
$-n // 10$	-173	-173 is the largest integer ≤ -172.9 . We will not use floor division for negative numbers in this book.

Lec
10

Operators' precedence

Operator	Description
()	Parenthesis
**	Exponentiation
*, /	Multiplication, Float Division
+, -	Sign positive, sign negative

Lec
10

Operators' precedence

```
a = 20  
b = 10  
c = 15  
d = 5  
e = 0
```

```
e = (a + b) * c / d      #( 30 * 15 ) / 5  
print "Value of (a + b) * c / d is ", e
```

```
e = ((a + b) * c) / d    # (30 * 15 ) / 5  
print "Value of ((a + b) * c) / d is ", e
```

```
e = (a + b) * (c / d);    # (30) * (15/5)  
print "Value of (a + b) * (c / d) is ", e
```

```
e = a + (b * c) / d;      # 20 + (150/5)  
print "Value of a + (b * c) / d is ", e
```

When you execute the above program, it produces the following result

```
Value of (a + b) * c / d is 90  
Value of ((a + b) * c) / d is 90  
Value of (a + b) * (c / d) is 90  
Value of a + (b * c) / d is 50
```

Lec
10

Strings

- A **string** is a **sequence of characters**. For example, the string "Hello" is a sequence of five characters.

`print("Hello")` String is "Hello"

`greeting = "Hello"` String is "Hello"

`print(greeting)` will print "Hello"

- We use double quotation marks "" around strings because this is a common convention in many other programming languages. However, interactive Python interpreter displays strings with single quotation mark ''

Lec
10

Strings

- A **string literal** denotes a particular string (such as "Hello"). In Python,
- String literals are specified by enclosing a sequence of characters within a matching pair of either single or double quotes.

```
>>> message = 'He said "Hello"'
>>> print(message)
He said "Hello"
```

```
>>> question='What is your name?'
>>> print(question)
What is your name?
>>> question="What is your name?"
>>> print(question)
What is your name?
```

Lec
10

Strings

- **Length, concatenation and repetition:**

Number of characters in a string is called the *length of the string*. For example, the length of "Harry" is 5

```
length = len("World!") # length is 6
```

- In Python, use **+** **operator** to **concatenate** two strings. For example,

```
firstName = "Harry"  
lastName = "Morgan"  
name = firstName + lastName
```



results in the string

"HarryMorgan"

- You can produce a string that is the result of **repeating a string multiple times**.
 - For example, suppose you need to print a dashed line. Instead of specifying a literal string with 10 dashes, you can use the ***** operator to create a string that is comprised of the string "-" repeated 10 times. For example

```
>>> dashes = "-" * 10  
>>> print(dashes)
```

Lec
10

Strings

- **Length, concatenation and repetition:**

What if you'd like the first and last name **separated by a space**?

```
name = firstName + " " + lastName
```

This statement concatenates three strings: first Name, the string literal " ", and lastName.

The result is ***Harry Morgan***

- A string of any length can be repeated using the * operator. For example, the statements

```
message = "Echo..."
```

```
print(message * 5)
```

The result is

Echo...Echo...Echo...Echo...Echo...

Lec
10

Strings

- Access Characters within Strings

Strings are sequences of **Unicode characters**. You can access the **individual characters of a string based on their position within the string**. This position is called the *index of the character*.

W O R D

A string is a sequence of characters.

For example:

H	a	r	r	y
0	1	2	3	4

H	a	r	r	y
0	1	2	3	4

first = H last = y

Lec
10

Math Library

- Python contains a standard library that can be used to create powerful programs.
- A **library is a collection of code that** has been written and translated by someone else, ready for you to use in your program.
- A **standard library is a library that is considered part of the language and must be** included with any Python system.
- Python's standard library is organized into **modules. Related functions and data** types are grouped into the same module.

Lec
10

Math Library

- Python's math module includes a number of mathematical **functions**. To use any function from this module, you must first **import the function**.
- To include all functions defined in math, write **import math** at the top of the python code.
- Or, to include certain function inside math (as **sqrt**), write **from math import sqrt**
- To **call** sqrt function, write **y = sqrt(x)**

Lec
10

Math Library

Table 5 Selected Functions in the math Module

Function	Returns
<code>sqrt(x)</code>	The square root of x . ($x \geq 0$)
<code>trunc(x)</code>	Truncates floating-point value x to an integer.
<code>cos(x)</code>	The cosine of x in radians.
<code>sin(x)</code>	The sine of x in radians.
<code>tan(x)</code>	The tangent of x in radians.
<code>exp(x)</code>	e^x
<code>degrees(x)</code>	Convert x radians to degrees (i.e., returns $x \cdot 180/\pi$)
<code>radians(x)</code>	Convert x degrees to radians (i.e., returns $x \cdot \pi/180$)
<code>log(x)</code> <code>log(x, base)</code>	The natural logarithm of x (to base e) or the logarithm of x to the given <i>base</i> .

Lec
10

Math Library

Table 4 Built-in Mathematical Functions

Function	Returns
<code>abs(x)</code>	The absolute value of x .
<code>round(x)</code> <code>round(x, n)</code>	The floating-point value x rounded to a whole number or to n decimal places.
<code>max(x₁, x₂, ..., x_n)</code>	The largest value from among the arguments.
<code>min(x₁, x₂, ..., x_n)</code>	The smallest value from among the arguments.

Lec
10

Math Library

Math Library

$Y = x + 1 * x + 2 * x + 3$

```
import math
```

```
x=10
```

```
y=math.sqrt(x+1)* math.sqrt(x+2)* math.sqrt(x+3)
```

```
print("y=",y)
```

Examples

Lec
10

```
x,y=eval(input("enter 2 numbers"))  
Sum=x+y  
avg=Sum/2  
print(avg)
```

```
#Data type  
#int 1 2 3  
#float 1.5 2,4  
#str "hello" 'hello'  
#eval 1) handle different data type  
#      2) more than one input  
  
# Command #
```

Lec
10

```
Python 3.13.1 (tags/v3.13.1:06714
51, Dec 3 2024, 19:06:28) [MSC v
.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credit
s" or "license()" for more inform
ation.
```

```
===== RESTART: C:/
Users/Alamin/Desktop/py.py =====
```

```
x= 2
y= 3
z= 3.5
```

```
File Edit Format Run Options \
x,y,z=2,3,3.5
print("x=",x)
print("y=",y)
print("z=",z)
```

Lec
10

```
Python 3.13.1 (tags/v3.13.1:0671451, Dec
3 2024, 19:06:28) [MSC v.1942 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "li
cense()" for more information.
```

```
===== RESTART: C:/Users/Ala
min/Desktop/py.py =====
aymanahmed
ayman ahmed
```

```
File Edit Format Run Op
x="ayman"
y="ahmed"
z=x+y
print(z)

e="ayman"
r="ahmed"
z=e+r
print(z)
```

Lec 10

```
Python 3.13.1 (tags/v3.13.1:0671451, Dec 3 2024, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
```

```
===== RESTART: C:/Users
/Alamin/Desktop/py.py =====
====
x= 5
y= 0
t= 1
p= 8
u= 12
.. 10
```

```
File Edit Format Run Options Window
x=5
X=10
print("x=", x)

y=8%2
t=9%2
p=2**3
print("y=", y)
print("t=", t)
print("p=", p)

u=11
u+=1 #u=u+1
print("u=", u)
```

Lec
10

Strategies for Making Faster and Better Computers

1. Pipelining

- Older CPUs executed one instruction at a time.
- Modern CPUs divide each instruction into stages and run multiple stages in parallel.
- **Benefit:** More instructions executed per second → Higher performance.

2. Multiprocessing & Parallel Processing**➤ Multiprocessing**

Multiple processors (or cores) handle different tasks.

➤ Parallel Processing

Several cores work together on the same operation to speed it up.

➤ Hyper-Threading (Intel)

Each core handles two threads at once, increasing efficiency.

Lec
10

Strategies for Making Faster and Better Computers

3. Improved Materials for CPUs

- Silicon is reaching its physical limits. New materials provide:
- Higher speed
- Less heat
- Lower energy consumption

New Materials:

- **Copper** – better conductor than aluminum
 - **High-k materials** – reduce leakage
 - **III-V semiconductors** (like Germanium) – faster switching
 - **Graphene** – extremely strong, super-fast, very low energy
- A graphene processor powered a device for **3 months without charging**

Lec
10

Strategies for Making Faster and Better Computers

4. 3D Chips (3D IC)

- Instead of building circuits on a single flat layer, chips are stacked vertically.
- **Benefits:**
 - ✓ Higher storage capacity
 - ✓ Faster communication between layers
 - ✓ Lower power consumption
 - ✓ Used in Intel's 3D Tri-Gate and 3D NAND SSDs

Lec
10

Terascale & Exascale Computing

- **Terascale:** Trillions of operations per second
- **Exascale:** Quintillions of operations per second
- Intel chips with 72 cores perform **3 teraflops**.
- Used in:
 - ✓ AI training
 - ✓ Climate simulation
 - ✓ Scientific modeling

Lec
10

Latest Technologies & Recent Developments in Information Technology

1- Artificial Intelligence (AI) & Generative AI

- AI is now the most important technology in IT.
- It can:
 - ✓ Understand language (ChatGPT)
 - ✓ Generate images and videos
 - ✓ Detect diseases in healthcare
 - ✓ Drive cars (self-driving systems)

AI makes computers “think” and solve problems like humans.

Lec
10

Latest Technologies & Recent Developments in Information Technology

2- Cloud Computing

- Cloud computing means using servers on the internet instead of your own device.
- Examples: **Google Cloud, AWS, Microsoft Azure**
- Benefits:
 - ✓ Access your files anywhere
 - ✓ No need for expensive hardware
 - ✓ Used by companies, apps, and websites

3- Optical Computing & Silicon Photonics

- Uses **light** instead of electricity to process data.
- Advantages:
 - ✓ Ultra-fast speed
 - ✓ Very low heat
 - ✓ Very high bandwidth
 - ✓ Used today in supercomputers and data centers.



Thank You