Python Programming: An Introduction to Computer Science

Seminar 1

Chapter 1 Computers and Programs Chapter 2 The Software Development Process Chapter 3 Computing with Numbers

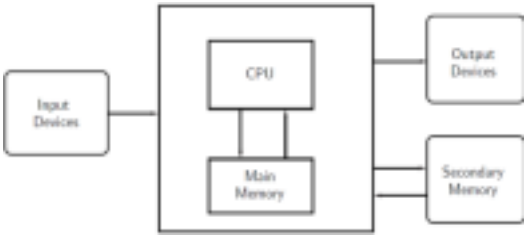
The Universal Machine Objectives

- To understand the roles of hardware and software in a computing system.
- To understand the software development process
- To begin using the Python programming language.
- To program with numbers

Hardware Basics

- Computer - hardware
- stores and manipulates information
- Computer program - software
- step-by-step set of instructions
- The process of creating software is called programming
- rule the hardware

~~"brain" of a computer.~~
carries out all the basic operations on the data.



to computer

stores

Present
processed information

Pass information

task.

programs (instructions) and data (information manipulated by program)

- makes computer performs a different set of actions or a different

Programming Languages

- Natural language
  - Ambiguous and imprecise
- Programming language
  - Unambiguous and precise
  - Every structure has a precise form, called its syntax
  - Every structure has a precise meaning, called its semantics.

Programming Languages

- High-level computer languages
  - Understood by humans
  - E.g., **c = a + b**
- Low-level or machine language
  - in 0s and 1s or mnemonics
  - Usually 1:1 mapping to computer hardware
  - E.g.,

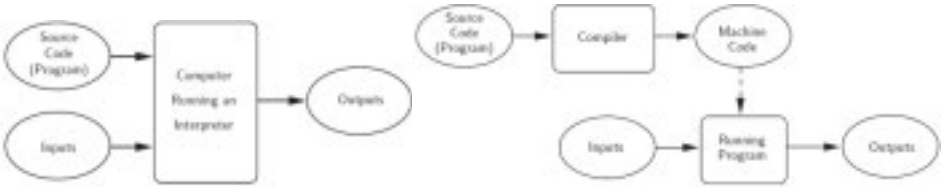
Load the number from memory location 2001 into the CPU Load the number from memory location 2002 into the CPU Add the two numbers in the CPU Store the result into location 2003

- Compilers convert programs written in a high-level language into the machine language of some computer.

Programming Languages

- Interpreters simulate a computer
  - analyzes and executes the source code instruction by instruction.

Programming Languages



Programming Languages

Once compiled, it can be executed over and over without the source code or compiler.	If it is interpreted, the source code and interpreter are needed each time the program runs
Generally run faster since the translation of the source code happens before the execution.	Generally run slower since the translation of the source code happens during the execution.
Not portable. Executable code produced from a compiler won't run on a different platform, without recompiling	More portable. If a suitable interpreter already exists, the interpreted code can be run with no modifications.

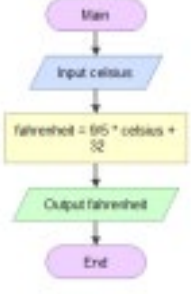
Stages of Software Development

- **Analyze the Problem**
  - What problem
    - The temperature is given in Celsius, user wants it expressed in degrees Fahrenheit."
- **Determine Specifications**
  - What your program will do.
    - Input – temperature in Celsius
    - Output – temperature in Fahrenheit
    - how they relate to one another.
      Output = 9/5(input) + 32

Stages of Software Development

- **Create a Design**
  - Formulate the overall structure of the program.
  - The how of the program gets worked out.
  - Develop an algorithm that meets the specifications.
    - Input, Process, Output (IPO) Structure
    - Pseudocode or Flowchart

Algorithm is a design of a solution. It defines the structure and the steps to solve a problem.



Algorithm

Input	**Input celsius**

Processing	**fahrenheit = 9/5 celsius + 32**
Output	**Output fahrenheit**

Pseudocode Flowchart

Stages of Software Development

- **Implement the Design**
  - Translate the design into a computer language.
  - In this course we will use Python.

```
#convert.py
# A program to convert Celsius temps to Fahrenheit
# by: Susan Computewell

def main():
 celsius = float(input("What is the Celsius temperature? "))
  fahrenheit = (9/5) * celsius + 32
 print("The temperature is ",fahrenheit," degrees Fahrenheit.")

main()
```


Stages of Software Development

- **Test/Debug/Run the Program**
  - Try out your program to see if it worked.

```
>>>
What is the Celsius temperature? 0
The temperature is 32.0 degrees Fahrenheit.
>>> main()
What is the Celsius temperature? 100
The temperature is 212.0 degrees Fahrenheit.
>>> main()
What is the Celsius temperature? -40
The temperature is -40.0 degrees Fahrenheit.
>>>
```

  - Errors (bugs) need to be located and fixed. This process is called debugging.
  - Three types of errors:
    - Compile/Syntax, Runtime, Logic

Stages of Software Development Python

- **Maintain the Program**
  - Continue developing the program in response to the needs of your users.

- In the real world, most programs are never completely finished – they evolve over time.
- The design of the program affects its maintainability.

- Created by Guido van Rossum
- Released in 1991

- Multiple programming paradigms: object-oriented, imperative, functional and procedural
- Large and comprehensive standard library
- Current version 3.7.0

Python Construct - Output

OUTPUT statement:

print(expr$_1$, ..., expr$_n$, end = "")

```
>>> print("Hello, world")
Hello, world
>>> print(2+3)
5
>>> print("2+3=", 2+3)
2+3= 5
```

- With end Output:

```
print("3+4=", end = "")
print(3+4)  3+4=7
```


Define FUNCTION: def func(arg$_1$, ..., arg$_n$): <body>

Python Construct – Function

- Making a new command:

Define FUNCTION: def func(): <body>

```
>>> def hello():
        print("Hello")
        print("Computers are Fun")

>>> hello()
Hello
Computers are Fun
```


Python Construct - Function

- Making a new command with parameters (or arguments):

```
>>> greet("Terry")
```

Python Construct - Input

INPUT statement: input (prompt)

- First the prompt is printed
- The `input` part waits for the user to enter a value and press <enter>
- The value entered is treated as a string of characters

```
>>> def greet(person):
 print("Hello",person)  print ("How are you?")

>>>
Hello Terry
How are you?
>>> greet("Paula") Hello Paula
How are you?
>>>

>>> friend = input('Enter name of person to greet: ') Enter name of friend to greet: Alan
>>> greet(friend)
Hello Alan
How are you?
```


Python Construct - Assignment

ASSIGNMENT statement: var = expr

- The value obtained from evaluating expr is assigned to the variable.

A variable is used to assign a name to a value so that we can refer to the value later.

A variable begins to exist when a value assigned to it.

```
>>> friend = input('Enter name of person to greet:
') Enter name of friend to greet: Alan
>>> greet(friend) Hello Alan
How are you?
friend Alan
```


Python Construct - Assignment

```
x = x + 1
```

Once the value on the RHS is computed, it is stored back into (assigned) into x

Simultaneous ASSIGNMENT statement:

$var_1, ..., var_n = expr_1, ..., expr_n$

```
sum, diff = x+y, x-y
```

- Evaluate the expressions in the RHS and assign them to the variables on the LHS

```
x, y = y, x
```

What does this statement do?

Python Construct – Selection

Selection statement: if condition: <true-body>

- A selection tells Python to perform the

Python Construct - Assignment

true-body if the condition is true.

```
if celsius < -273:
```

```
print(celsius, "is invalid")
```
Python

Python Construct - Loop

LOOP statement: for var in exprList:
<body>

- A loop tells Python to repeat the same thing over and over.

```
for i in range(10):
    x = 3.9 * x * (1 - x)
print(x)
```

```
x = 3.9 * x * (1 - x) print(x)
x = 3.9 * x * (1 - x) print(x)
x = 3.9 * x * (1 - x) print(x)
x = 3.9 * x * (1 - x) print(x)
x = 3.9 * x * (1 - x) print(x)
```


Python Construct - Loop

```
for i in range(10):
    x = 3.9 * x * (1 - x)
print(x)
```

equivalent to

```
x = 3.9 * x * (1 - x) print(x)
x = 3.9 * x * (1 - x) print(x)
x = 3.9 * x * (1 - x) print(x)
x = 3.9 * x * (1 - x) print(x)
x = 3.9 * x * (1 - x) print(x)
```


Python Construct - Comment

Comment: # comment

- Lines that start with #
- Intended for human readers and ignored by Python
- Python skips text from # to end of line

```
# File: chaos.py
# A simple program illustrating chaotic behavior
```

Elements of Programs

- Identifiers
  - Names to identify variables (celsius, fahrenheit), functions (main, convert), etc.
  - Naming convention
    - begin with a letter or underscore (“_”)
    - followed by any sequence of letters, digits, or underscores
    - case sensitive.

Elements of Programs

- Keywords
  - Identifiers that are part of Python itself. are known as reserved words (or keywords). not available for you to use as a name for a variable, etc. in your program.
    - and, del, for, is, raise, assert, if, in, print, etc. For a complete list, see Table 2.1 (p. 32)
  - Meanings already assigned

Elements of Programs

- Expressions
  - Evaluated to value e.g., `x = 3.9 * x * (1 - x)`
  - May include:
    - Literals are used to represent a specific value, e.g.
      - number literals 3.9, 1, 1.0 or
      - string literals (like "Hello" and "Alan")
    - Identifiers such as variables or functions
    - Operators and function calls:
      - +, -, *, /, //, **, %, abs
      - normal mathematical precedence applies.

Numeric Data Types

- Two number data types or classes in Python
  - `int` : whole positive or negative numbers e.g., 3, -4, 0
  - `float` data type : decimal fractions

e.g., 3.0, -0.2523

Every data in Python is an object. An object has
• content (the value),
• type (the data type of the value) and • id or an identity (the address where the value is stored in memory)

```
>>> type(3)
<class 'int'> >>> type(3.0) <class 'float'> >>>
myInt = 3 >>> myInt
3
>>> type(myInt) <class 'int'> >>> id(myInt)
493790368
>>> id(3)
>>> x = 10 >>> id(x)
```

Numeric Data Types

- `int` and `float` are immutable

Values of immutable data types cannot be changed without changing the identities .

```
493790592
>>> x = x + 1 >>> x
11
>>> id(x)
493790624
>>> id(10) 493790592
>>> id(11)
```

493790624

Numeric Data Types

- Operations on `int` produce `int` operations on `float` produce `float`.

```
>>> 3.0 + 4.0
7.0
>>> 3 + 4
7
```

Integer division // produces a whole number or a float with 0 in the decimal part

```
3.3333333333333335 >>> 10 // 3
3
>>> 10.5 // 3.0 3.0
>>> 10.5 % 3.0 1.5
```

Modulus % is the remainder of the integer division.
a = a // b * b + a **%** b

```
>>> 10.0 / 3.0 3.3333333333333335 >>> 10 / 3
```


Type Conversion & Rounding

- Implicit typing
  - Python converts ints to floats in mixed-typed expressions : `3 + 4.0` evaluates to `7.0`
- Explicit typing
  - To control the type conversion.
    `3 + int(4.0)` evaluates to `7`
    Converting to an `int` simply discards the fractional part of a `float` – the value is truncated.

`int("32")` and `float("32")` evaluate to `32` and `32.0`

Type Conversion & Rounding

- `round` function
  - to the nearest whole value.
  - to another float value, if second parameter specifies the number of digits after the decimal point.

```
>>> float(22//5) 4.0
>>> int(4.5)
>>> round(3.9) 4
>>> round(3)
```

Using the Math Library

- A library is a module with useful functions, e.g., Math library
  - Importing a library makes whatever functions are defined within it available to the program, e.g., import math
- To compute the roots of a quadratic equation: $^{2}4$

$$-\pm-$$
$$b\ b\ ac$$

x_a

4

3

```
>>> int(3.9) 3
>>> round(3.1415926, 2)
3.14
```

=

2

discRoot = math.sqrt(b*b – 4*a*c)

Python Programming, 3/e 9

Using the Math Library

pi	π	An approximation of pi
e	e	An approximation of e
sqrt(x)	$\sqrt{x}$	The square root of x
sin(x)	sin x	The sine of x
cos(x)	cos x	The cosine of x
tan(x)	tan x	The tangent of x
asin(x)	arcsin x	The inverse of sine x

acos(x)	arccos x	The inverse of cosine x
atan(x)	arctan x	The inverse of tangent x

exp(x)	e^x
ceil(x)	$\lceil x \rceil$
floor(x)	$\lfloor x \rfloor$

Using the Math Library

log(x)	ln x	The natural (base e) logarithm of x
log10(x)	$\log_{10} x$	The common (base 10) logarithm of x