CS1011 Computer Programmin g in Python

LECTURE 1:INTRODUCTION TO COMPUTER SCIENCE AND PROGRAMMING IN PYTHON

Essential References

- Guttag, John. Introduction to Computation and Programming Using Python: With Application to Understanding Data Second Edition. MIT Press, 2016. ISBN: 9780262529624.
- Mark Lutz (2025). Learning Python, 6th Edition. Publisher(s):O'Reilly Media, Inc. ISBN: 9781098171308
- Zelle, J. M. (2016). Python Programming: An Introduction to Computer Science. 3rd Edition. Franklin, Beedle & Associates Inc.
- McKinney, W. (2022). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter. 3rd Edition. O'Reilly Media.
- Grus, J. (2019). Data Science from Scratch: First Principles with Python. 2nd Edition. O'Reilly Media.

Assessment

D. Students Assessment Activities

No	Assessment Activities *	Assessment timing (in week no)	Percentage of Total Assessment Score
1.	Quizzes	4-10	10%
2.	Assignment(s)	3,7,11	15%
3.	Midterm Exam	8	20%
4.	Lab	1-14	10%
5.	Lab Project	12-15	10%
6.	Final Exam	16 or 17	35%

Objectives

- Introduction to computational thinking and problem solving
- Introduction to Python and its ecosystem
- Mathematical Operations
- Basic syntax: variables, keywords, and expressions
- Data types: integers, floats, and strings
- Basic input/output (print, input)

What is Computational Thinking (CT)?

Definition	Source
‘Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out.’	(Wing, 2014)
‘The mental activity for abstracting problems and formulating solutions that can be automated.’	(Yadav et al., 2014)
‘The process of recognising aspects of computation in the world that surrounds us and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes.’	(Furbe, 2012)
’A mental orientation to formulating problems as conversions of some input to an output and looking for algorithms to perform the conversions. Today the term has been expanded to include thinking with many levels of abstractions, use of mathematics to develop algorithms, and examining how well a solution scales across different sizes of problems.’	(Denning, 2009)

The Software Development Process

- The process of creating a program is often broken down into stages according to the information that is produced in each phase.

The Software Development Process

- **Analyze the Problem**
  Figure out exactly the problem to be solved. Try to understand it as much as possible.

The Software Development Process

- **Determine Specifications**
  Describe exactly what your program will do.
  - Don’t worry about *how* the program will work, but *what* it will do.
  - Includes describing the inputs, outputs, and how they relate to one another.

The Software Development Process

- **Create a Design**
  - Formulate the overall structure of the program.
  - This is where the *how* of the program gets worked out.
  - Develop your own algorithm that meets the specifications.

The Software Development Process

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

The Software Development Process

- **Test/Debug the Program**
  - Try out your program to see if it works.
  - If there are any errors (*bugs*), they need to be located and fixed. This process is called *debugging*.
  - Your goal is to find errors, so try everything that might "break" your program!

The Software Development Process

- **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.

Example Program: Temperature Converter

- Analysis – the temperature is given in Celsius, user wants it expressed in degrees Fahrenheit.
- Specification
  - Input – temperature in Celsius
  - Output – temperature in Fahrenheit
  - Output = 9/5(input) + 32

Example Program: Temperature Converter

- Design
  - Input, Process, Output (IPO)
  - Prompt the user for input (Celsius temperature)
  - Process it to convert it to Fahrenheit using F = 9/5(C) + 32
  - Output the result by displaying it on the screen

Example Program: Temperature Converter

- Before we start coding, let’s write a rough draft of the program in pseudocode
- Pseudocode is precise English that describes what a program does, step by step.
- Using pseudocode, we can concentrate on the algorithm rather than the programming language.

Example Program: Temperature Converter

- Pseudocode:
  - Input the temperature in degrees Celsius (call it celsius)
  - Calculate fahrenheit as (9/5)*celsius+32
  - Output fahrenheit
- Now we need to convert this to Python!

Example Program: Temperature Converter

```
# Celsius to Fahrenheit Converter

# Take input from the user
celsius = float(input("Enter temperature in Celsius: "))

# Convert to Fahrenheit
fahrenheit = (celsius * 9/5) + 32

# Display the result
print(f"{celsius}°C is equal to {fahrenheit}°F")
```

Example Program: Temperature Converter

- Once we write a program, we should test it!

```
Enter temperature in Celsius: 34.5
34.5°C is equal to 94.1°F
```

What is Python?

- Python is a general-purpose programming language
- It was released in 1991
- It supports multiple programming paradigms such as object-oriented

Why Python?

Works on different operating systems

Has a simple syntax

Its code can be executed as soon as it is written

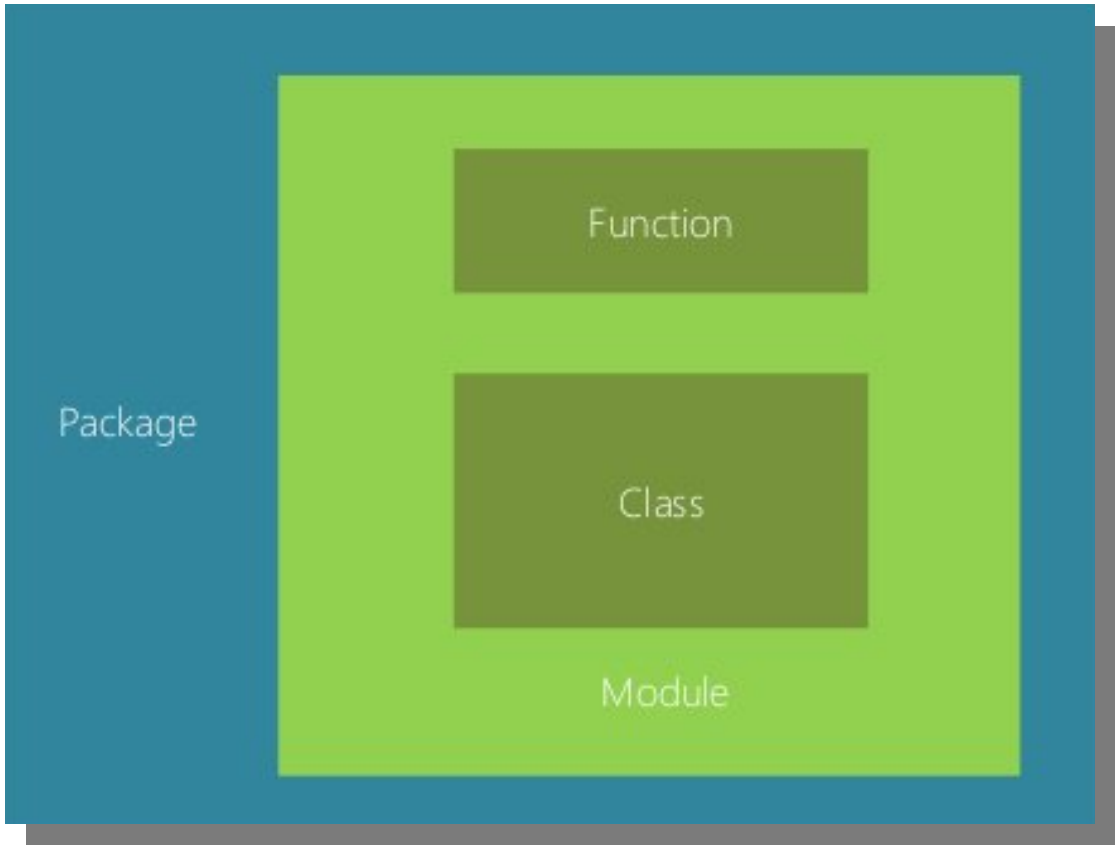
Python Components

1) Functions
2) Classes
3) Modules

   collection of functions & classes

4) Packages

   collection of modules



Mathematical Operations

Operator	Meaning	Example	Y
+	Addition	x= 10 y = x + 3	13
-	Subtraction	x= 6 y = x – 4	2
*	Multiplication	x= 2 y = x * 5	10
/	Division	x= 10 y = x / 7	1.4
%	Modulus (reminder)	x= 10 y = x % 7	3
**	Exponent	x= 3 y = x ** 3	27

Variables

- A variable is a named location in the memory that can store a value
- A variable is created when you first assign a value to it
- Variables do not need to be declared with a type & can even change type after they have been set
- A variable that has been given a value can be used in expressions
- **Syntax:**

Variables

main.py saved

```
1  x = 5
2  y = "John"
3  z = x + 3.5
4  print(x)
5  print(y)
6  print (z)
7  x = "Robert"
8  print (x)
```

```
>
5
John
8.5
Robert
>
```

Variables: Names Rules

A variable name must start with a letter or _

A variable name can only contain numbers, letters, and _

Variable names are case-sensitive (age ≠ Age ≠ AGE)

A variable name can't be a keyword (see below)

Variables: Names Rules

Variable Name	Status
1st_num	x
Buffer_20	√
_city	√
std id:	x
sum&avg	x
import	x

Constants

- A constant is a variable whose value cannot be changed
- To distinguish between a variable & a constant, programmes used to write constants in all capital letters and underscores separating the words
- Constants follow the same naming rules of variables
- **Syntax:**

  CONSTANT_NAME = value

Constants

- **In the following table, which constant names violate/conform to the naming rules & programmers style**

CONSTANT NAME	STATUS
1st_num	x
BUFFER-20	x
_CITY	√
STUDENT_NAME	√
SUM&AVG	x
RATE	√

Keywords

- Some identifiers are part of Python itself. These identifiers are known as *reserved words* (or *keywords*). This means they are not available for you to use as a name for a variable, etc. in your program.

and	as	assert	break	class	continue
def	del	elif	else	except	exec
finally	for	from	global	if	import
in	is	lambda	nonlocal	not	or
pass	raise	return	try	while	with
yield	True	False	None		

Expressions

You can produce new data (numeric or text) values in your program using *expressions*

- This is an expression that uses the *addition operator*
- This is another expression that uses the *multiplication operator*
- This is yet another expression that uses the *addition operator* but to *concatenate* (or glue)
  strings together

```
x = 2 + 3
print(x)
5
print(5 * 7)
35
print("5" + "7")
57
```

Data types: integers, floats, and strings

Data Type	Example
Integers	12
Floats	2.75
Strings	Hello friends
Booleans	one of two values: True or False

input/output (print, input)

The **input()** method asks the user for input and returns it

- Its important to assign the **input()** method to a variable to store the input data
- The **print()** function prints the specified message to a standard output device
- The **print()** uses the comma (**,**) to concatenate separate arguments passed to it, and it inserts a single space between them so that you don’t end up with a squashed message

Syntax:

- Variable_name = input(“prompt message”)

input (input)

- Prints whatever is in the quotes
- User types in something and hits enter
- Binds that value to a variable

```
text = input("Type anything... ")
print(5*text)
```

output

```
Type anything... 8
88888
```

- `input` gives you a string, so must cast if working with numbers

```
num = int(input("Type a number... "))
print(5*num)
```

output

```
Type anything... 8
40
```

Output (print)

- Used to output stuff to console
- keyword is print

```
x = 1
print(x)


x_str = str(x)
print("my fav num is", x, ".", "x =", x)


print("my fav num is " + x_str + ". " + "x = " + x_str)
```

#output

```
1
my fav num is 1 . x = 1
my fav num is 1. x = 1
```

Example

ASK FOR THE USER'S NAME AND PRINT IT

```
x  = input("Enter your name: ")
print ("name is", x)
```

```
Enter your name: daliah
name is daliah
```

Conversion Functions: int(), float()

- **int()** function: converts from a string (or a float) to an integer
- **float()** function: converts from a string (or an integer) to a float
- These functions are very useful with the **input()** method. The input() method returns the user input as string even if the data are numbers. Thus, use float() or int() after entering numeric data

Example

- **Add two numbers and print the result**

```
n1 = float(input("enter first number: "))
n2 = float(input("enter second number: "))
sum = n1 + n2
print("the result= ", sum)
```

```
enter first number: 4.5
enter second number: 7
the result=  11.5
>
```

Example

- **Ask for the user's name, age, weight and print it**

```
x  = input("Enter your name: ")
y  = int(input("Enter your age: "))
z  = float(input("Enter your weight: "))
print ("name is",x,"age is",y,"weight is",z)
```

```
Enter your name: daliah
Enter your age: 25
Enter your weight: 48.5
name is daliah age is 25 weight is 48.5
>
```