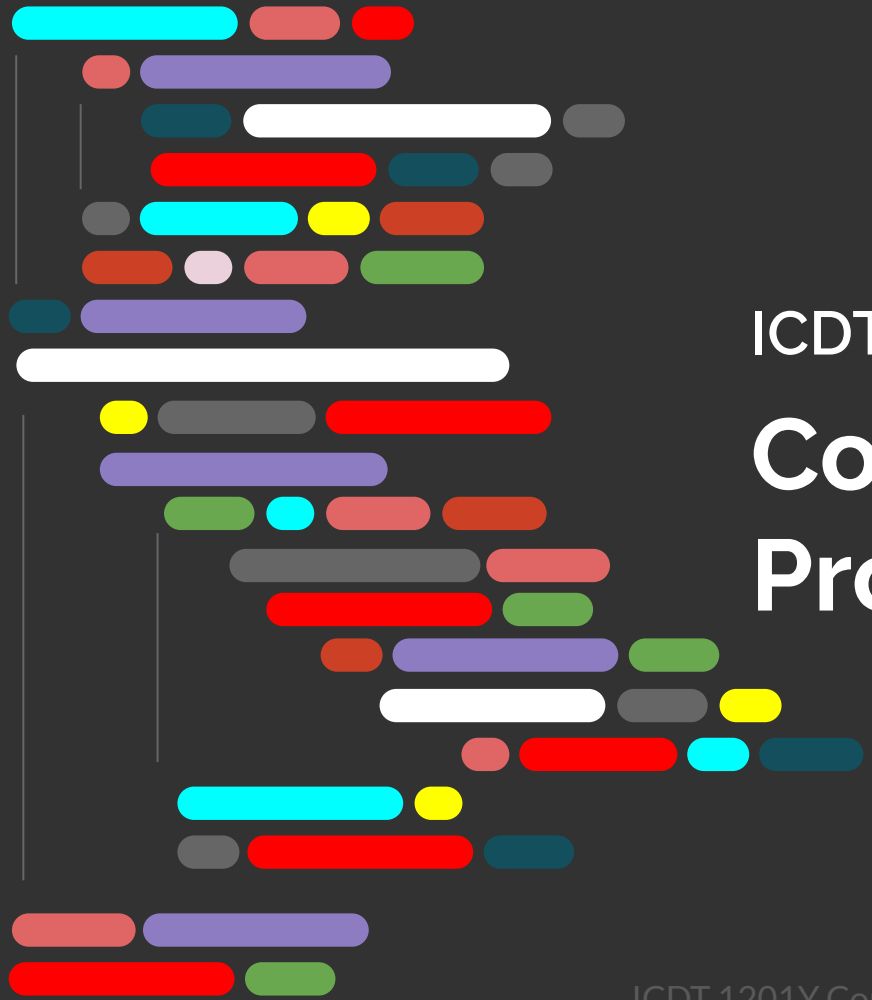
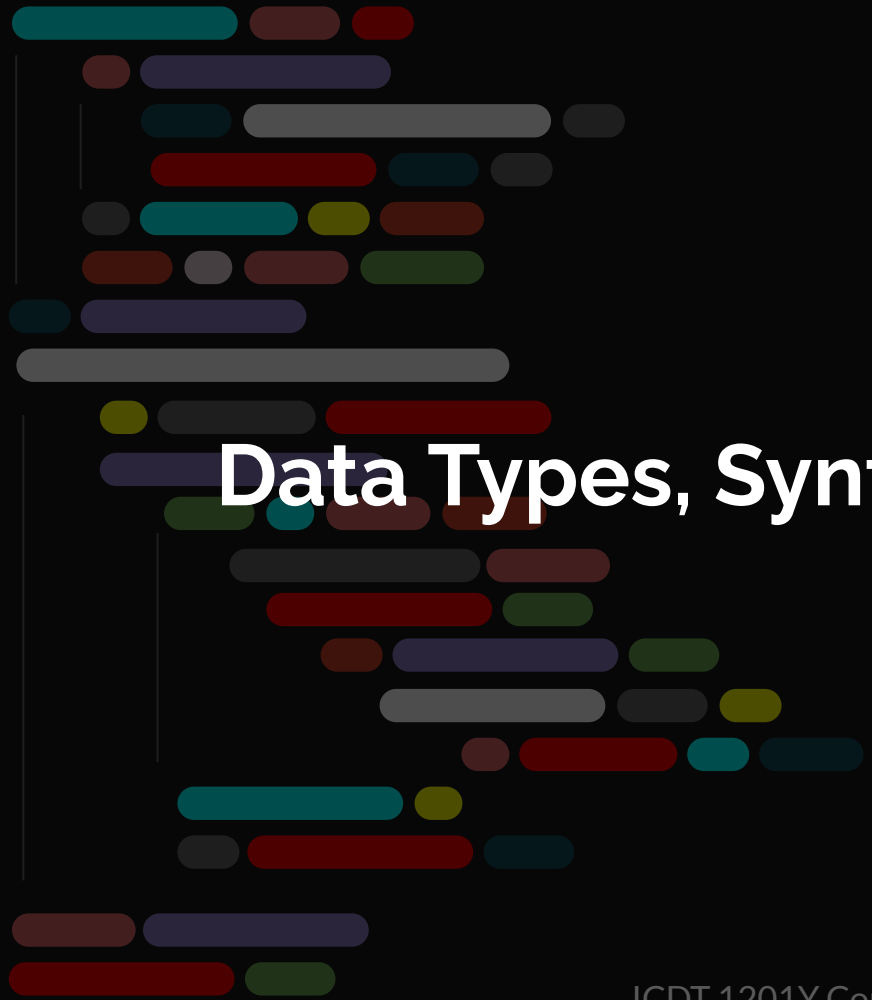


ICDT 1201Y (3)

Computer Programming (week 2)



Data Types, Syntax and Semantics

Sub-topics

- Syntax
- Semantics
- Variables
- Arithmetic Operations
- Expressions and Assignments

Learning Outcomes & Assessment Criteria

Learning Outcomes:

- Demonstrate an Understanding of Data Types, Variables, Syntax, and Semantics:
  - Explain the importance of data types in programming.
  - Identify common data types (integer, float, string, boolean) and their uses.
  - Understand the concept of variables and their role in storing data.
  - Differentiate between syntax (rules for writing code) and semantics (meaning of the code).
- Define the Key Components of a Statement in a Program:
  - Recognize the basic structure of a program

Assessment Criteria:

- Understand the concepts of literals, identifiers, and mathematical operations.
- Understand the difference between a valid and an invalid identifier.
- Distinguish between different types of data.
- Use variables and write simple expressions.
- Understand the construct of simple statements such as assignments and *function calls*.

Introduction to Python

- Python is a general-purpose programming language that can be used effectively to build almost any kind of program that does not need direct access to the computer's hardware.
- Easy to learn, powerful programming language
- It has efficient data structures
- It is "interpreted"!
- Other characteristics: interactive, portable, extensible, scalable, has a large standard library of built-in functions

Python Programs

- A Python program, sometimes called a script, is a sequence of definitions and commands.
- A **program** is a sequence of definitions and commands
  - definitions evaluated
  - commands executed by Python interpreter in a shell
- **commands** (**statements**) instruct interpreter to do something
  - can be typed directly in a shell or stored in a file that is read into the shell and evaluated

Data Types in Programming

- What are data types?
  - Kinds of data that a ***variable*** can hold.
  - Common Data types in Python (primitive):
    - Integer (int)- Whole numbers[8,-4,0]
    - Float (float): Decimal numbers [3.142, -34.5]
    - String (str): Sequence of characters/ Text [“Hello”, “24”, “Bonjour Moris”]
    - Boolean (bool): True or False values
    - There are more complex data types which

- Data Type Analogy in Real-life
  - Liquids….stored in bottles
  - Large Vegies - Potatoes & Onions….stored in ‘net-bag’? [forgive us if you do not believe these are vegies]

- **Liquids** and **Large Vegies** would be equivalent to two different ***data types***
- The **Bottles** and **‘Net-Bags’** would be the ***variables*** (containers!)

Experimenting with Data Types

- In Python, a special code (function) enable us to find out the data type of some values.
- You already know the **print()** function
- Now we will use the **type()** function [inside on the **print()**]
- Python automatically assigns the data type based on the value provided.

Python Code	Output
mark = 10 print(type(mark))	<class 'int'>
pi = 3.142 print(type(pi))	<class 'float'>
greeting = "Hello" print(type(greeting))	<class 'str'>
is_married = True print(type(is_married))	<class 'bool'>

*Take note of mark, pi, greeting, is_married…these are names/identifiers for **variables***

Type Conversions

- Sometimes values of one data type need to be converted to another
  - The float() function converts an int to a float
  - The int() function converts numbers into ints
  - The round() function rounds a float off to the nearest whole number and returns a float
  - The str() ?

Try:

- int(4.5)
- int(3.9)
- float(int(3.3))
- round(-3.5)

Understanding Syntax

- Syntax refers to the rules that define the structure of a programming language.
- Correct syntax is crucial for the program to compile and run.
- For example:
  - Code blocks/groups of code are defined with indentation
  - A statement usually end with a newline (not a full-stop or semi-column)
  - **Comments start with a '#' - What is a comment and its role in programming?**

1. I ate pizza. Tomorrow I will eat cake.
2. eat pizza i cake will eat tomorrow I

The examples above demonstrate the importance of syntax/rules… Even in normal English we need to respect it if we want our communication to be understandable.

Comments in Python

- A good way to enhance the readability of code is to add comments.
  - Text following the symbol # is not interpreted by Python. For example, one might write:

```
1  pi=3.142
2  side = 1 #length of sides of a unit square
3  radius = 1 #radius of a unit circle
4  #subtract area of unit circle from area of unit square
5  areaC = pi*radius**2
6  areaS = side*side
7  difference = areaS - areaC
```

Investigate multiple-line commenting in Python.

Understanding Semantics

- Semantics refers to the meaning of the code or what the code does when executed.
- **Syntax** is about **structure**, while **semantics** is about **meaning**.

Understanding Literals and Identifiers

- **Literals** are fixed values assigned to variables (e.g., numbers, strings).

```
1    gavin_age = 64  # 64 is a literal
2    full_name = "Gavin Hacker"  # "Gavin Hacker" is a literal
```

- **Identifiers** are names given to variables, functions, etc.
  - Must start with a letter or underscore (_).
  - Can contain letters, digits, and underscores.
  - Cannot be a reserved keyword.

Valid Identifier in Python

- **Rules for Valid Identifiers:**
  - Must start with a letter or underscore.
  - Can contain letters, digits, and underscores.
  - Cannot be a reserved keyword (e.g., def, class, if).

Examples:

Valid Identifiers

name
my_name
name2
_age
myName
NAME

Invalid Indentifiers

1stname
middle-name
def
My name
allName$

Variables in Programming

- Variables are used to store data that can be manipulated and retrieved.
- To declare a variable, We give it a unique name (identifier) and use

```
name = "Gavin"
age = 49
height = 5.9
is_lecturer = True
```

- Remember our liquids, potatoes and onions?
- We need appropriate containers to keep them
- The contents of the containers can be removed/replaced/changed
- We give names to our containers: Pot Disel / Sac Winners Nouvo… why?

Storing Data Values in Variables

- To store data values, programs make use of variables
  - Variables have a type but are never declared in Python.
  - They are instantiated when they are assigned for the first time.
  - For Example:

```
x =5
y=10
z=x+5
```

Using Variables

```
1   # Declare variables
2   name = "Gavin"
3   age = 49
4
5   # Use variables
6   print(name)  # Output: Gavin
7   print(age)   # Output: 49
8
9   # Update variables
10  age = 50 # we forgot Gavin's birthday...he is now 50
11  print(age)   # Output: 50
12
```

Variables can be updated by assigning a new value.

The value of a variable can be printed using the print() function.

Exercise 1. on Variables

- What would be the expected output if the codes below are executed?

```
1  a=5
2  b=10
3  c=a+5
4  print(a)
5  print(B)
6  print(c)
```

- Remember….Pythons are sensitive snakes…b & B are different…case-sensitive they call it.

Displaying Variables Names and Values

Consider

x = 5

What is the difference between the following two statements:

print(“x”)

print(x)

How do we display **x = 5** ?

Assigning String Values to Variables

- In the previous slides, numerical values had been assigned to variables.
- To assign non-numerical (string) values, the values should be between single or double quotes, eg.:

lecturer = “Gavin Hacker”
course= “BSc(hons) Hacking Methodologies”

- If we run the statement

print(lecturer,course)

- We will see the output

Gavin Hacker BSc(hons) Hacking Methodologies

Exercise 2. on String and Variables

- Write a program that assigns the value “**Pizza**” to the variable **menu** and “**Cheese and Capsicum**” to the variable **topping** and makes use of the variables **menu** and **topping** to display the following output:

The menu Pizza has topping Cheese and Capsicum

Arithmetic Operations

- Arithmetic operations involve basic mathematical operations.
- Common Operators:
  - Addition (+): Adds two numbers.
  - Subtraction (-): Subtracts one number from another.
  - Multiplication (*): Multiplies two numbers.
  - Division (/): Divides one number by another. (Result is always a float)
  - Modulus (%): Returns the remainder of a division.

```
1   # Addition
2   week = 5 + 2
3   print(week)  # Output: 7
4
5   # Subtraction
6   weekdays = 7 - 2
7   print(weekdays)  # Output: 5
8
9   # Multiplication
10  month = 4 * 7
11  print(month)  # Output: 28
12
13  # Division
14  weeks = 31 / 7
15  print(weeks)  # Output: 4.428571428571429
```

```
quiz = 9 % 2
print(quiz)
```

Expressions and Assignments

- An Expression is a combination of variables, literals, and operators that produces a value.
- An Assignment is the operation during which the result of an expression is assigned to a variable using the = operator.

```
1   # Declare variables
2   start_age = 24
3   expected_retirement = 65
4
5   # Expression and assignment
6   years_of_service = expected_retirement - start_age
7   print(years_of_service)  # Output: 41
8
9   # Update variable with expression
10  start_age = start_age + 3
11  print(start_age)  # Output: 27
```

Variables can store the result of expressions.
Variables can be updated with new expressions.

Complex Expressions

- More complex and interesting expressions can be constructed, by combining simple expressions, through the use of operators.

```
a=2
b=3
print(a*a + b*b)
```

Precedence in Arithmetic Operations

- The arithmetic operators have the usual mathematical precedence.

- For example, * binds more tightly than +

so the expression x+y*2 is evaluated by first multiplying y by 2 and then adding the result to x. The order of evaluation can be changed by using parentheses to group subexpressions, e.g., (x+y)*2 first adds x and y, and then multiplies the result by 2 .

Additional Practice Exercise 3

Dry-run and write down the expected outcome for each of the command below. Then in the Python shell, the commands, pressing 'Enter' after each, to verify your answer.

```
print ("Hello, World")
print (2 + 3)
print("2 + 3 =", 2 + 3)
print ("Hi! How are you?")
print ("I am now in the Computer Lab at UoM.")
print( "Hi! How are\n you?")
print ("I am now \t in the Computer Lab at UoM.")
x = 2
print( x)

x = 2
y = 3
print( x + y)
```

Additional Practice Exercise 4

Write down the expected output (afterwards you may verify on your machines)

```
1    num1 = 15
2    num2 = 25
3    num3 = 5
4    sum = num1 + num2 + num3
5    print("The sum is:" + sum)
```

String concatenation is when two or more values of type String are ‘joined’ or ‘concatenated’. The ‘+’ operator is used for this. However, Python does not allow Strings to be joined with integers directly. Investigate possible techniques for addressing this issue.

Additional Practice Exercise 5

Write a Python program that declares a string variable, assigns a value to it, and prints the length of the string.

One ‘easy’ technique would be for you to manually count the number of characters; Will this technique still work if you were provided with a huge paragraph of text?

Just like the **print()** or the **type()** functions that you are now already familiar with, there is another in-built function called **len().**

```
1  text = "Hello everybody, my name is Gavin, also known as Gavin Hacker"
2  length = len(text)
3  print("The length of the string is:", length)
```

Additional Practice Exercise 6

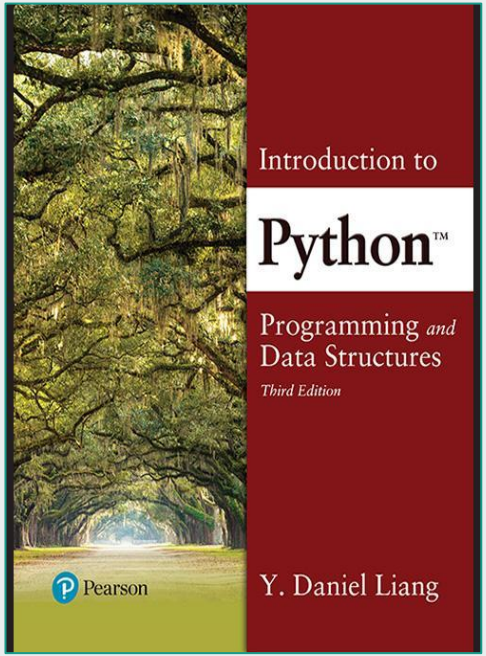
Simultaneous Assignments: Investigate how to use Simultaneous Assignments in Python through the general format:

<var>, <var>, ..., <var> = <expr>, <expr>, ..., <expr>

And look into how a swap can be performed through Simultaneous Assignments

Acknowledgements

- Slides adapted from:
- & Dr P.Appavoo's and Dr Baichoo's slides



Disclaimer: Gavin is a fictitious character used as an example….