# Programming and Database Applications for ISE

INSY 3010

Programming / Python Concepts

Dan O’Leary // dan.oleary@auburn.edu

# Class Admin

- Schedule posted
- HW1 due next Tues, Feb 4
- Testing environment – experimental
- Accommodation requests processed
- MLYC7F

# Recap

- Chapter 1
  - arithmetic operators: `+, -, *, /, //, **`
  - expressions
  - arithmetic functions
    - `round()`
    - `abs()`
  - integers, floating point, strings
  - `len()`
  - `type()`
  - `int()`, `float()`, `str()`

- Chapter 2
  - variables and assignment
    - naming rules
    - reserved keywords
  - import modules
    - "dot" (`.`) operator
    - math module: `math.pi`, `math.sqrt()`, `math.pow()`
  - expressions and statements
  - `print()`
  - arguments - single / multiple, optional / required
  - comments `# like this...`
  - error types
    - syntax - problem with code format
    - runtime - problem with code contents
    - semantic - problem with code logic

# What is Python?

Language (use), Source Code (write), and Interpreter (run)

- Language – standard syntax, provided libraries
- Source Code – text files with Python statements, the “program”
- Interpreter – executes Python instructions, two modes:
  - Run commands individually (**interactive**) or in batches (**script**)
  - Interactive mode: Console or “REPL” (read-eval-print-loop)
    - e.g. System Python install, Jupyter / Colab Notebooks
  - Script mode: processes source code (.py files)
    - e.g. CodeGrader interface

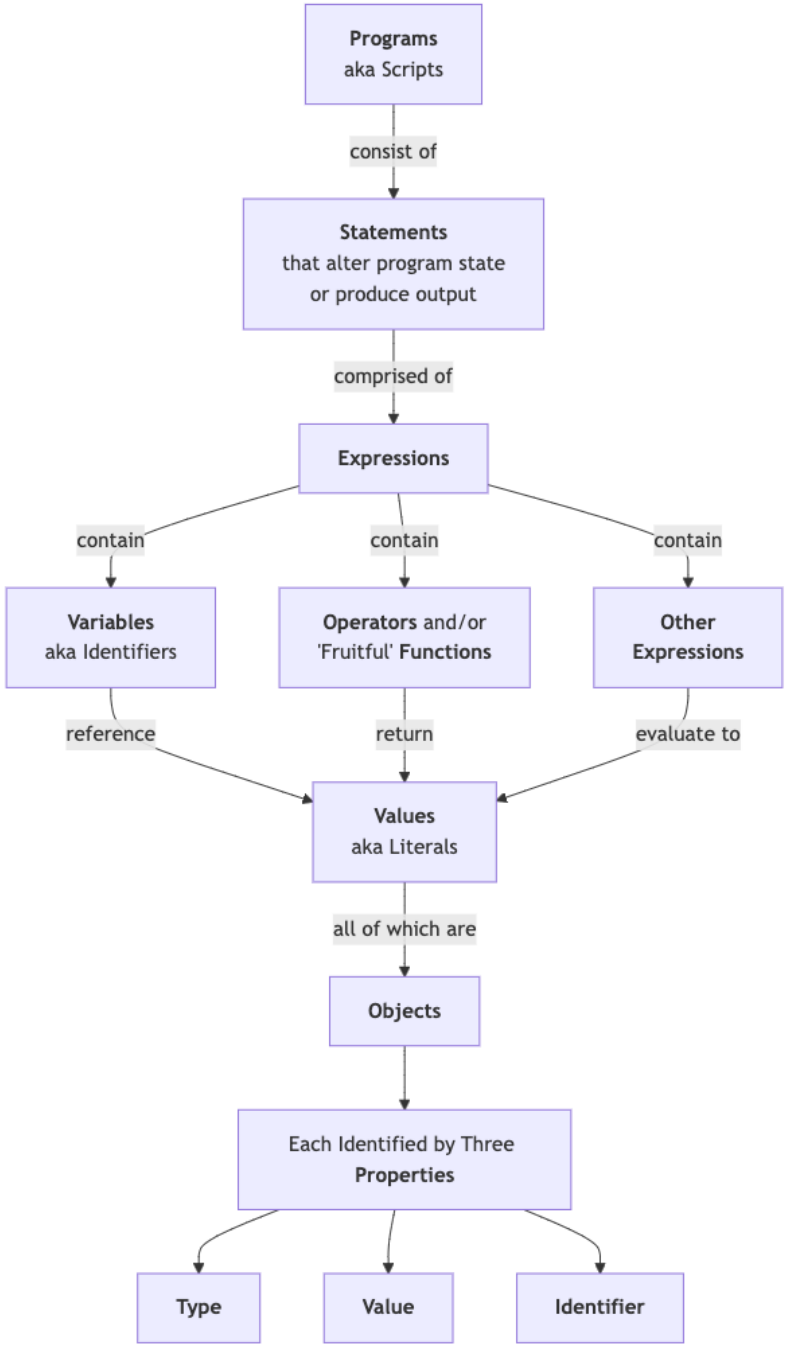
# Programming using Python

- ***Expressions*** are code that return a value when evaluated
  - expressions create and process **objects**
  - combination of variables, operators, and values (literals)
  - e.g.: “Hello world.”, miles * 5280
- A ***statement*** is a program instruction, containing expressions
  - unit of code with effect, like creating a variable, or displaying a value
  - each statement usually appears on its own line
  - e.g.: print(“Hello world.”)
- ***Programs*** mostly consist of a series of statements
- ***Expressions*** return values, ***statements*** do not
  - Interactive: values returned by expressions are displayed (REPL)
  - Script: values returned by “stray” expressions are lost

# Objects

- Python is an Object-Oriented Programming Language
  - everything is an **object**: integers, strings, functions, lists, etc.
- Every object has three defining properties
  - ***Value***: The associated value, such as "20", "abcdef", or 55
  - ***Type***: The type of the object, such as integer or string
  - ***Identity***: A unique identifier that describes the object
- Type determines object's supported behavior
  - integers can be added, multiplied
  - strings can be added or multiplied, but not divided

# Python Object Hierarchy



# Programming (general)

- A computer ***program*** consists of instructions executing one at a time. Basic instruction types are:
  - ***Input***: A program gets data, perhaps from a file, keyboard, touchscreen, network, etc.
  - ***Process***: A program performs computations on that data, such as adding two values like x + y.
  - ***Output***: A program puts that data somewhere, such as to a file, screen, or network.
- Programs use ***variables*** to refer to data… The name is due to a variable's value "varying" as a program assigns a variable … with new values.

# Computational Thinking

- Understanding a problem and breaking it down into a sequence of steps to solve it – an **algorithm**
- Algorithm: a step-by-step procedure for solving a problem or accomplishing some end especially by a computer. *http://www.merriamwebster.com/dictionary/algorithm*
- How?

# “Computational Thinking” Approach

Problem to Program

- Understand the **problem**
  - What do I have to accomplish? → Identify requirements
- Develop a **strategy**
  - How will I approach the problem? → Break it down (decompose)
- Design required **algorithms**
  - How will I solve the problem(s)? → Procedures
- Document the solution
  - How do I communicate the plan? → **Pseudocode**
- Write the **code**
  - How do I implement the pseudocode? → Python!

# Writing Algorithms - Pseudocode

- Pseudocode is a kind of “structured English” for describing algorithms. It allows the designer to focus on the logic of the algorithm without being distracted by details of language syntax (e.g., Python, C++, or Java).
- Alternatively, pseudocode is a narrative for someone who knows the requirement (problem domain) and is trying to learn how the solution is organized.
- Before writing the code for a problem in a certain programming language, we describe the entire logic of the algorithm in pseudocode so that implementation becomes a rote mechanical task of translating line by line into source code.

# Trivial Example: MAX value…

## Write a program to identify the largest number in a list

Example: 7, 12, 1, -1, 5, 10

- Problem?
- Strategy?
- Algorithm?
- Pseudo code
- Python

Intuition (look at the list and see it) → Pseudo code

# Trivial Example: MAX value…

## Write a program to identify the largest number in a list

- Problem?
  - Find the max value
- Strategy?
  - Look through the list and select the highest number
- Algorithm?
  - assume the first value in the list is the largest
  - look at each value in the list
    - if that value is larger than the current max, update max
  - print the result

# Trivial Example: MAX value…

## Write a program to identify the largest number in a list

- Pseudocode:
  - SET Max to value of first item in List
  - FOR every other Item in List
    - IF Item > Max
      - Max = Item
  - PRINT Max
- Note:
  - Semi-formal language
  - Indenting shows structure

# Four Pillars of Programming

1. Assignment – assign or change the value of variables
2. Sequence – execute instructions one at a time, in order
3. Selection – decision points that alter the course of action
4. Repetition – repeating sections of code until a condition is met

With all four, a language is “Turing Complete” – can solve any problem that is computationally feasible. Other features just make it easier.

# The Four Pillars of Programming

1. Assignment
2. Sequence
3. Selection
4. Repetition

***Spot them in this code?***

```
### find max value in list

# create an arbitrary list
my_list = [7, 12, 1, -1, 5, 10]

# pick the first value as a candidate for maximum
max_val = my_list[0]

# loop through the list looking for larger values,
# update the maximum as required
for val in my_list:
    if val > max_val:
        max_val = val

# print the result
print("The maximum value is:", max_val)
```

```
The maximum value is: 12
```

# 2nd Example: Sample Mean…

## Compute the sample mean for a list of numbers

- Problem?
- Strategy?
- Algorithm?
- Pseudo code
- Python

Let $X = [x_1, x_2, x_3, \ldots, x_n]$
be a sample of size $n$ from a population.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

# 2nd Example: Sample Mean…

**Compute the sample mean for a list of numbe**

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- Problem?
  - Find the average value
- Strategy?
  - Divide the total of all values by the number of values
- Algorithm?
  - Loop through the list to count number of items and sum values
  - Calculate average by dividing the list total sum by number of items
  - Print the result

# 2nd Example: Sample Mean…

**Compute the sample mean for a list of numbe**

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- Pseudocode:
  - SET Count = 0, Total = 0
  - FOR each Item in the List
    - INCREMENT Count
    - INCREMENT Total by Item
  - Avg = Total / Count
  - Print Avg
- Note:
  - “INCREMENT Count” and “Count = Count + 1” are equivalent

# Basics of Programming

## input, process, output

- Output – print
- Process – assignment, operations
- Input?
  - The `input()` function, of course!
  - Gets and returns user input **as a string**
  - Assign to a variable: `user_name = input()`
  - Display a message: `result = input("who will win on Saturday?")`
  - Convert to number: `value = int(input("points for Auburn: "))`
  - Reminders: `help(input)`

# Exercise – Average Production

- Write a Python program that:
  1. Asks for the name of a product being manufactured
  2. Asks for the units produced during each of three shifts (one at a time)
  3. Calculates and displays the average hourly production rate, assuming 8-hour shifts.
  4. Format your input and output exactly as shown below.

```
What is the product name? widget
How many units were produced in the first shift? 100
How many units were produced in the second shift? 120
How many units were produced in the third shift? 170

The average number of widget units produced is: 130.0
```