**COURSE NAME:** Principals of Programming

**LECTURE TOPIC / MODULE:** Introduction to Programming Concepts / Module 1

**DATE(S):** 07.14.25 – 07.20.25

**INSTRUCTOR:** Steven Evans

**KEY CONCEPTS:**

* Examine the basics of programming.
* Determine the methods of computational thinking.
* Analyze programming statements that result in basic input and output.
* Examine algorithm design, computers, and programs.

**DEFINITIONS:**

* A **switch** controls whether or not electricity flows through a wire.
* 0s and 1s are known as ***bits*** (*b*inary dig*its*).
  + “1” positive voltage
  + “0” Zero voltage
* They built connections of *switches*, known as **circuits**, to perform calculations such as multiplying two numbers.
* **Processors** were created to process (aka *execute*) a list of desired calculations
* Each calculation is called an **instruction**.
* Instructions are stored in a memory. A **memory** is a circuit that can store 0s and 1s.
* The programmer-created sequence of instructions is called a **program**, **application**, or just **app**.
  + 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.
* Instructions represented as 0s and 1s are known as **machine instructions**.
* A sequence of machine instructions together form an **executable program** (sometimes just called an *executable*).
* An **assembler** is a program that translates assembly language instructions into machine instructions.
* **High-level languages** support programming using formulas or algorithms.
* **Compilers**, which are programs that automatically translate high-level language programs into executable programs.
* Programs use **variables** to refer to data
* **Computational thinking**, or creating a sequence of instructions to solve a problem.
* A sequence of instructions that solves a problem is called an **algorithm**.
* **Scripting languages** used to execute programs without the need for compilation.
* A **script** is a program whose instructions are executed by another program called an **interpreter**.
  + Advantages include avoiding the compilation step during programming, and being able to run the same script on different processors as long as each processor has an interpreter installed.
* **Python** is a scripting language
* **Open-source** language, meaning the community of users participate in defining the language and creating new interpreters, and is supported by a large community of programmers.
* **Python interpreter** is a computer program that executes code written in the Python programming language.
* An **interactive interpreter** is a program that allows the user to execute one line of code at a time.
* **Code** is a common word for the textual representation of a program.
* A **line** is a row of text.
* A **prompt** (">>>") indicates the interpreter is ready to accept code.
* A **statement** is a program instruction.
  + A program mostly consists of a series of statements, and each statement usually appears on its own line.
* **Expressions** are code that return a value when evaluated.
* **Variables** are named references to values stored by the interpreter.
* A new variable is created by performing an **assignment** using the = symbol, such as salary = wage \* hours \* weeks, which creates a new variable called salary.
* The **print()** function displays variables or expression values.
* '#' characters denote **comments**, which are optional but can be used to explain portions of code to a human reader.
* A new output line starts after each print statement, called a **newline**.
* Specifying **end=' '**, keeps the next print's output on the same line separated by a single space.
* Any space, tab, or newline is called **whitespace**.
* A **type** determines how a value can behave.
* The **int()** function can be used to convert that string to the integer 123.
* **Logic error**, because the program is logically flawed.
  + A logic error is often called a **bug**.
* Code development is usually done with an **integrated development environment**, or **IDE**.
* **Whitespace** is any blank space or newline.

**FORMULAS / NOTES:**

* Output can be moved to the next line by printing \n, known as a newline character.
* A programmer can combine input() and int() to read in a string from the user and then convert that string to an integer for use in a calculation.
* Adding a string inside the parentheses of input() displays a prompt to the user before waiting for input and is a useful shortcut to adding an additional print statement line.

**EXAMPLE(S) / ILLUSTRATION(S):**

**A screenshot of a computer instructions

AI-generated content may be incorrect.**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**A blue box with black text

AI-generated content may be incorrect.**

**A white background with black text

AI-generated content may be incorrect.**

**A white background with black text

AI-generated content may be incorrect.**

**A close-up of a computer screen

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**A blue rectangular object with text

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**QUESTIONS TO REVISIT / COMMON PITFALL(S):**

* A common error is to forget the comma between items, as in print('Name' user\_name).

**ADDITIONAL READING(S) / LINKS:**

* IDLE is the official Python IDE that is distributed with the installation of Python from http://www.python.org. IDLE provides a basic environment for editing and running programs.
* PyDev (http://pydev.org) is a plugin for the popular Eclipse program. PyDev includes extra features such as code completion, spell checking, and a debugger that can be useful tools while programming.
* For learning purposes, web-based tools like CodePad (http://www.codepad.org) or Repl (http://www.repl.it) are useful.

**COURSE NAME:** Principals of Programming

**LECTURE TOPIC / MODULE:** Introduction to Programming Concepts / Module 2

**DATE(S):** 07.21.25 – 07.27.25

**INSTRUCTOR:** Steven Evans

**KEY CONCEPTS:**

* Evaluate the differences between Python objects and variables.
* Develop Python programs using variables and expressions.
* Examine a variable in programming.
* Investigate how to find and import modules.

**DEFINITIONS:**

* Increasing a variable's value by 1, as in x = x + 1 is known as **incrementing** the variable.
* An **identifier**, also called a **name**, is a sequence of letters (a-z, A-Z) **underscores** (\_), and digits (0–9), and must start with a letter or an underscore.
* **Case sensitive**, meaning upper- and lowercase letters differ.
* **Reserved words**, or **keywords**, are words that are part of the language.
* **PEP 8** (PEP is an acronym for Python Enhancement Proposal) is a document that outlines the basics of how to write Python code neatly and consistently.
* An **object** represents a value and is automatically created by the interpreter when executing a line of code.
* Deleting unused objects is an automatic process called **garbage collection** that helps to keep the memory of the computer less utilized.
* **Name binding** is the process of associating names with interpreter objects.
* Each Python object has three defining properties: value, type, and identity.
  + **Value**: A value such as "20", "abcdef", or 55.
  + **Type**: The type of the object, such as integer or string.
    - **type()** prints the type of an object.
  + **Identity**: A unique identifier that describes the object.
    - **id()** gives the value of an object's identity.
* **Mutability** indicates whether the object's value is allowed to be changed.
* Integers and strings are **immutable**; modifying their values with assignment statements results in new objects being created and the names bound to the new object.
* A **floating-point** number is a real number, like 98.6, 0.0001, or -666.667.
* A **floating-point literal** is written with the fractional part even if that fraction is 0, as in 1.0, 0.0, or 99.0.
* Thus, **float** is a data type for floating-point numbers.
* **Scientific notation** is written using an e preceding the power-of-10 exponent, as in 6.02e23 to represent 6.02x1023.
* **OverflowError** happens when assigning a floating-point value outside of the following range:
  + The maximum floating-point value is approximately 1.8x10308
  + Minimum floating-point value is 2.3x10-308.
  + **Overflow** occurs when a value is too large to be stored in the memory allocated by the interpreter.
* A **literal** is a specific value in code like 2.
* An ***operator*** is a symbol that performs a built-in calculation, like +, which performs addition.
* An **expression** evaluates to a value, which replaces the expression.
  + Ex: If x is 5, then x + 1 evaluates to 6, and y = x + 1 assigns y with 6.
* An expression is evaluated using the order of standard mathematics, and such order is known in programming as **precedence rules**.
* Minus (-) used as negative is known as **unary minus**.
* **Compound operators** provide a shorthand way to update a variable, such as age ***+=*** 1 being shorthand for age = age + 1. Other compound operators include ***-=***, ***\*=***, ***/=***, and ***%=***.
* The **modulo operator** (**%**) evaluates the remainder of the division of two integer operands.
* A **module** is a file containing Python code that can be used by other modules or scripts.
* **Script** is a file containing Python code that is passed as input to the interpreter.
* **Dot notation** is used to reference an object in an imported module.
* **Import** executes the contents of a file containing Python code and makes the definitions from that file available.
* A **function** is a list of statements that can be executed simply by referring to the function's name.
* The process of invoking a function is referred to as a **function call**.
* The item passed to a function is referred to as an **argument**.
* **Unicode** to represent every possible character as a unique number, known as a **code** **point**.
* The two-item sequence is called an **escape** **sequence**.
* A **raw string** is created by adding an 'r' before a string literal, as in r'this is a raw string\'', which would output as this is a raw string\'.
* The built-in function ***ord()*** returns an encoded integer value for a string of length one.
* The built-in function ***chr()*** returns a string of one character for an encoded integer.

**FORMULAS / NOTES:**

* An assignment statement's left side must be a variable. The right side can be an expression.
* In programming, = is an assignment of a left-side variable with a right-side value. = is NOT equality as in mathematics. Thus, x = 5 is read as "x is assigned with 5," and not as "x equals 5."
* Names that start and end with double underscores are allowed but should be avoided because Python has special usages for double underscore names.
* Words like "and" or "True" cannot be used as names.
* A good practice when naming variables is to use all lowercase letters and to place underscores between words.
* Floating-point types *should be used* to represent quantities that are measured, such as distances, temperatures, volumes, and weights, whereas
* Integer types *should be used* to represent quantities that are counted, such as numbers of cars, students, cities, hours, and minutes.
* Commas are not allowed in an integer literal. So 1,333,555 is written as 1333555.
* The floored division operator **//** can be used to round down the result of a floating-point division to the closest whole number value.

**EXAMPLE(S) / ILLUSTRATION(S):**

**A screenshot of a test

AI-generated content may be incorrect.**

**A screenshot of a test

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a math application

AI-generated content may be incorrect.**

**A screenshot of a white sheet with black text

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer error

AI-generated content may be incorrect.**

**A close-up of a computer screen

AI-generated content may be incorrect.**

**A close up of text

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**QUESTIONS TO REVISIT / COMMON PITFALL(S):**

* N *A common error is to omit parentheses and assume an incorrect order of evaluation, leading to a bug. Ex: If x is 3, then 5 \* x+1 might appear to evaluate as 5 \* (3+1) or 20, but actually evaluates as (5 \* 3) + 1 or 16 (spacing doesn't matter).*

**ADDITIONAL READING(S) / LINKS:**

Python Software Foundation. <https://docs.python.org/3/reference/expressions.html>

GeeksforGeeks. <https://www.geeksforgeeks.org/short-circuiting-techniques-python/>

[http://docs.python.org/3/reference/lexical\_analysis.html](http://docs.python.org/3/reference/lexical_analysis.html#keywords)

**COURSE NAME:** Principals of Programming

**LECTURE TOPIC / MODULE:** Introduction to Programming Concepts / Module 3

**DATE(S):** 07.28.25 – 08.03.25

**INSTRUCTOR:** Steven Evans

**KEY CONCEPTS:**

* Design Python programs using numeric data types and numeric operators.
* Evaluate the purpose of an array data structure.
* Identify a method that utilizes an array data structure.

**DEFINITIONS:**

* **Numeric types** int and float represent the most common types used to store data.
* **Sequence types** string, list, and tuple are all containers for collections of objects ordered by position in the sequence, where the first object has an index of 0 and subsequent elements have indices 1, 2, etc.
* The only **mapping type** in Python is the dict type. Like a sequence type, a dict serves as a container. However, each element of a **dict** is independent, having no special ordering or relation to other elements.
  + A dictionary uses key-value pairs to associate a key with a value.
* A **type conversion** is a conversion of one type to another, such as an int to a float.
* An **implicit conversion** is a type conversion automatically made by the interpreter, usually between numeric types.
* A processor stores a number using base 2, known as a **binary number**.
* A **decimal number**, each digit must be 0-9 and each digit's place is weighed by increasing powers of 10.
* The string **format()** function allows a programmer to create a string with placeholders that are replaced by values or variable values at execution.
* A placeholder surrounded by curly braces { } is called a **replacement field**.
  + Values inside the format() parentheses are inserted into the replacement fields in the string.
* A **keyword argument**  defines a name and value in the format() parentheses.
* **Format specification** inside of a replacement field allows a value's formatting in the string to be customized.
  + Ex: Using a format specification, a variable with the integer value 4 can be output as a floating-point number (4.0) or with leading zeros (004).
* A **presentation type** for the value, such as integer (4), floating point (4.0), fixed precision decimal (4.000), percentage (4%), binary (100), etc.
  + A presentation type can be set in a replacement field by inserting a colon : and providing one of the presentation type characters
* An **array** is a special variable having one name, but storing a list of data items, with each item being directly accessible.
  + Some languages use a construct similar to an array called a **vector**.
  + Each item in an array is known as an **element**.
* In an array, each element's location number is called the **index**.
* An **array variable** is an ordered list of items of a given data type and size.
* **Swapping** two variables x and y means to assign y's value with x, and x's value with y. If x is 33 and y is 55, then after swapping x is 55 and y is 33.

**FORMULAS / NOTES:**

* **Numbers:** Python supports Booleans, integers, floating point numbers, and complex numbers. They are defined as bool, int, float, and complex class in Python. Integers and floating points are separated by the presence or absence of a decimal point: 5 is an integer whereas 5.0 is a floating point number.
* **Booleans:** A Boolean operation returns a value of either True or False. Boolean operations are ordered in priority, so if more than one Boolean operation occurs in an expression, the operation with the highest priority will occur first.

A screenshot of a computer

AI-generated content may be incorrect.

**Important Note:** Python is case-sensitive. True and False are different than true and false. Be sure to capitalize True and False when writing Python code.

* Good practice is to use named replacement when formatting strings with many replacement fields to make the code more readable.
* The positional and inferred positional replacement types cannot be combined. Ex: '{} + {1} is {2}'.format(2, 2, 4) is not allowed. However, named and either positional replacement type can be combined. Ex: '{} + {} is {sum}'.format(2, 2, sum = 4)
* **Integers: A screenshot of a computer

  AI-generated content may be incorrect.**
* **Floating Point: A screenshot of a computer

  AI-generated content may be incorrect.**

Additional functions are in the math module

* Import Math
  + a = math.sqrt(x)
  + b = math.sin(x)
  + c = math.cos(x)
  + d = math.tan(x)
  + e = math.log(x)
* The colon : in the replacement field separates the "what" on the left from the "how" on the right.
* A white background with black text

  AI-generated content may be incorrect.

**EXAMPLE(S) / ILLUSTRATION(S):**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a math problem

AI-generated content may be incorrect.**

**A screenshot of a math test

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A green board with white text and numbers

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A white rectangular object with a red border

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A white screen with red text

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**QUESTIONS TO REVISIT / COMMON PITFALL(S):**

* Need to revisit the concept of Arrays and Loops for full understanding.

**ADDITIONAL READING(S) / LINKS:**

**COURSE NAME:** Principals of Programming

**LECTURE TOPIC / MODULE:** Introduction to Programming Concepts / Module 4

**DATE(S):** 08.04.25 – 08.10.25

**INSTRUCTOR:** Steven Evans

**KEY CONCEPTS:**

**DEFINITIONS:**

**FORMULAS / NOTES:**

**EXAMPLE(S) / ILLUSTRATION(S):**

**QUESTIONS TO REVISIT / COMMON PITFALL(S):**

**ADDITIONAL READING(S) / LINKS:**