Module 1

Introducing Python

CENGAGE

Lesson 1.1: Working with the Python Interactive Shell

Python Interactive Shell:

- Open at command line using command: `python`
- Interfaces with Python interpreter.
- Allows user to run Python commands at command line.
    - Sample command: `print("Message")`
- Anything typed into shell is echoed back.
- Can even do calculations.

Example

- Simple Print Statement
  - print(“Happy Birthday”)
- Print Calculations
  - print(5 * 4)
  - print(24 + 6 * 3)
- Print whole numbers using commas to separate
  - print(0,1,2,3,4)
- Repeat Strings
  - print(“- - - - -”)
  - print(“-” * 5)
  - print(“hello   “ * 5)

Lesson 1.2: Writing and Running Simple Scripts

- Can also run code saved in files.
  - The files are called modules.
- A script is a module that can be run.
- File extension used is: `.py`
- Execute script using command: `python sample1.py`
- Sample script, `sample1.py`, is shown in Snippet 1.6.
  - Run using command: `python sample1.py`

Lesson 1.2.1 Running a File Containing Invalid Commands

- Invalid commands in script will cause errors.
- Output will be stack trace (also called traceback).
  - Gives error info such as where, what kind, and what other calls were triggered.
  - Read from bottom to top.
  - **Example:**
    - **print(invalid instruction)**

Passing User Arguments to Scripts

- To pass arguments to script, script must include line: `import sys`
- The sys function allows you to read command-line arguments that have been passed to the interpreter
- The `sys.argv` list contains the argument(s).
  - Can pass any number of arguments, separating each with comma.
  - First argument will have index 1, `sys.argv[1]`, second argument index 2, `sys.argv[2]`, and so on.

Example

- Simple Print Statement

```
import sys

print(“First Name: ", sys.argv[1])
print(“Middle Name: ", sys.argv[2])
print(“Last Name: “, sys.argv[3])
```

- Type within interpreter to call the module

```
>>> python sample1.py Eric James Clayborn
```

Your Turn!!! Code the Following

- Use two print statements to print the following
- Course Title:
- Course Number:

- Use the print statements to print 10 dashes (-) as borders above and below the course title and course number
- Run the script with python sample1.py and pass two string arguments

- The course title is: ISYS
- The course number is: 229

Lesson 1.3.1: Variables

- Variables can reference values of different data types.
  - They refer to values in memory
- They don't need to be declared before use.
  - X = "No need to declare this string before use"
- Value and type can change during runtime.
- Use `type` function to check type.
  - Example: `type(X)`

Lesson 1.3.1: Variables

Naming Identifiers and Reserved Words

- Some rules for variables and other identifiers:
  - Can consist of upper and lowercase letters, underscores, unicode identifiers, and digits 0 to 9.
  - Cannot begin with digit.
  - No other characters can be in identifiers.
    - Module names with spaces should be avoided.
  - Python reserved words or keywords can’t be used.
- Identifier names are case sensitive.

Lesson 1.3.1: Variables

Python Naming Conventions: These are guidelines only

- Compound variable names should be written in snake case notation.
  - Example: `snake_case`
- Avoid using the following
  - Example: `PascalCase`
  - Example: `camelCase`
- Constant names should be in all caps.
- Avoid lowercase L or uppercase O as single character names.

Lesson 1.3.1: Data Type Variables

Numeric Values—Integers

- Integers are whole numbers that are either positive or negative.
- Arithmetic operations can be performed on them.

String Values

- Strings are sequence of characters between two quotation marks.
  - Can use either double or single quotes.
- Can contain letter, numbers, or symbols.

Lesson 1.3.1: Variables

Float Values—Decimal

- Numbers with decimal palaces.
- Arithmetic operations can be performed on them.

Boolean Values

- Return a True of False value

Lesson 1.3.1: Variables

Type Conversion

- Can convert integer to string.
  - Example: `str(7)`
- Can convert string to integer.
  - Example: `int("100")`
  - Will get error if trying to convert string that doesn’t contain integer.

Lesson 1.3.1: Variables

Assigning Variables

- To assign value to variable use equal sign.
  - Example: `number_seven = 7`
- Error is raised if trying to use variable before it is assigned value.
- Can use `+` to concatenate two strings
- Commas , can be used to connect strings and integers
- Variables are not deeply linked.

Multiple Assignment

- Can assign multiple variables in one statement.
  - **Example:** `a, b, c = 1, 2, 3`

Example

- Simple Print Statement to concatenate string values

```
Print("I Love Python" + "!" * 3)
```

OR

```
message = "I Love Python"
print(message , "!" * 3)
```

Lesson 1.3.1: Variables

Multiple Assignment

- Can assign multiple variables in one statement.
  - **Example:** `a, b, c = 1, 2, 3`

Example

- Simple Print Statements

```
a, b, c = 1, 2, 3
print(a)
print(b)
print(c)

    OR

print(a, b, c)
```

Example

- Simple Print Statements

```
first_name, middle_initial, last_name = “Eric”, “James”, “Clayborn”

print(“Your first name is: “ + first_name)

print(“Your Middle Initial is: “ + middle_initial)

print(“Your Last Name is: “ + last_name)
```

Lesson 1.4: User Input, Comments, and Indentations

User Input from the Keyboard

- Use `input()` function to get user input from keyboard.
  - Example: `message = input()`
  - Program execution halts until user inputs value and presses Enter key.

Lesson 1.4: User Input, Comments, and Indentations

Passing in a Prompt to the Input Function

- Syntax: `input("Insert prompt here")`

Using Different Input Data Types in Your Program

- The `input()` function always returns string.
- Use built-in `int()` function to convert to integer.

Lesson 1.4: Example

User Input from the Keyboard

- Use `input()` function to get user input from keyboard.

Example:

```
age = input("Please enter your age: ")
print("Before" , type(age))

age = int(age)
print("After" , type(age))
```

Lesson 1.4: User Input, Comments, and Indentations

Comments

- Block
  - Start with # sign.
  - Comes on line before statement it annotates.
  - Placed at same indent level as statement.
- Inline
  - Start with # sign.
  - Placed on same line as statement it annotates.
- Documentation String (docstrings)
  - String wrapped in triple (double or single) quotation marks.
  - Module docstrings should be at beginning of file.
  - Can be used for multiple line comments.

Your Turn!!! Code the Following

- Import the sys function
- Use the input statement to gather the year a user was born
- Use print statements to print the first name, middle initial, and last name using arguments gathered from the python interpreter
- Use the print statement to display the difference between 2023 and the year a user was born.
  - Display a print statement stating “You are” X “years old”.