Lesson 1: Introduction to Python Basics and Algorithmic Thinking

Objective:

Understand basic Python programming concepts: variables, data types, and basic algorithms. Apply algorithmic thinking to solve simple problems related to space science.

Lesson Plan

1. Introduction (10 minutes)

Engagement:

- Discuss how Python is used in space science (e.g., NASA uses Python for data analysis).
- Show a simple example of Python code to pique interest.
    - Use the python (turtle) sandbox in CodeHS to draw a smiley face:

```
# Set up the turtle
speed(5)
bgcolor("white")
pensize(2)
penup()

# Draw the face
goto(0, -100)
pendown()
color("yellow")
begin_fill()
circle(100)
end_fill()
penup()

# Draw the left eye
goto(-35, 35)
pendown()
color("black")
begin_fill()
circle(10)
end_fill()
penup()

# Draw the right eye
goto(35, 35)
pendown()
color("black")
begin_fill()
circle(10)
end_fill()
penup()
```

```
# Draw the mouth
goto(-40, -20)
pendown()
setheading(-60)
circle(40, 120)
penup()

hideturtle()
```

2. Explanation (15 minutes)

Comments, Variables and Data Types:

Importance of Comments in Code:

- Introduce the ‘#’ symbol to make comments and explain what is happening on specific lines of your code. This practice is particularly helpful when you share your code with others, as it helps them understand your thought process and the functionality of your code.
- It's crucial to emphasize the importance of commenting early in the learning process. Although the example code below may seem excessive in terms of comments, it serves as excellent practice for students. As an assessment, have students go back and add comments to their code. This not only helps reinforce their understanding but also ensures they can communicate their knowledge effectively.

Example code:

```
# Set up the turtle
speed(1)             # Sets the turtle's drawing speed to a slow pace
bgcolor("white")     # Sets the background color of the canvas to white
pensize(2)           # Sets the thickness of the pen to 2 units
penup()              # Lifts the pen so that it doesn't draw while moving

# Draw the face
goto(0, -100)        # Moves the turtle to the position (0, -100) without drawing
pendown()            # Lowers the pen so that it draws while moving
color("yellow")      # Sets the pen color to yellow
begin_fill()         # Starts filling the shape with the current pen color (yellow)
circle(100)          # Draws a circle with a radius of 100 units
end_fill()           # Fills the circle with yellow color and stops filling
penup()              # Lifts the pen to move without drawing

# Draw the left eye
goto(-35, 35)        # Moves the turtle to the position (-35, 35) without drawing
pendown()            # Lowers the pen to draw
color("black")       # Sets the pen color to black
begin_fill()         # Starts filling the shape with the current pen color (black)
circle(10)           # Draws a circle with a radius of 10 units for the left eye
end_fill()           # Fills the circle with black color and stops filling
penup()              # Lifts the pen to move without drawing

# Draw the right eye
goto(35, 35)         # Moves the turtle to the position (35, 35) without drawing
pendown()            # Lowers the pen to draw
color("black")       # Sets the pen color to black
begin_fill()         # Starts filling the shape with the current pen color (black)
circle(10)           # Draws a circle with a radius of 10 units for the right eye
end_fill()           # Fills the circle with black color and stops filling
penup()              # Lifts the pen to move without drawing

# Draw the mouth
goto(-40, -20)       # Moves the turtle to the position (-40, -20) without drawing
pendown()            # Lowers the pen to draw
setheading(-60)      # Sets the turtle's heading to -60 degrees
circle(40, 120)      # Draws an arc with a radius of 40 units and an extent of 120
degrees
penup()              # Lifts the pen to move without drawing

hideturtle()         # Hides the turtle cursor
```

Variables as Storage Containers for Data:

- Variables are used to hold information that you will use repeatedly in your program.
- You must define your variables by giving them a name.

  Important Tips:
  - Choose descriptive names for your variables to make your code more readable.
  - Avoid using Python keywords or commands as variable names to prevent errors in your code.

Data types: integers, floats, and strings

Integers:

- Represent whole numbers without a decimal point.
- Examples: 1, 42, 1000.

Floats:

- Represent numbers that include a decimal point.
- Examples: 3.14, 0.001, 100.0.

Strings:

- Represent sequences of characters (words, phrases, or numbers in quotes).
- Examples: "hello", "12345", "3.14".

Example Code:

```
# Variable is "planet_name" and the data type is a string
planet_name = "Mars"
# Variable is "distance_from_sun_km" and the data type is a float
distance_from_sun_km = 227.9
#We can use print statements to write a sentence using a combination of strings (in
quotes) and variables (not in quotes)
print("The distance from the Sun to", planet_name, "is", distance_from_sun_km,
"million kilometers.")
```

Understanding Algorithms and Their Role in Problem-Solving

What is an Algorithm?

- An algorithm is a set of steps designed to solve a problem or accomplish a task.
- The concept of algorithms is fundamental to computer science.

Everyday Examples of Algorithms:

- Brushing your teeth
- Baking cookies
- Making a sandwich
- These daily activities involve following specific steps, just like algorithms in programming.

Algorithms in Computer Science:

- In computer science, algorithms are used to break down complex problems into smaller, manageable steps to find solutions efficiently.

Example: Calculating the Time Light Takes to Travel from the Sun to a Planet

1. **Understanding the Problem:**
   - You need to determine how long it takes light to travel from the Sun to a specific planet.

2. **Breaking Down the Problem:**
   - What do you know?
     - The equation

$$\text{Speed} = \frac{\text{Distance}}{\text{Time}}$$

     - The distance between the Sun and the planet
     - The speed of light
3. **Rearranging the Equation:**
   - To find the time, rearrange the equation:

$$\text{Time} = \frac{\text{Distance}}{\text{Speed}}$$

4. **Implementing in Code:**
   - Use variables to store the distance and speed of light.
   - Write the steps in Python to calculate and print the time.

Breaking it Down Step by Step:

- Define the variables for distance and speed.
- Use the equation to calculate the time.
- Print the result.

By breaking down the problem into smaller steps, you can effectively solve it using an algorithm. This approach helps manage complex tasks and ensures accuracy in problem-solving.

Example Code:

```
planet_name = "Jupiter"
speed_of_light_km_s = 299792  # km per second
distance_from_sun_km = 227.9 * 10**6  # convert to kilometers
time_s = distance_from_sun_km / speed_of_light_km_s
print("Light takes", time_s, "seconds to travel from the Sun to", planet_name)
```

Vocabulary:

Algorithm: A sequence of steps or instructions designed to perform a specific task or solve a problem.
Variable: A storage location identified by a name used to hold data that can be changed during program execution.
Data Type: A classification that specifies the type of data a variable can hold, such as integers, floats, or strings.
Integer: A whole number without a decimal point.
Float: A number that includes a decimal point.
String: A sequence of characters enclosed in quotes, used to represent text in a program.

3. Hands-On Activity (20 minutes)

Task:

- Students will write Python scripts to calculate and print the distances of various planets from the Sun and the time light takes to travel that distance.

Worksheet (below):

- Practice with defining variables and printing their values.
- Practice with simple arithmetic operations and algorithms.

Example Worksheet Tasks:

Example Code:

```
# Define variables for the planet Jupiter
planet_name = "Jupiter"
distance_from_sun_km = 778.5  # in millions of kilometers
print("The distance from the Sun to", planet_name, "is", distance_from_sun_km,
"million kilometers.")

# Calculate the time light takes to travel from the Sun to Jupiter
speed_of_light_km_s = 299792  # km per second
distance_from_sun_km = distance_from_sun_km * 10**6  # convert to kilometers
time_s = distance_from_sun_km / speed_of_light_km_s
print("Light takes", time_s, "seconds to travel from the Sun to", planet_name)
```

4. Review (10 minutes)

Q&A:

- Address any questions students might have about variables, data types, or the example code.

Exit Ticket:

1. What is a variable in Python?
   A. A storage location identified by a name used to hold data.
   B. A type of loop in Python.
   C. A function that takes user input.
   D. A mathematical operation in Python.
   Answer: A

2. What are the three basic data types in Python we discussed today?
   A. Lists, dictionaries, and tuples
   B. Integers, floats, and strings
   C. Loops, functions, and classes
   D. Variables, conditions, and loops
   Answer: B

3. Which of the following is the correct algorithm in Python to calculate the time it takes for light to travel from the Sun to Earth?

```
A. speed_of_light_km_s = 299792
   distance_from_sun_km = 149.6 * 10**6
   time_s = distance_from_sun_km / speed_of_light_km_s
   print("Light takes", time_s, "seconds to travel from the Sun to Earth")

B. speed_of_light_km_s = 299792
   distance_from_sun_km = 149.6
   time_s = distance_from_sun_km + speed_of_light_km_s
   print("Light takes", time_s, "seconds to travel from the Sun to Earth")

C. speed_of_light_km_s = 299792
   distance_from_sun_km = 149.6 * 10**3
   time_s = distance_from_sun_km / speed_of_light_km_s
   print("Light takes", time_s, "seconds to travel from the Sun to Earth")

D. speed_of_light_km_s = 299792
   distance_from_sun_km = 149.6 * 10**6
   time_s = speed_of_light_km_s / distance_from_sun_km
   print("Light takes", time_s, "seconds to travel from the Sun to Earth")
```

Answer: A

Worksheet 1: Variables and Basic Algorithms

Section 1: Understanding Variables

In your CodeHS Sandbox, you will be writing your own code based on your planet. Remember the example given during class time to help guide you.

4. Define a variable to store the name of a planet.

5. Define a variable to store the distance from the Sun to a planet in kilometers.

6. Print the name and distance of the planet.

7. *BONUS* Use a complete sentence to print the name and distance from the Sun.

Section 2: Understanding Data Types

8. Define a string variable to store the type of a celestial object.

9. Define an integer variable to store the number of moons.

10. Define a float variable to store the mass of a planet in kilograms.

11. Print the type, number of moons, and mass of the planet.

12. *BONUS* Use a complete sentence to print the type, number and mass

Section 3: Basic Algorithms

Write an algorithm to calculate the time it takes for light to travel from the Sun to a planet. Use the speed of light (299,792 km/s).

ANSWER KEY:

Section 2: Understanding Data Types

Define a string variable to store the type of a celestial object.

```
object_type = "Planet"
```

Define an integer variable to store the number of moons.

```
number_of_moons = 1  # Earth has one moon
```

Define a float variable to store the mass of a planet in kilograms.

```
planet_mass_kg = 5.972e24  # mass of Earth in kilograms
```

Print the type, number of moons, and mass of the planet.

```
print("Object Type:", object_type)
print("Number of Moons:", number_of_moons)
print("Mass of Planet (kg):", planet_mass_kg)
```

Section 3: Basic Algorithms

Write an algorithm to calculate the time it takes for light to travel from the Sun to a planet. Use the speed of light (299,792 km/s).

```
speed_of_light_km_s = 299792  # km per second
distance_from_sun_km = 149.6 * 10**6  # convert to kilometers
time_s = distance_from_sun_km / speed_of_light_km_s
print("Light takes", time_s, "seconds to travel from the Sun to Earth")
```