Variables and Datatypes

Variables in Python are used to store values. A variable is created by assigning a value to a name, using the assignment operator (=). For example:

```
x = 5
y = "Hello, World!"
```

In the above example, the variable `x` is assigned the value `5`, and the variable `y` is assigned the value `"Hello, World!"`. We can display these values in the terminal using the `print()` function.

```
print(x)
print(y)
```

There are several data types in Python, including:

- Integers (`int`) - Whole numbers, such as `1`, `2`, and `3`.
- Floating-point numbers (`float`) - Numbers with decimal points, such as `2.5`, `3.14` or even `1.`.
- Strings (`str`) - A sequence of characters, such as `"Hello, World!"` or `"Hello, Python!"`. Strings can be enclosed in single (`'`) or double (`"`) quotes.
- Booleans (`bool`) - A true or false value, represented by the keywords `True` and `False`.
- Lists (`list`) - An ordered collection of values, enclosed in square brackets and separated by commas. For example: `[1, 2, 3]`.
- Tuples (`tuple`) - Similar to lists, but enclosed in parentheses and cannot be modified once created. For example: `(1, 2, 3)`.
- Dictionaries (`dict`) - An unordered collection of key-value pairs. For example: `{"name": "John", "age": 30}`.

Take the below example, what datatypes are variables `x` and `y`? What output do we predict to see displayed in the terminal?

```
x = 5
y = "5"

print(x)
print(y)
```

How can we check the data type of a variable? We use the `type()` function. For example:

```
print(type(x))
print(type(y))
```

It's also possible to change the data type of a variable by using the appropriate casting function, like `int()`, `float()`, `str()`, etc.

```
x = 5
print(x)
print(type(x))

x = float(x)
print(x)
print(type(x))

x = str(x)
print(x)
print(type(x))
```

It's important to note that variables can be reassigned new values at any time.

```
x = 5
print(x)
x = 10
print(x)
x = 123
print(x)
```

You can also perform operations on variables of certain data types, like addition, multiplication or division.

```
x = 5
y = 2
print(x + y)
print(x * y)
print(x / y)
```

Please note that different data types can have different behavior when performing operations. For now let's look at strings. Using the addition operator on two strings causes the two strings to become concatenated. The multiplications of a string with an integer `n`, causes the string to be concatenated upon itself `n` times.

```
x = "Hello"
y = "world"
print(x + y)
print(x * 3)
print((x * 3) + (y * 2))
```

Invalid Variable Names

Not all variable names are valid. A variable name must start with a letter or underscore, and can only contain letters, numbers and underscores. For example, the following are all valid variable names:

```
x = 5
my_variable = 3
myVariable = 3
_my_variable = 3
MY_VARIABLE = 3
myVariable2 = 3
```

The following are not valid variable names:

```
2my_variable = 3
my-variable = 3
my variable = 3
```

Also be cautious of using the names of Python built-ins as variable names as this can cause unexpected behavior. For example, the following should not be considered valid variable names:

```
def = 3
class = 3
if = 3
```

Notice the syntax highlighting in the above examples. The names of Python built-ins are highlighted in a different color than the names of variables. This is a useful feature to help you avoid using built-in names as variable names.

Commenting Your Code

You can add comments almost anywhere in your Python code by using the `#` symbol. Comments are ignored by the Python interpreter, and are used to explain what your code does.

For example:

```
x = 5 # x is an integer equal to 5
y = 2 # y is an integer equal to 2
# z = x + y # z is an integer equal to 7. This line is commented out.
```

Commenting your code is a good practice, as it makes your code more readable and easier to understand. It's also a good idea to comment out code that you don't want to run, instead of deleting it. This way you can easily uncomment it later if you need to.

Exercises

1. Create a variable `x` and assign it the value `10`. Print the value of `x`.
2. Create a variable `y` and assign it the value `"Hello, World!"`. Print the value of `y`.
3. Create a variable `z` and assign it the value `3.14`. Print the value of `z`.
4. Check the data type of each of the variables `x`, `y`, and `z` using the `type()` function.
5. Change the value of `x` to `20` and re-print its value.
6. Calculate and print the results of the following operations on the values of `x` and `z`: a) addition, b) multiplication and c) division.
7. Concatenate the strings in `y` and `" How are you?"` and assign the result to a new variable `greeting`. Print the value of `greeting`.
8. Convert the value of `x` to a string and assign the result to a new variable `x_str`. Print the value of `x_str` and its data type.
9. Create a list `numbers` that contains the integers `1`, `2`, and `3`. Print the list and its data type.
10. Create a tuple `coords` that contains the floating-point numbers `1.0`, `2.0`, and `3.0`. Print the tuple and its data type.
11. Create a dictionary `person` that contains the keys `"name"`, `"age"`, and `"gender"` and respective values `"Jane Doe"`, `36`, and `"Female"`. Print the dictionary and its data type.

Project: Personal Information Manager

This project will help you practice the concepts you've learned so far on variables and datatypes in Python.

Project Description:

Create a Python program that acts as a personal information manager. The program should ask the user to enter their name, age, address, email, and phone number. Then it should store this information in variables and display it back to the user.

Project Requirements:

1. The program should ask the user to enter their name, age, address, email, and phone number.
2. The program should store this information in separate variables.
3. The program should display the stored information back to the user in a formatted manner.
4. The program should handle data types appropriately, for example, age should be stored as an integer, and email should be stored as a string.

Example Output

```
Welcome to Personal Information Manager!
Please enter your name: John Doe
Please enter your age: 25
Please enter your address: 123 Main St, Anytown, USA
Please enter your email: john.doe@example.com
Please enter your phone number: 555-555-5555

Your information has been saved:
Name: John Doe
Age: 25
Address: 123 Main St, Anytown, USA
Email: john.doe@example.com
Phone Number: 555-555-5555
```

How do you accept input from the user and form the output as shown? Well, that's for you to find out through the power of research! Check the `Further Reading` section below for some resources to get you started.

Further Reading

Check out these articles on W3Schools for more information on variables and data types in Python:

- Python Variables
- Python Data Types
- Python Casting
- Python Print Statement
- Python Input

Also, check out the Python documentation for more information on variables and data types in Python.

Finally, don't be afraid to use a search engine to supplement your learning and fill any gaps in your knowledge. For example, you can search for "python variables" or "python data types" to find more information on variables and data types in Python. Websites Like StackOverflow and W3Schools are great resources for finding answers to your questions.

This is an essential part of learning Python, as not everything can be covered in a single set of lessons or tutorials. It's also critical that you consider alternative perspectives on any learning topic, as there are often multiple ways to approach a problem.