PYTHON

Lecture - 03

RECAP

Variable

- Variables are ***containers*** for storing data values.
- Python has ***no command*** for declaring a variable.
- A variable is created the moment you first assign a value to it.
- Variables do not need to be ***declared with any particular type***, and can even ***change type after*** they have been set.

```
x = 5
y = "John"
print(x)
print(y)
```

Variable Names

- A variable can have a ***short name*** (like x and y) or a more ***descriptive name*** (age, carname, total_volume).
- Rules for Python variables:
  - A variable name must start with a letter or the underscore character
  - A variable name cannot start with a number
  - A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
  - Variable names are case-sensitive (age, Age and AGE are three different variables)
  - A variable name cannot be any of the Python keywords.

Variable Examples

- **Legal Variable**: *myvar, my_var, _my_var, myVar, MYVAR, myvar2*
- **Illegal variable names:** *2myvar,my-var,my var*
- **Multi words variable**
    - Camel case: myVariableName
    - Pascal case: MyVariableName
    - Snake Case: my_variable_name

Assigning Value in Variable

- **Single value**:

  age= 20

  my_name =“Alice”

  myWeight=60.50

- **Printing output of variable**

```
age= 20
my_name = "Alice"
myWeight=60.50

print("My age is:", age)
print("My name is:", my_name)
print("My weight is:", myWeight)
```

```
My age is: 20
My name is: Alice
My weight is: 60.5
```

```
age= 20
my_name = "Alice"
myWeight=60.50

print(age)
print(my_name)
print(myWeight)
```

Assigning Multiple Values in Variable

- Python allows you to assign values to multiple variables in one line:

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

- And you can assign the *same* value to multiple variables in one line:

```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

Exercises

- Write a python code which declares three variables name, phone number and department and assign values and show the output as following:

"*Hello, my name is [name], my phone number is [phone number] and name of my department is [department]*"

Practice:

1. Number→ addition, subtraction, multiplication, division
2. String → concat (+)
3. Printing different type in single print function (Using comma).

Data Types

- Variables can store data of different types, and different types can do different things.
- Python has the following data types built-in by default, in these categories:

Text Type:	*str*
Numeric Types:	*int, float, complex*
Sequence Types:	*list, tuple, range*
Mapping Type:	*dict*
Set Types:	*set, frozenset*
Boolean Type:	*bool*
Binary Types:	*bytes, bytearray, memoryview*
None Type:	*NoneType*

Data Types (Examples)

Example	Data Type
`x = "Hello World"`	str
`x = 20`	int
`x = 20.5`	float
`x = 1j`	complex
`x = ["apple", "banana", "cherry"]`	list
`x = ("apple", "banana", "cherry")`	tuple
`x = range(6)`	range
`x = {"name" : "John", "age" : 36}`	dict
`x = {"apple", "banana", "cherry"}`	set
`x = frozenset({"apple", "banana", "cherry"})`	frozenset
`x = True`	bool
`x = b"Hello"`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	memoryview
`x = None`	NoneType

Numeric Data Types

- There are three numeric types in Python:
  - **int** : Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length. (*Example: x=10 , y=-2876*)
  - **float**: Float, or "floating point number" is a number, positive or negative, containing one or more decimals. Float can also be scientific numbers with an "e" to indicate the power of 10. (*Example: y=2.8, a=12E4*)
  - **complex**: Complex numbers are written with a "j" as the imaginary part. (*Example: z=1j, m=2+2j*)
- To verify the type of any object in Python, use the ***type()*** function:
  - print (type(z))

Casting

- Casting in Python refers to the process of ***converting one data type to another.***
- Python provides built-in functions to perform type conversion between different data types.
- `int()` - *constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)*
- `float()` - *constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)*
- `str()` - *constructs a string from a wide variety of data types, including strings, integer literals and float literals*

```
x = int(1)    # x will be 1
y = int(2.8)  # y will be 2
z = int("3")  # z will be 3
```

```
x = float(1)      # x will be 1.0
y = float(2.8)    # y will be 2.8
z = float("3")    # z will be 3.0
w = float("4.2")  # w will be 4.2
```

```
x = str("s1")  # x will be 's1'
y = str(2)     # y will be '2'
z = str(3.0)   # z will be '3.0'
```

Resources

- https://www.tutorialspoint.com/python/index.htm
- https://www.w3resource.com/python/python-tutorial.php
- https://www.w3resource.com/python-exercises/string/
- https://www.w3schools.com/python/
- https://www.geeksforgeeks.org/python-programming-language/
- https://youtu.be/t2_Q2BRzeEE?si=OO6J_YNCZykedqsT
- https://realpython.com/
- Head First Python, 3rd Edition by Paul Barry
- Automate the Boring Stuff with Python By Al Sweigart.

Thank You