Introduction to Programming: 3rd lesson – Data Types

Common data types in programming:

Integers, floats, Booleans, strings

Integers

* Numbers without any fractional part and can be positive (e.g. 1, 2, 3, …), negative (e.g., -1, -2, -3, …), or zero (e.g., 0).
* Example:

x = 14

print(x)

print(type(x))

14

<class ‘int’>

Floats

* Numbers with fractional parts, can have many numbers after decimal.
* Single float example:

nearly\_pi = 3.141592653589793238462643383279502884197169399375105820974944

print(nearly\_pi)

print(type(nearly\_pi))

3.141592653589793

<class ‘float’>

* Fractioned float example:

almost\_pi = 22/7

print(almost\_pi)

print(type(almost\_pi))

3.142857142857143

<class ‘float’>

* Round function:

# Round 5 to decimal places

rounded\_pi = round(almost\_pi, 5)

print(rounded\_pi)

print(type(rounded\_pi))

3.14286

<class ‘float’>

* Decimal-inputted float

y\_float = 1.

print(y\_float)

print(type(y\_float))

1.0

<class ‘float’>

Booleans

* Representative of two values: True or False.
* Example 1:

z\_one = True

print(z\_one)

print(type(z\_one))

True

<class ‘bool’>

* Example 2:

z\_two = False

print(z\_two)

print(type(z\_two))

False

<class ‘bool’>

* Example 3:

z\_three = (1 < 2)

print(z\_three)

print(type(z\_three))

True

<class ‘bool’>

* Example 4:

z\_four = (5 < 3)

print(z\_four)

print(type(z\_four))

False

<class ‘bool’>

* Example 5:

z\_five = not z\_four

print(z\_five)

print(type(z\_five))

True

<class ‘bool’>

Strings

* A collection of characters like alphabet letters, punctuation, numerical digits, or symbols contained in quotation marks, and commonly used to represent text.
* Printing strings:

w = “Hello, Python!”

print(w)

print(type(w))

Hello, Python!

<class ‘str’>

* Calculating length of string:

print(len(w))

14

* Empty string:

shortest\_string = “”

print(type(shortest\_string))

print(len(shortest\_string))

<class ‘str’>

0

* Number-inputted string:

my\_number = “1.12321”

print(my\_number)

print(type(my\_number))

1.12321

<class ‘str’>

* Float conversion:

also\_my\_number = float(my\_number)

print(also\_my\_number)

print(type(also\_my\_number))

1.12321

<class ‘float’>

* String concatenation:

new\_string = “abc” + “def”

print(new\_string)

print(type(new\_string))

abcdef

<class ‘str’>

* Multiplying a string by an integer:

newest\_string = “abc” \* 3

print(newest\_string)

print(type(newest\_string))

abcabcabc

<class ‘str’>