Basic Programming

Lesson 2. Variables in Python

Alexandra Ciobotaru

22 oct 2021

Syllabus

- Lesson 1. Computers, Programming and Cognitive Science. From pseudocode to programming languages.
- **Lesson 2. Variables in Python. Basic calculus. Using Math library, type() and help() functions.**
- Lesson 3. Working with matrices in Python. Python numpy library.
- Lesson 4. Strings. Working with strings.
- Lesson 5. Branching and decisions: Logical operators, If-Statements, Nested conditions. Loops: For and While
- Lesson 6. Collections: Lists, Tuples. Quiz 1 (25%).
- Lesson 7. Collections: Dictionaries. JSON construction.
- Lesson 8. Working with files. Reading and Writing.
- Lesson 9. Analyzing dataframes. Pandas and Matplotlib Python libraries.
- Lesson 10. Creating functions. Recursive functions. Quiz 2 (25%).
- Lesson 11. Object-Oriented Programming: Encapsulation, Inheritance and Polymorphism.
- Lesson 12. Object-Oriented Programming (cont.). Error handling. Best practices when programming.
- Lab (20%) + final exam (30%).

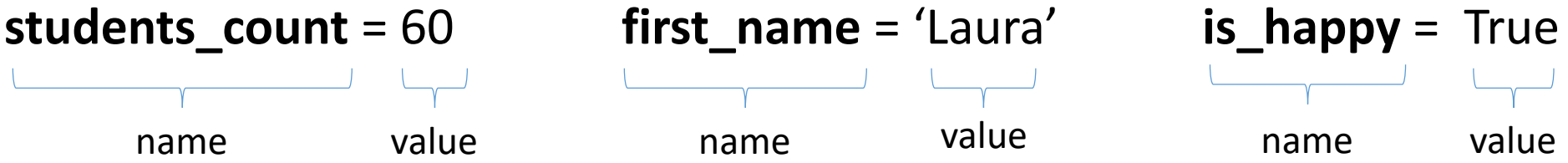
Variables

- A variable stores a value, with the scope of completing a task within our code.
- A variable is like a bucket that has a name and a value inside; the buckets can be of various types:
  - Yellow bucket – *integers (int)*
  - Red bucket – *floats (float)*
  - Green bucket – *strings (str)*
  - Black bucket – *booleans (bool)*

These are "**data types**"

- The good news is – you don't need to tell python what kind of bucket you want, he knows by inspecting the value you put into it!

- Variables consist of three different elements:
  - A symbolic name
  - An assignment operator (which in most cases is the equality sign)
  - The value we are looking to store
- Examples:



- Printing variables:
  - print("My variable is: ", var_name)
  - print("My variable is: " + var_name)
  - print("My variable is: {}".format(var_name))

Naming variables

- 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)
  - Variable names cannot be **reserved words**

Keywords

- Keywords are reserved words (have special meaning)
- Keywords cannot be used as variable names (compiler would give an error)

```
>>> help("keyword")
False               break               for                 not
None                class               from                or
True                continue            global              pass
__peg_parser__      def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
```

Built-in types - cannot be assigned as variables as well

Type Category	Type Name	Description
None	types.NoneType	The null object None
Numbers	int	Integer
	long	Arbitrary-precision integer
	float	Floating point
	complex	Complex number
	bool	Boolean (True or False)
Sequences	str	Character string
	list	List
	tuple	Tuple
	xrange	Returned by xrange()
Mapping	dict	Dictionary
Sets	set	Mutable set
	frozenset	Immutable set
Callable	type	Type of built-in types and classes
	object	Ancestor of all types and classes
Files	file	File

Built-in Data Types – what kind of ‘buckets’ are there?

Text Type:	str	x = "Hello World"
Numeric Types:	int	x = 20
	float	x = 20.5
	complex	x = 1j
Sequence Types:	list	x = ["apple", "banana", "cherry"]
	tuple	x = ("apple", "banana", "cherry")
	range	x = range(6)
Mapping Type:	dict	x = {"name" : "John", "age" : 36}
Set Types:	set	x = {"apple", "banana", "cherry"}
	frozenset	x = frozenset({"apple", "banana", "cherry"})
Boolean Type:	bool	x = True

Python id() Method

- The id() function returns an identity of an object
- In Python, all variables are objects, and each object has an unique identity as an integer number that remains constant for that object throughout its lifetime.

```
a = 5
```
-------→ creates an integer object with value 5, and a is a reference to it

```
id(a)
Out[1]: 140713644447520
```

```
b = a
```
-------→ creates a second reference to the already existing object

```
id(b)
Out[2]: 140713644447520
```

- Following the execution of these statements, Python has created one object and two references

What data type am I dealing with?

Getting the data type of our variable is done with the method type():

```
x = 5
print(type(x))
>>> <class 'int'>
```

```
x = 'Hello world!'
print(type(x))
>>> <class 'str'>
```

```
x = 5/7
print(x, type(x))
>>> 0.714, <class 'float'>
```

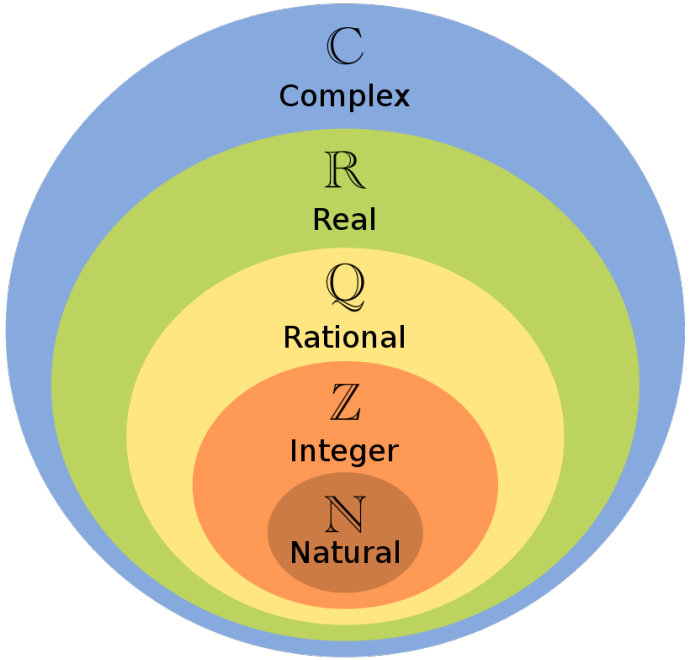
https://www.w3schools.com/python/python_datatypes.asp

Casting

- Casting means converting a variable value from one type to another by forcing it to fit in. It is done by using constructor functions:
- 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

Integers, floats and complex numbers

- Integers: 1,4,2,45,224,121434,-5,-56,-786,-5,-3424… $\mathbb{N}$ $\mathbb{Z}$
- Floats: 1.23, 5.(4), -8.98, $\sqrt{2}$, -123.9, $-\sqrt{5}$, 34.562… $\mathbb{Q}$ $\mathbb{R}$
- Complex: a + bi (a is the real part and b the imaginary part) $\mathbb{C}$



Strings (we'll talk more on this on Lesson 4)

- Strings are declared like this:
  - x = 'some characters, word or sentence'
  - x = "some characters, word or sentence"
- Strings can be **added** and **multiplied** (but not substracted!).
  - "Adding" character strings concatenates them.
  - Multiplying a character string by an integer replicates it (Since multiplication is just repeated addition.)

```
>>> a = 'coffee '
>>> b = 'sugar'
>>> print(a+b)
coffee sugar
>>> print(a*2)
coffee coffee
>>> print(b*5)
sugarsugarsugarsugarsugar
>>> print(len(b*5))
25
```

Strings have a length (but numbers don't).

- The built-in function *len* counts the number of characters in a string.

Booleans

- Booleans are variables which can have one of the values: True (with capital T) and False (with capital F)

```
>>> my_boolean = (3<6)
>>> print(my_boolean)
True
```

- == declaration – checks if the variable has that value and returns True or False

```
>>> a = 5
>>> a == 5
True
>>> a == 7
False
```

Constants

- A constant is a type of variable that holds a value, which cannot be changed.
- In reality, we don’t use constants in Python, but we declare some variables that we leave unchanged throughout our script.
- To know that those values needn’t be changed, it’s a good practice to write them in capital letters at the beginning of the code:

```
MAX_LEN = 60
PI = 3.14
GRAVITY = 9.8
```

Thank you!