Programming Fundamentals

Variables, expressions and statements

João Correia Lopes

INESC TEC, FEUP

27 September 2018

Goals

By the end of this class, the student should be able to:

- Describe and distinguish the concepts of variable, location, value, type
- Identify the Python reserved words
- Describe the concepts of statement and expression
- Identify some of the Python operands and their precedence
- Use operators with suitable operands
- Describe the Python cast operations and use them
- Describe how to get input, at runtime, from an user of the program

BIBLIOGRAPHY

- Peter Wentworth, Jeffrey Elkner, Allen B. Downey, and Chris Meyers, How to Think Like a Computer Scientist — Learning with Python 3, 2018 (Section 2.1) [PDF]
- Brad Miller and David Ranum, Learning with Python: Interactive Edition. Based on material by Jeffrey Elkner, Allen B. Downey, and Chris Meyers (Chapter 2) [HTML]

Tips

- There's no slides: we use a script and some illustrations in the class. That is NOT a replacement for **reading the bibliography** listed in the *class sheet*
- "Students are responsible for anything that transpires during a class—therefore **if you're not in a class**, you should get notes from someone else (not the instructor)"—David Mayer
- The best thing to do is to **read carefully** and **understand** the documentation published in the Content wiki (or else **ask** in the class)
- We will be using **Moodle** as the primary means of communication

Contents

PYTHON

```
1  #!/usr/bin/env python3

3  import datetime
   now = datetime.datetime.now()
5
   print()
7  print("Current date and time using str method of datetime object:")
   print()
9  print(str(now))

11 print()
   print("Current date and time using instance attributes:")
13 print()
   print("Current year: %d" % now.year)
15 print("Current month: %d" % now.month)
   print("Current day: %d" % now.day)
17 print("Current hour: %d" % now.hour)
   print("Current minute: %d" % now.minute)
19 print("Current second: %d" % now.second)
   print("Current microsecond: %d" % now.microsecond)
21
   print()
23 print("Current date and time using strftime:")
   print(now.strftime("%Y-%m-%d %H:%M"))
```

⇒ https://github.com/fpro-admin/lectures/blob/master/01/basics.py

Values and data types

- A **value** is one of the fundamental things that a program manipulates
- Values are classified into different classes, or **data types**
- **type()** is a function that tell us the type of a value

VARIABLES

- A variable is a name that refers to a value
- The **assignment statement** gives a value to a variable
- The assignment statement binds a *name*, on the left-hand side of the operator, to a *value*, on the right-hand side
- Later, one can assign a different value to the same variable (this is different from maths!)
- The assignment token, =, should not be confused with the equals token, ==

⇒ Visualise a **state snapshot**

Variable names

- **Variable names** can be arbitrarily long
- They can contain both letters and digits, but they have to begin with a letter or an underscore
- It is legal to use uppercase letters, but it is not done (by convention)
- Names should be “meaningful to the human readers” (not to be confused with “meaningful to the computer”)

KEYWORDS

- Keywords define the language's syntax rules and structure
- They cannot be used as variable names

and	as	assert	break	class	continue
def	del	elif	else	except	exec
finally	for	from	global	if	import
in	is	lambda	nonlocal	not	or
pass	raise	return	try	while	with
yield	True	False	None		

STATEMENTS

- A **statement** is an instruction that the Python interpreter can execute
- Statements don't produce any result
- Further to the assignment statement, there are others (`while` statements, `for` statements, `if` statements, `import` statements)

EVALUATING EXPRESSIONS

- An **expression** is a combination of values, variables, operators, and calls to functions
- The Python interpreter evaluates expressions and displays its result (a value)
- A value all by itself is a simple expression, and so is a variable

Operators and operands

- **Operators** are special tokens that represent computations like addition, multiplication and division
- The values the operator uses are called **operands**
- When a variable name appears in the place of an operand, it is replaced with its value before the operation is performed
- Operations in Python (+, −, /) mean what they mean in mathematics
- Asterisk ($\star$) is the token for multiplication, and $\star\star$ is the token for exponentiation

Time for Kahoot!

Not now!

⇒ https://kahoot.com/

TYPE CONVERTER FUNCTIONS

- Type converter functions `int()`, `float()` and `str()`
- will (attempt to) convert their arguments into types `int`, `float` and `str` respectively

Order of operations

- When more than one operator appears in an expression, the order of evaluation depends on the rules of precedence
- Python follows the same precedence rules for its mathematical operators that mathematics does (PEM-DA-S)
- Operators with the same precedence are evaluated from left-to-right (*left-associative*)
- An exception to the left-to-right left-associative rule is the exponentiation operator $**$

OPERATIONS ON STRINGS

- One cannot perform mathematical operations on strings, even if the strings look like numbers
- The + operator represents concatenation, not addition
- The $\star$ operator also works on strings; it performs repetition

INPUT

- There is a built-in function in Python, `input()`, for getting input from the user
- The user of the program can enter the input and click OK
- The `input()` function always return a string (without the new-line)

COMPOSITION

- One of the most useful features of programming languages is their ability to take small building blocks and **compose** them into larger chunks.
- Let's do four-step program that asks the user to input a value for the radius of a circle, and then computes the area of the circle from the formula πr^2

- TIP: try to make code as simple as you can for the human to follow

⇒ https://github.com/fpro-admin/lectures/blob/master/02/area.py

THE MODULUS OPERATOR

- The **modulus operator** works on integers (and integer expressions)
- and gives the remainder when the first number is divided by the second
- In Python, the modulus operator is a percent sign (`%`)
- It has the same precedence as the multiplication operator

⇒ https://github.com/fpro-admin/lectures/blob/master/02/remainder.py

EXERCISES

- Moodle activity at: LE02: Variables, expressions and assignments