universidade de aveiro

deti departamento de electrónica, telecomunicações e informática

Glossários

How to think like a computer scientist

Gonçalo Matos | MEC 92972

Licenciatura em Engenharia Informática

1.º Semestre | Ano letivo 2018/2019

Índice

1. General introduction

activecode

A unique interpreter environment that allows Python to be executed from within a web browser.

algorithm

A general step by step process for solving a problem.

bug

An error in a program.

byte code

An intermediate language between source code and object code. Many modern languages first compile source code into byte code and then interpret the byte code with a program called a *virtual machine*.

codelens

An interactive environment that allows the user to control the step by step execution of a Python program

comment

Information in a program that is meant for other programmers (or anyone reading the source code) and has no effect on the execution of the program.

compile

To translate a program written in a high-level language into a low-level language all at once, in preparation for later execution.

debugging

The process of finding and removing any of the three kinds of programming errors.

exception

Another name for a runtime error.

executable

Another name for object code that is ready to be executed.

formal language

Any one of the languages that people have designed for specific purposes, such as representing mathematical ideas or computer programs; all programming languages are formal languages.

high-level language

A programming language like Python that is designed to be easy for humans to read and write.

interpret

To execute a program in a high-level language by translating it one line at a time.

low-level language

A programming language that is designed to be easy for a computer to execute; also called machine language or assembly language.

natural language

Any one of the languages that people speak that evolved naturally.

object code

The output of the compiler after it translates the program.

parse

To examine a program and analyze the syntactic structure.

portability

A property of a program that can run on more than one kind of computer.

print function

A function used in a program or script that causes the Python interpreter to display a value on its output device.

problem solving

The process of formulating a problem, finding a solution, and expressing the solution.

program

A sequence of instructions that specifies to a computer actions and computations to be performed.

programming language

A formal notation for representing solutions.

Python shell

An interactive user interface to the Python interpreter. The user of a Python shell types commands at the prompt (>>>), and presses the return key to send these commands immediately to the interpreter for processing.

runtime error

An error that does not occur until the program has started to execute but that prevents the program from continuing.

semantic error

An error in a program that makes it do something other than what the programmer intended.

semantics

The meaning of a program.

shell mode

A style of using Python where we type expressions at the command prompt, and the results are shown immediately. Contrast with **source code**, and see the entry under **Python shell**.

source code

A program, stored in a file, in a high-level language before being compiled or interpreted.

syntax

The structure of a program.

syntax error

An error in a program that makes it impossible to parse — and therefore impossible to interpret.

token

One of the basic elements of the syntactic structure of a program, analogous to a word in a natural language.

2. Simple Python Data

assignment statement

A statement that assigns a value to a name (variable). To the left of the assignment operator, `=`, is a name. To the right of the assignment token is an expression which is evaluated by the Python interpreter and then assigned to the name. The difference between the left and right hand sides of the assignment statement is often confusing to new programmers. In the following assignment:

```
n = n + 1
```

`n` plays a very different role on each side of the `=`. On the right it is a *value* and makes up part of the *expression* which will be evaluated by the Python interpreter before assigning it to the name on the left.

assignment token

`=` is Python's assignment token, which should not be confused with the mathematical comparison operator using the same symbol.

class

see **data type** below

comment

Information in a program that is meant for other programmers (or anyone reading the source code) and has no effect on the execution of the program.

data type

A set of values. The type of a value determines how it can be used in expressions. So far, the types you have seen are integers (`int`), floating-point numbers (`float`), and strings (`str`).

decrement

Decrease by 1.

evaluate

To simplify an expression by performing the operations in order to yield a single value.

expression

A combination of operators and operands (variables and values) that represents a single result value. Expressions are evaluated to give that result.

float

A Python data type which stores *floating-point* numbers. Floating-point numbers are stored internally in two parts: a *base* and an *exponent*. When printed in the standard format, they look like decimal numbers. Beware of rounding errors when you use `float`s, and remember that they are only approximate values.

increment

Both as a noun and as a verb, increment means to increase by 1.

initialization (of a variable)

To initialize a variable is to give it an initial value. Since in Python variables don't exist until they are assigned values, they are initialized when they are created. In other programming languages this is not the case, and variables can be created without being initialized, in which case they have either default or *garbage* values.

int

A Python data type that holds positive and negative **whole** numbers.

integer division

An operation that divides one integer by another and yields an integer. Integer division yields only the whole number of times that the numerator is divisible by the denominator and discards any remainder.

keyword

A reserved word that is used by the compiler to parse program; you cannot use keywords like `if`, `def`, and `while` as variable names.

modulus operator

Also called remainder operator or integer remainder operator. Gives the remainder after performing integer division.

object

Also known as a data object (or data value). The fundamental things that programs are designed to manipulate (or that programmers ask to do things for them).

operand

One of the values on which an operator operates.

operator

A special symbol that represents a simple computation like addition, multiplication, or string concatenation.

prompt string

Used during interactive input to provide the user with hints as to what type of value to enter.

reference diagram

A picture showing a variable with an arrow pointing to the value (object) that the variable refers to. See also **state snapshot**.

rules of precedence

The set of rules governing the order in which expressions involving multiple operators and operands are evaluated.

state snapshot

A graphical representation of a set of variables and the values to which they refer, taken at a particular instant during the program's execution.

statement

An instruction that the Python interpreter can execute. So far we have only seen the assignment statement, but we will soon meet the `import` statement and the `for` statement.

str

A Python data type that holds a string of characters.

type conversion function

A function that can convert a data value from one type to another.

value

A number or string (or other things to be named later) that can be stored in a variable or computed in an expression.

variable

A name that refers to a value.

variable name

A name given to a variable. Variable names in Python consist of a sequence of letters (a..z, A..Z, and _) and digits (0..9) that begins with a letter. In best programming practice, variable names should be chosen so that they describe their use in the program, making the program *self documenting*.

4. Turtle Graphics

Method	Parameters	Description
Turtle	None	Creates and returns a new turtle object
forward	distance	Moves the turtle forward
backward	distance	Moves the turle backward
right	angle	Turns the turtle clockwise
left	angle	Turns the turtle counter clockwise
up	None	Picks up the turtles tail
down	None	Puts down the turtles tail
color	color name	Changes the color of the turtle's tail
fillcolor	color name	Changes the color of the turtle will use to fill a polygon
heading	None	Returns the current heading
position	None	Returns the current position
goto	x,y	Move the turtle to position x,y
begin_fill	None	Remember the starting point for a filled polygon
end_fill	None	Close the polygon and fill with the current fill color
dot	None	Leave a dot at the current position
stamp	None	Leaves an impression of a turtle shape at the current location
shape	shape name	Should be 'arrow', 'classic', 'turtle', 'circle' or 'square'

6. Functions

chatterbox function

A function which interacts with the user (using `input` or `print`) when it should not. Silent functions that just convert their input arguments into their output results are usually the most useful ones.

composition (of functions)

Calling one function from within the body of another, or using the return value of one function as an argument to the call of another.

dead code

Part of a program that can never be executed, often because it appears after a `return` statement.

fruitful function

A function that yields a return value instead of `None`.

incremental development

A program development plan intended to simplify debugging by adding and testing only a small amount of code at a time.

None

A special Python value. One use in Python is that it is returned by functions that do not execute a return statement with a return argument.

return value

The value provided as the result of a function call.

scaffolding

Code that is used during program development to assist with development and debugging. The unit test code that we added in this chapter are examples of scaffolding.

temporary variable

A variable used to store an intermediate value in a complex calculation.

7. Selection

block

A group of consecutive statements with the same indentation.

body

The block of statements in a compound statement that follows the header.

boolean expression

An expression that is either true or false.

boolean function

A function that returns a boolean value. The only possible values of the `bool` type are `False` and `True`.

boolean value

There are exactly two boolean values: `True` and `False`. Boolean values result when a boolean expression is evaluated by the Python interepreter. They have type `bool`.

branch

One of the possible paths of the flow of execution determined by conditional execution.

chained conditional

A conditional branch with more than two possible flows of execution. In Python chained conditionals are written with `if ... elif ... else` statements.

comparison operator

One of the operators that compares two values: `==`, `!=`, `>`, `<`, `>=`, and `<=`.

condition

The boolean expression in a conditional statement that determines which branch is executed.

conditional statement

A statement that controls the flow of execution depending on some condition. In Python the keywords `if`, `elif`, and `else` are used for conditional statements.

logical operator

One of the operators that combines boolean expressions: `and`, `or`, and `not`.

modulus operator

An operator, denoted with a percent sign (`%`), that works on integers and yields the remainder when one number is divided by another.

nesting

One program structure within another, such as a conditional statement inside a branch of another conditional statement.

8. More about iteration

algorithm

A step-by-step process for solving a category of problems.

body

The indented statements after a heading ending in a colon, for instance after a for-loop heading.

counter

A variable used to count something, usually initialized to zero and incremented in the body of a loop.

cursor

An invisible marker that keeps track of where the next character will be printed.

definite iteration

A loop where we have an upper bound on the number of times the body will be executed. Definite iteration is usually best coded as a `for` loop.

escape sequence

An escape character, \, followed by one or more printable characters used to designate a nonprintable character.

generalize

To replace something unnecessarily specific (like a constant value) with something appropriately general (like a variable or parameter). Generalization makes code more versatile, more likely to be reused, and sometimes even easier to write.

infinite loop

A loop in which the terminating condition is never satisfied.

indefinite iteration

A loop where we just need to keep going until some condition is met. A `while` statement is used for this case.

iteration

Repeated execution of a set of programming statements.

loop

A statement or group of statements that execute repeatedly until a terminating condition is satisfied.

loop variable

A variable used as part of the terminating condition of a loop.

nested loop

A loop inside the body of another loop.

newline

A special character that causes the cursor to move to the beginning of the next line.

reassignment

Making more than one assignment to the same variable during the execution of a program.

tab

A special character that causes the cursor to move to the next tab stop on the current line.

11. Files

open

You must open a file before you can read its contents.

close

When you are done with a file, you should close it.

read

Will read the entire contents of a file as a string. This is often used in an assignment statement so that a variable can reference the contents of the file.

readline

Will read a single line from the file, up to and including the first instance of the newline character.

readlines

Will read the entire contents of a file into a list where each line of the file is a string and is an element in the list.

write

Will add characters to the end of a file that has been opened for writing.

absolute file path

The name of a file that includes a path to the file from the *root* directory of a file system. An *absolute file path* always starts with a `/`.

relative file path

The name of a file that includes a path to the file from the current working directory of a program. An *relative file path* never starts with a `/`.