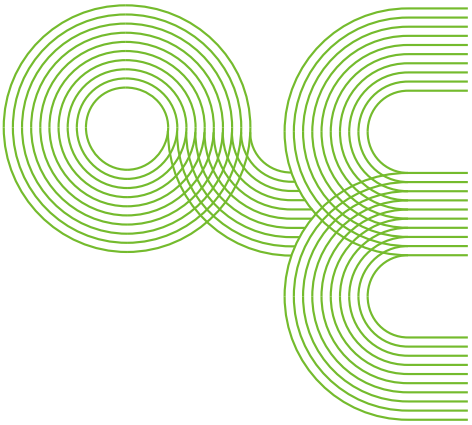
Universidade Vigo

Computer Science / Informatics – Exercises



Escola de Enxeñaría Aeronáutica e do Espazo
Pavillón Manuel Martínez-Risco
Campus universitario
32004 Ourense

http://aero.uvigo.es
mailto:aero.info@uvigo.es

Referencia: 1.0
Documento: practicas-inf
Fecha: 28 de noviembre de 2017
Páginas: 9

Índice

First steps

Objectives: Work with python. Get to know simple data types. Perform simple computations with simple data types. Introduce and visualize data using keyboard and terminal. Use character strings.

1. Execute IDEL (Python 3.5), write the command `print("hello world")`. Save the file as `E1_exercise1.py`, execute the program.

2. Try the following program that visualizes some data types of python:

```
age = 20
price = 49.5
dni = "12345678Z"
fellowship = True
complex = 1.3 + 2.5j
print(age, price, dni, fellowship, complex)
```

   Modify the previous program changing the values of some of the variables and revisualize on the terminal. Check that it is possible to change the type of a variable by simply assigning a value of a different type.

3. Write a program that performs the following operations:

   a) Compute the mean value of two floating point numbers.
   b) Show this mean value on the terminal.
   c) To compute the final price of an article, assign to a variable a base price (neto) of, for instance, 100 euros, and to another variable the tax of 21 %, compute the final price of the article as base + tax.
   d) Show the price on the terminal.
   e) Assign to a variable an age, and in another variable store whether the age stands for an adult or not.
   f) Show the result on the terminal.
   g) Compute the rest that remains by an integer division, show it on the terminal.
   h) Compute some exponential of a value and visualize its value.
   i) Assign in one python command both the weight and the height to the corresponding variables. Visualize on screen.

4. The following program shows some examples how python treats character strings (text). Analize the syntax and usage of the different operators and functions.

```
url_school = "aero.uvigo.es"
print(url_school[2])
print(url_school[5:10])
user = "formella"
```

```
domain = "uvigo.es"
print(user + "@" + domain)
line = 80 * "-"
print(line)
dni = "12345678Z"
print(len(dni))
code = "OU-" + str(32000)
print(code)
print(code.lower())
city = "Ourense"
print(city.upper())
print(url_school.split("."))
print(url_school.split(".")[1])
print(url_school.find("."))
print(url_school.replace("."," "))
```

---

5. Complete the following program to calculate the letter of a dni:

```
letterset_dni = "TRWAGMYFPDXBNJZSQVHLCKE"
dni = ..............
posicion = dni%23
letter_dni = .........
NIF = ..............
print("NIF = ",NIF)
```

---

6. Write a python program that asks for a value in degrees Fahrenheit and shows the value converted to degrees Celsius. You should use the following formula: `degrees_celsius = ( degrees_fahrenheit - 32 ) / 1.8`
7. Write a python program that reads-in a first name, a name, and a dni of a person and outputs the data on the terminal.
8. Modify the previous program such that it uses the format method.
9. Write a python program that reads-in a first name and name and generates (and visualizes) the corresponding email address where the address is built by the initial followed by the second name (assuming spanish naming conventions) and ended by `@alumnos.uvigo.es`.
10. Write a python program that reads-in the coordinates x and y of a point and visualizes them in the format (x, y) on screen.
11. Write a python program that computes the distance between two points.
12. Write a python program that computes the circumference and the area of a circle given its radius,

First algorithms and input/output

Objectives: Work with first simple algorithm with flow control and loops.

1. Write a program that reads-in a numerical value and visualizes the value as text on the terminal.
2. Write a program that reads-in a numerical value and visualizes whether the value is odd or even.
3. Write a program that reads-in the age of a person visualizes whether the person is an adult or not.
4. Write a program that visualizes which one of two input numbers is the larger one. Extend the program to work for three numbers.
5. Write a small calculater, i.e., a program that accepts three parameters: two numbers and a symbol. If the symbol is a + the values are added, if it is a - the values are subtracted, likewise `*`, `/`, and `^` (exponential).
6. Write a program that asks for a number and outputs all integer square-numbers less than the given value.
7. Write a program that computes the mean value of a series of numbers given by the user. Think about a method how to determine that the user wants to stop input.
8. Write a program that finds the minor and mayor value of a series of numbers given by the user.
9. Write a program that shows on the terminal the corresponding multiplication table (for instance: `1*7=7`, `2*7=14`, etc.).
10. Modify the previous program such that the output is nicely formatted in column format.
11. Write a program that shows whether a given number is prime or not.

Functions

Objectives: Using functions in python. Reinforcement of loops. More input/output. Automatic checking.

1. Try the following program that uses a function to compute a weighted average of three grades as final grade.

```
def compute_grade(test1, test2, exercises):
    return test1*0.35 + test2*0.35 + exercises*0.3

print(compute_grade(7.5, 8.5, 7))
print(compute_grade(4.7, 7.2, 6.8))
```

2. Observe how the following program, beside using a function which encapsulates the input operation, that function returns three grades given by the user:

```
def ask_for_grades():
    exam1 = float(input("Grade 1. exam: "))
    exam2 = float(input("Grade 2. exam: "))
    exercises = float(input("Grade exercises: "))
    return exam1, exam2, exercises

def show_grades(test1, test2, exercises):
    final = test1*0.35 + test2*0.35 + exercises*0.3
    print("Grade final: {:.2f}".format(final))

grade1, grade2, grade3 = ask_for_grades()
show_grades(grade1,grade2,grade3)
```

3. The following program asks for new input and visualizes the grades until the user indicates to finish.

```
def ask_for_grades():
    exam1 = float(input("Grade 1ª exam: "))
    exam2 = float(input("Grade 2ª exam: "))
    exercises = float(input("Grade exercises: "))
    return exam1, exam2, exercises

def show_grades(test1, test2, exercises):
    final = test1*0.35 + test2*0.35 + exercises*0.3
    print("Final grade: {:.2f}".format(final))
```

```
repeat = "yes"
while (repeat != "no"):
    grade1, grade2, grade3 = ask_for_grades()
    show_grades(grade1,grade2,grade3)
    repeat = input("keep going? (yes/no) ")
    repeat = repeat.lower()
else:
    print("\nEnd of program")
```

---

4. Modify the previous program such that any negative input terminates implicitly the input loop (clearly, this last entry should not participate in the average).
5. Modify the function `show_grades()` such that it accepts weights as default parameters. Make experiments with your function.
6. Write a function to compute the Collatz sequence of a positive integer number a_0, where for such a sequence we have $a_{n+1} = 1/2 \cdot a_n$, if a_n is even, and $a_{n+1} = 3 \cdot a_n + 1$, if a_n is odd. The sequence terminates whenever we arrive at the number 1. For instance, for $a_0 = 3$ we obtain the sequence: 3, 10, 5, 16, 8, 4, 2, 1. (Note: no one knows whether we reach 1 for all positive integer numbers. Collatz conjecture.)
7. Write a function that computes the factorial of a number and check your function in a program.
8. Write a function to check whether a year is a leap year and check your function in a program.
9. Write a program to play the game stone-paper-scissors. The program should generate a random value and ask the user for his or her choice. Declare as winner who first has won three rounds.
10. Write a program with functions that finds the roots of a polynomial of degree 2 ($ax^2 + bx + c = 0$). Asks the user for the coefficients as floating point numbers.
11. Write a program that uses a function to convert polar coordinates to rectangular coordinates.
12. Write a program that uses functions to convert cylindric coordinates to rectangular coordinates and the other way round. Use your functions to check whether your implementation is correct, as we have $cyl2rec(rec2cyl(x,y,z)) = (x,y,z)$, don't we?

Simple data structures and a complete program

Objectives: Using simple data structures such as lists, tuples, sets, and dictionaries in python. Reinforcement of loops and functions. Extension of input/output with files. Re-using strings.

1. Create a file that contains pairs of passengers and seats. In each line there should appear first the name of the person and then an identifier of the seat, separated by a comma, for example:

   ```
   Maria, 2A
   Manuel, 21D
   Dorothea, 14F
   Samuel, 10C
   ```

   Write a program that reads-in the file and converts it automatically into a file that contains in its first line the identifiers of the seats and in its second line the corresponding names. (Extension: try to sort the list according to the seat identifier.)

2. Generate a small reservation system for an airplane A320. The program should work with a main menu where an entry of a single character selects one action, for instance, `r` activates the reservation module, `l` activates the list generation module, `q` quits the program.

   The menu should contain at leat the following modules:

   - read a file (in the same format as given above)
   - list all occupied seats
   - add a reservation given a name and a seat number (take care, only one passenger per seat, and only available seats)
   - store the reservations into a file
   - cancel a reservation (either by passenger name or by seat number)
   - show unoccupied seats
   - extend your program with functions you consider useful (especially for debugging purposes).

   In order not to make things too complicated, assume that the seat numbers are consecutive and there are always the same number of seats in a row. Make good use of the data structures you know, for instance, use may use a tuple to join a passenger and a seat, a dictionary for the occupation of the airplane, sets for the characters of the menu etc. You will realize: the more you develop a modular program, the easier you will succeed.

   Consider working in small groups, where each participant contributes one or several modules.

Recursive functions

Objectives: Usage of recursion, observation of the pro's and con's, analysis of the program flow, use of simple data structures to store intermediate results to speed-up computations.

1. Write a recursive function to calculate a binomial coefficient $\binom{n}{k}$ using the formula

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

   - Ask the user for the values of n and k.
   - Use a main loop to compute and visualize the results until the user wants to stop (for instance, giving a negative number for n).
   - Make experiments to figure out what are the largest input values such that python is able to compute the coefficients using this recursive method in a reasonable amount of time (What is the largest k respective to n that generate the largest coefficient?).

2. Write a recursive function to compute the nth value of the Fibonacci series (remember, the series starts with: 1,1,2,3,5,8,13,21,34,...). Not considering the first two values $f_0 = 1$ and $f_1 = 1$ Quitando los dos casos iniciales $f_0 = 1$ y $f_1 = 1$, the recursion formula is $f_n = f_{n-1} + f_{n-2}$.

   - Ask the user for a value of n.
   - Use a main loop to compute and visualize the result until the user wants to stop (for instance, giving a negative number for n).
   - Make experiments to figure out what is the largest input value such that python is able to compute the series using this recursive method in a reasonable amount of time.

3. Analize the function calls your programs are performing, i.e., try to find an answer to the question: How many times a recursive function is called?

   Improve your programs considerably (i.e., the run times) with the help of simple data structures that store intermediate results such that simple look-ups are sufficient, rather than recalculation of a value (for the Fibonacci series we saw it in class).