1. Assignment #2: Simple data

**Master in Informatics and Computing Engineering  
Programming Fundamentals  
Instance: 2018/2019**

# 0. Introduction

***Goals****: to write programs using variables, operations using simple data, input and output.*

***Pre-requirements (prior knowledge)****: Basic data and operations as in Lecture #2*

***Rules****: You may work with colleagues, however, each student must write and submit in Moodle his or her this assignment separately. Be sure to indicate with whom you have worked. We may run tools to detect plagiarism (e.g.duplicate code submitted).*

***Deadline****: 8:00 Monday of the week after (08/10/2018)*

***Collaborators****:*

*list here their codes (2018nnnn1, 2018nnnn2, …)*

# 1. Variables and print

Take the sentence: “*The quick brown fox jumps over the lazy dog*”.

Use Spyder3 to create a file called fox.py in your “Working space” (see Assignment #1).

Store each word of the sentence in a separate variable and then print the sentence using the variables you created.

Then, run the program and copy the result followed by your program (fox.py) here:

# copy of the result  
The quick brown fox jumps over the lazy dog

# copy of the program

a - declarar uma variável contendo a nossa string

foo = 'The quick brown fox jumps over the lazy dog'

# b - declarar uma função que vai receber uma string como argumento

def inlineWords(str):

# separar cada palavra da string e metê-la numa lista

listOfWords = str.split()

# para cada palavra da posição n da lista de palavras

# guardar essa palavra numa tabela de simbolos local

# e imprimir essas palavras na mesma linha

for n, word in enumerate(listOfWords):

locals()[f'palavra{n}'] = word

print(word, end=' ')

# c - chamar a função

inlineWords(foo)

# 2. Input and string operators

Use Spyder3 to create a file called html.py and write a program that asks a user to input a tag h and a string text and prints an HTML valid element of the form *<tag>text</tag>*.

For example, for a user input tag *h1* and a text “*I’m an HTML text*”, the output must be “*<h1>I’m an HTML text</h1>*”

Then, run the program for a user input tag *b* and *html’s* as text, and copy the result followed by your program (the content of the file html.py) here:

# copy of the result

Escreva aqui o tipo de header: b

Escreva aqui o texto: html's

<b>html's</b>

# copy of the program

# a - declaração das vars que recebem o input do header

# e do texto respetivamente

headerTag = input('Escreva aqui o tipo de header: ')

headerTxt = input('Escreva aqui o texto: ')

# b - print do resultado injetando as variaveis criadas

# anteriormente numa formatted string

print(f'<{headerTag}>{headerTxt}</{headerTag}>')

# 3. Input and cast

Use Spyder3 to create a file called cast.py and write a program that asks a user to input a number n, calculates the expression n + nn + nnn and prints its value.

For example, for a user input 5, the output must be 5 + 55 + 555 = 615.

Then, run the program for a user input **9**, and copy the result followed by your program (the content of the file cast.py) here:

# copy of the result  
Introduza um valor numérico: 9

9 + 99 + 999 = 1107

# copy of the program  
# a - declarar a var que recebe o input do tipo Int

userInput = int(input('Introduza um valor numérico: '))

# b - declarar uma função que recebe um parâmetro <n>

# e imprime o resultado de n + nn + nnn

def funkySum(n):

print(f'{n} + {n}{n} + {n}{n}{n} = ', int(n) + int("%d%d" % (n, n)) + int("%d%d%d" % (n, n, n)))

# c - chamar a função

funkySum(userInput)

# 4. Operations and operands

The formula for computing the final amount if one is earning compound interest is given on Wikipedia as:

Where:

* P = principal amount (the amount that the interest is provided on)
* r = the interest rate
* n = the frequency that the interest is paid out (per year)
* t = the number of years that the interest is calculated for

Use Spyder3 to create a file called interests.py and write a program that replaces these letters with something a bit more human-readable, and calculate the final amount (A) at the end of the second year, for some varying amounts of money (P) at realistic interest rates (r):

1. P = 1000, n = 2, and r = 1%
2. P = 650, n = 1 and r = -0.05%
3. Values of P, n and r introduced by the user

Then, run the program, and copy the result followed by your program (the content of the file interests.py) here:

# copy of the result  
Principal Amount: 1000

Interest Rate: 0.01

Frequency: 2

Final amount: 1020.1505006249996

# copy of the program

# a - declarar variáveis que recebem o input do utilizador

userPrAmount = int(input('Principal Amount: '))

userInRate = float(input('Interest Rate: '))

userFrq = int(input('Frequency: '))

# b - declarar uma função que recebe como parâmetro essas variáveis

# (quantia principal, juros e frequência de pagamento) e devolve

# o resultado da aplicação da fórmula

def finalAmount(prAmount, inRate, frq):

return prAmount \* ((1 + (inRate/frq))) \*\* (frq \* 2)

# c - print da chamada da função com o resultado

print('Final amount: ', finalAmount(userPrAmount, userInRate, userFrq))

# 5. Remainder

Use Spyder3 to create a file called alarm\_clock.py and write a program that determines what is the actual time (using now of class datetime from module datetime) and prints it in the form *hh:mm* (hours and minutes).

Given that an alarm is set up for *8 hours and 30 minutes* later, the program prints the time on display at the time of the alarm, in the same format.

Then, run the program, and copy the result followed by your program (the content of the file alarm\_clock.py) here:

# copy of the result  
*Hora(s): 8*

*Minuto(s): 30*

*Agora: 00:40:08*

*Alarme: 09:10:08*

# copy of the program

# a - fazer import do módulo <datetime>

import datetime

# b - declarar uma função que calcula e imprime a hora do

# alarme tendo em conta as horas e os minutos introduzidos

# pelo user

def setAlarm(hour, minute):

# objeto datetime do dia de hoje

today = datetime.datetime.now()

# objeto datetime para o alarme

alarm = today + datetime.timedelta(hours=hour, minutes=minute)

# converter para String e formatar em hora:minuto:segundo

currTime = today.strftime('%H:%M:%S')

alarmTime = alarm.strftime('%H:%M:%S')

# imprimir resultado

print(f'Agora: {currTime} \nAlarme: {alarmTime}')

# c - declarar variaveis que guardam input do user (horas e minutos

# até ao alarme disparar e chamar a func com esse input

hoursToTrigger = int(input('Hora(s): '))

minutesToTrigger = int(input('Minuto(s): '))

setAlarm(hoursToTrigger, minutesToTrigger)

# **The end.**

*FPRO, 2018/19*