Programming and algorithmic exercises

Aerospatial engineering
2024-2025

Part I

Python programming

Chapter 1

Basic input/output and Math

1.1 Basic input/output

1. **Basic I/O operations**
   (a) Write programs that print the following message in the output console

   ```
   "Hello world"
   ```

   (b) Write a program to receive the first and the last name of a user and reverse their order., e..g "Antonio Silva" => "Silva, Antonio"

1.2 Basic Math

1. **Integers**
   (a) Write a Python program that takes two integers as input from the user and calculates their sum, difference, product, and quotient (integer division) numbers.

   (b) Create a Python program that reads an integer from the user and checks if it's even or odd. Print a message indicating whether it's even or odd.

   (c) Create a Python program that reads an integer with exactly three digits and reverse their order;

2. **Integers (different basis)**
   (a) Create a Python program that reads three numbers in hexadecimal sums the first two, substracts the third and shows the result:
      i. in hexadecimal;
      ii. in octal;
      iii. in binary.

   (b) Create a Python program that reads a number in binary and shows it

        i. In octal;
        ii. In hexadecimal.
    (c) Create a Python program to count the number of 1s in a binary number of 4 digits. [**HARD**]

3. **Floating point**
    (a) Write a Python program that takes two floating-point numbers as input and calculates their average (mean). Print the average.
    (b) Build a Python program that reads a decimal number (floating-point) from the user and rounds it to a specified number of decimal places. The number of decimal places should also be entered by the user. Print the rounded value.
    (c) Build a Python program that reads a decimal number (floating-point) from the user and calculates its square root.
    (d) Build a Python program that reads four numbers, where the first two define the point (x_1, y_1) and the second ones the point (x_2, y_2). Calculate and show the distance between them.
    (e) Design a program that converts currency amounts between two different currencies. Ask the user for the exchange rate and the amount in one currency, then calculate and display the equivalent amount in the other currency.

4. **Physics**
    (a) Calculate a program to convert Celsius degrees into Farhenheit. The formula is as follows:

$$F = \left(\frac{9}{5}C\right) + 32$$

where C and F are given by the user.

1.3 Aerospace

Some problems in aerospace that only require basic mathematics

1.3.1 Airplane fuel efficiency calculator

You are tasked with creating a program that calculates the fuel efficiency of an airplane for a given flight. The user will provide the following inputs:

- Distance of the flight in kilometers (as a floating-point number).
- Amount of fuel consumed during the flight in liters (as a floating-point number).

Your **program** should perform the following steps:

- Read the distance traveled and the amount of fuel consumed from the user.

- Calculate the fuel efficiency (in liters per kilometer) by dividing the amount of fuel consumed by the distance traveled.
- Calculate the fuel efficiency in miles per gallon (1 liter = 0.264172 gallons, 1 kilometer = 0.621371 miles).
- Display the fuel efficiency in liters per kilometer and miles per gallon to the user.

1.3.2 Altitude Conversion and Display

Imagine you are designing a basic altitude display system for an airplane. The system's purpose is to convert altitude from meters to feet and display it to the pilot.

The system operates as follows:

- Read the altitude in meters from a sensor (as a floating-point number).
- Convert the altitude from meters to feet using the conversion factor: 1 meter = 3.28084 feet.
- Display the converted altitude in feet to the pilot.

1.3.3 Airspeed Calculator

Create a program that calculates the true airspeed (TAS) of an airplane based on the indicated airspeed (IAS) and altitude. The user should input IAS in knots and altitude in feet. Use the following formula:

$$TAS = IAS \times \sqrt{\frac{T_0 + 273.15}{T + 273.15}}$$

Where:

- T_0 is the standard sea-level temperature (in Celsius).
- T is the temperature at the given altitude (in Celsius).

You can assume standard values for T_0 and the temperature lapse rate. After calculating TAS, display it to the user.

1.3.4 Mach Number Calculator

Design a program that calculates the Mach number of an aircraft based on its TAS and the speed of sound at the given altitude. The user should input TAS (in meters per second) and altitude (in meters). Use the following formula:

$$Mach = \frac{TAS}{\sqrt{\gamma.R.T}}$$

Where:

- γ is the specific heat ratio (assumed to be constant).
- R is the specific gas constant for dry air.

- T is the temperature in Kelvin at the given altitude.

Calculate the Mach number and display it to the user.

1.3.5 Range and Endurance Calculator

Create a program that calculates the range and endurance of an aircraft based on its fuel consumption rate and speed. The user should input fuel consumption rate (in liters per hour), speed (in knots), and the amount of fuel on board (in liters). Use the following formulas:

$$Range = \frac{\text{Fuel on board}}{\text{Fuel consumption rate}} \times Speed$$

and

$$Endurance = \frac{\text{Fuel on board}}{\text{Fuel consumption rate}}$$

After calculating the range and endurance, display both values to the user.

1.3.6 Glide Ratio Calculator

Create a program that calculates the glide ratio of an aircraft based on its altitude loss and horizontal distance traveled during a glide. The user should input the altitude loss (in meters) and the horizontal distance (in meters). Use the following formula:

$$\text{Glide Ratio} = \frac{HorizontalDistance}{AltitudeLoss}$$

After calculating the glide ratio, display it to the user.

1.3.7 Calculate time dilation

Calculate the time an astronaut experiences in space. The time dilation factor is calculated by

$$\text{time dilation factor} = \frac{1}{\sqrt{1 - (\frac{v}{c})^2}}$$

where v is the velocity of the astronaut vessel, c is the speed of light, $3 * 10^8 (3e8)$ m/s. The time experienced by an astronaut is given by

$$\text{time astronaut} = \text{time/time dilation factor}$$

all given in years. Given the velocity of a spaceship and a number of years, calculate the number of years experienced by the astronaut.

Examples: (Follow exactly the same message format)

```
Input velocity(m/s): 10
Input number of years: 7
The number of years experienced by the astronaut is: 6.999999999999996

Input velocity(m/s): 2.98e8
Input number of years: 5
The number of years experienced by the astronaut is: 0.5763872155263555

Input velocity(m/s): 2e6
Input number of years: 10
The number of years experienced by the astronaut is: 9.999777775308587
```

Chapter 2

The numpy, random

2.1 The numpy package

2.1.1 Constants

Write programs to

1. Print the constant π.
2. Write a program that calculates the area of a circle based on the radius provided by the user. Display the result with two decimal places.
3. Print the constant e

2.1.2 Trigonometric functions

Write programs to:

1. Convert a number from degrees to radians;
2. Convert radians to degrees;
3. Calculate the $cos()$, $sin()$ $tan()$ of a certain angle in degrees;
4. Calculate the arc length of a circle sector: $radius{*}angle\_radians$. Observe that the perimeter of a circle is a particular case of this function.
5. Convert Cartesian coordinates to polar coordinates: $(x, y) \rightarrow rcos(\theta)$, where $r = |||(x, y)|||$ and $\theta = arctan2(y, x)$ and comes in degrees.
6. Given the cos of a first angle in radians, the sin of an second angle in radians and the cos of the difference between these two angles, determine the sign of the first angle;
7. Calculate the area between two given points of the cos function. Remember that $\frac{d}{dx}sin(x) = cos(x)$.

2.1.3 Numerical functions

Write programs to

1. Calculate the logarithm in base 2 of a given number;
2. Calculate the logarithm in base 10 of a given number;
3. Calculate the exponential of a given number;
4. Calculate the exponential - 1 of a given number;
5. Calculate the greatest common divisor of two given numbers, n and m.
6. Calculate the least common multiple between two numbers, n and m
7. Calculate the factorial of a given number.

2.1.4 Rounding

Using the numpy package, write programs to:

1. Round a number to a specific number of decimals;
2. Round to the immediately lower integer;
3. Round the immediately upper integer;
4. Just takes off the decimal part of a number.

2.2 The random package

Devise programs to:

1. Simulate a coin flip;
2. Simulate a dice roll (possible values 1-6);
3. Simulate a euro millions number key;
4. Simulate a *Russian roulette* game.
5. Shuffle a collection of items, for instance, movies or music names.

2.3 All

1. Consider that the launch of space ship is conditioned by the **weather conditions** and the **mechanical conditions**, both subject to random oscillations. Implement a program that:

    - Reads an interval of two integers from the user, regarding the mechanical conditions;

- Generate a random **real** number using an uniform distribution from that **interval**. Divide the difference between the number obtained and the start of the interval, divide for the interval length and that will give you the **probability of mechanical failures**.
- Generate a random number for the wind speed (between 0 and 200).
- If the mechanical conditions are below 50% and the wind is between 20 and 80, authorize the launch, by displaying a message (Flight authorized: (True/False))) (**no if statements**).

2. Implement a **fortune cookie** app, that shows a random portuguese saying to the user.

3. Calculate the trajectory of a object after a time t considering that there maybe an obstacle in its way:

   - The object is just a point;
   - Consider that the object is at position (0,0) at time 0, and it there is no acceleration and the object moves at constant speed. The object is launched with an initial angle θ, between 0 and 90. **Both the speed and the initial angle are given by the user**.
   - The position formula reads as
     (a) $p_x = v * sin(\theta) * t$
     (b) $p_y = v * cos(\theta) * t$
   - The obstacle is another point.
   - Determine if there will be a collision and the exit angles for the object (180-θ).

4. Considere o seguinte programa para calcular a distância a que um projéctil, lançando com um ângulo inicial ”angulo” e uma velocidade v, fica quando cai no chão, a um alvo também ele pousado no chão:

```python
from numpy import sin, cos, deg2rad

# Funcao que calcula a distancia entre dois pontos
def distancia (x1, y1, x2, y2):
    a)

# Funcao que calcula a posicao no eixo x para um dado t.
def calcula_x_dado_t (v, t, theta):
    b)

# Funcao que calcula a posicao no eixo y para um dado t.
def calcula_y_dado_t (v, t, theta):
    return t*v*sin (theta) - (1/2)*g*(t**2)

# Funcao que calcula o t para a queda do objecto (y=0)
def resolve_t_para_y_igual_a_zero (v, g, theta):
    c) # (por exemplo, utilize a formula resolvente)

# Funcao que calcula o t para uma posicao x
def resolve_t_dado_x (v, theta, x):
    return x/(v*cos (theta))

g = 9.81 # Constante da gravidade
x_do_alvo = 100 # Posicao do alvo (pousado no chao)

# leitura dos dados
v = d)
angulo = d)

t = resolve_t_para_y_igual_a_zero (v, g, angulo)
x_queda = calcula_x_dado_t (v, t, angulo)
dist = distancia (x_queda, 0, x_do_alvo, 0)

print ("O projectil aterrara a ", round (dist, 2), "m do
    alvo!")
```

A equação da trajectória para o eixo x é (i.e. $\theta = angulo$):

$$x_t = v.cos(\theta).t; y_t = v.sin(\theta).t - \frac{1}{2}.g.t^2$$

Implemente:

(a) A função de distância (**0.25v**);

(b) A função calcula_x_dado_t (**0.25v**);

(c) A função resolve_t_para_y_igual_a_zero (**0.25v**)

(d) A leitura dos dados na forma: ”Escreva velocidade (m/s): ” e ”Escreva o angulo em **graus**: ” (**0.25v**).

Chapter 3

If statements

1. Implement a day of the week program: given a number 1 to 7, determine what is the correspondent day of the week;
2. Implement a small calculator: given two numbers and a possible operation: '+', '-', '*', '/', return the numbers and the operation;
3. Implement a 'Rock', 'Paper', 'Scissors' game: ask the user for his play, generate a random item from the collection, see who wins!
4. Implement a program to calculate the body-mass index, given user's height and weight in meters and kilograms respectively. Index is calculated as $weight/(height^2)$. The conclusions shall be as follows:
    - 18.5 to 24.9 - Healthy;
    - 25 to 29.9 - Overweight;
    - Over 30 - Obese.
5. Leap year checker. A leap is characterized by being divisible by four, but not 100 unless it is also divisible by 400.
6. Implement a triangle type checker that is able to, given three points is able to tell whether the triangle is scalene, isosceles or equilateral.
7. Implement a program to sort four numbers.

Chapter 4

Recursivity

1. Implement a program to calculate the factorial.
2. Implement a program to calculate the fibonacci function.
3. Implement a program to count the number of digits of a number, using a recursive function
   - You can also calculate the number of digits in the decimal base through:
   $$int(ceiling(log_{10}(number))).$$
4. Implement a program to count the number of times a specific digit appears in a given number.
5. Implement a program to return the most significant digit of a number.
6. Calculate the number of odd numbers, divisible by 3 between an interval, using a recursive function
7. Implement a program to count the number of leap years between two years, using recursive functions.
8. Implement a program to invert the digits of a number, in decimal, using a recursive function. (you can use the functions developed in 1)) [**HARD**]
9. Implement a program to calculate the greatest common divisor, using a recursive function. The recursive function can be implmented based in the following laws:
   (a) MDC (x, y) = MDC (x-y, y) se x>y;
   (b) MDC (x,y) = MDC (y,x)
   (c) MDC (x,x) = x
10. Implement a program to calculate the combinations function, which can be obtained recursively using the following relations:
    $$C(n,k) = C(n-1,k-1) + C(n-1,k)$$
    With the base cases:
    $$C(n,0) = 1 \quad \text{and} \quad C(n,n) = 1$$

Chapter 5

While and for instructions

Implement programs to:

1. calculate the factorial of a number (without using numpy);
2. calculate the n^{th} number of the fibonacci sequence;
3. generate and show a multiplication table asked by the user, or all of them.
4. implement a count down number;
5. play a guessing number game: generate a random number and give feedback to the user until it is guessed right. Give him the option to continue or quit and count the number of attempts it took to guess the right number.
6. Reverse the digits of a number.
7. **Collatz Sequence Length:** Given a number, apply the following rules until the number becomes 1:
   - If the number is even, divide it by 2.
   - If the number is odd, multiply it by 3 and add 1.
   - Count how many steps it takes to reach 1 and display the count.
8. **Prime Finder:** Write a program that asks the user for a number and determines if it's prime or not.
9. Calculate the constant e, which is given by the following formula:

$$e = \sum_{0 \leq k < \infty} \frac{1}{k!}$$

10. Make a program to calculate the function arctan, given by the following formula:

$$arctan(x) = \frac{x}{1+x^2} \sum_{n=0}^{\infty} \prod_{k=1}^{n} \frac{2kx^2}{(2k+1)(1+x^2)}$$

11. **Lunar Landing Simulation:** You're tasked with simulating a lunar lander's descent onto the Moon's surface. The spacecraft starts at an altitude h with a vertical speed v. The Moon's gravitational acceleration is $1.625 m/s^2$. Your lunar module has a throttle that can provide thrust to decelerate at varying rates between $0 m/s^2$ and $4 m/s^2$. The goal is to ensure a soft landing, meaning your final vertical speed should be less than $2 m/s$ when altitude is 0. Write a program that simulates this descent in time intervals of 1 second. At each interval, the program should prompt the user for the amount of thrust to apply (between 0 and 4). Display the altitude and velocity at each interval. The simulation ends when the lunar module lands.

Chapter 6

Tuples, Lists

6.1 Tuples

Write programs to:

1. Tuple creation: Create a tuple with elements of different data types and print the type of each element.
2. Tuple Indexing: Given a tuple, print its first and last element;
3. Check if an element exists in a tuple;
4. Count the existence of an element in a tuple;
5. Find the Outlier: In a tuple of numbers, there's one strange number (or vice versa). Find it and returns this number.

6.2 Lists

1. (Basic lists)
   (a) Create an empty list
   (b) Make a function that returns the head (first element) and the tail (the remaining elements of a list);
   (c) Concatenate three lists;
   (d) Split a list of elements in half, and exchange the first half to the end of the list and the second half to the beggining of the list, i.e. second half becomes first and the first one becomes second;
   (e) Implement a swap function to exchange two elements of a list.
2. (Standard list functions)

   Without using the **standard list functions** and trying to implement both recursive and non-recursive solutions, write programs to mimic:

   (a) The **len (list)** fuction, that counts the number of elements on a list;

(b) The **pop (element, list)** fuction (eliminates an element on given position of a list and returns the element);

(c) The **append (element, list)** function (adds an element at the end of the list);

(d) The **remove (element, list)** function (eliminates a specific element of the list);

(e) The **insert_at (element, list)** function (inserts an element at a specific position of the list) ;

(f) The **count (element, list)** function (counts the number of instances of the element at a list);

3. (Linear list iteration)

   (a) Given a list of numbers, sum the elements of the list and calculate the average and return both results;

   (b) Find the minimum and maximum element of a list;

   (c) Find the median of a list (if the length of the list is even, return the average of its central elements);

   (d) Given a list with numbers, create another list with just the even ones;

   (e) Reverse a list;

   (f) Determine if a list is a palindrome;

   (g) Instersect two lists. Given two lists l and h, implement a program to discover the common elements between them.

   (h) List sorting: sort the elements of a list in ascending order;

   (i) Triplicate the repeated elements of a list;

   (j) *Rotate* to the right every element of a list: the first one becomes second, the second becomes third, etc. Also, the last one becomes first.

   (k) Partition a list based on a pivot: return the list of elements left of the pivot and the list of elements right to the pivot. The list $[1, 2, 3, 4, 5]$ shall yield $[1, 2]$ and $[4, 5]$ for the pivot 3.

4. (Quadratic list iteration)

   Make programs (using recursivity or not) to:

   (a) Eliminate the repeated elements of a list, leaving just one instance of each element;

   (b) Sort a list without using the sort method, using the **Bubble sort** algorithm:

   - Starting in the first item of the list, swap with the next one if it is bigger. Do this for N items on the list
   - Decrease N and redo the previous step (every time the process is repeated the last element of previous iteration is left out of the process).

   (c) Eliminate the repeated elements of a list, leaving **at most two** instances of each element;

5. Complex list problems
   (a) Return the k-lowest elements of a list
   (b) Generate tuples with all the combinations with three elements of a list (modulo permutations);
   (c) Rotate elements of a list to the left k positions.
   (d) Give a list of tuples representing intervals, e.g. $[(1,3),(2,6),(8,10),(15,18)]$, return a list with the insersected intervals, e.g. $[(1,6),(8,10),(15,18)]$

6.3 Strings

Make programs to:

1. (About individual character processing):
   (a) Count the vowels in a string;
   (b) Duplicate the vowels in a string;
   (c) Count the number of spaces existent on a string;
   (d) Eliminate all numeric characters;
   (e) Switch the capitalization of individual characters.
2. (About words)
   (a) Count the number of words in a given text;
   (b) Calculate the most and less frequent word in a string;
   (c) Count the number of strings in a text, i.e. the words that do not contain only numbers;
   (d) Find the largest word in a text;
   (e) Print the list of words that contain a certain substring;
   (f) Print the list of words that start with a certain letter.
3. (About find and replace)
   (a) Replace a specific substring by another in a string;
   (b) Find all instances of a specific string in a text;
4. Write the translation of a number into its representation in the Roman numerical system (numbers until 2999)
   - I - 1; V - 5; X - 10; L - 50; C - 100; D - 500; M - 1000

Chapter 7

Sets and Dictionaries

7.1 Dictionaries

Implement programs to

1. Create a dictionary. Add, change and remove values for a given key.
2. Eliminate repeated elements from a list;
3. Count the number of pairs of elements in a given list;
4. manage the grades of a set of students, i.e. a dictionary of dictionaries, where the keys are students for the first dictionary, and values another dictionary with the grades.

   Implement actions to:

   - Calculate the average grades for a student;
   - Calculate the average grade on a given course.
5. A phonebook application. Use a dictionary, where keys are user names and values are phone numbers. Implement functions to add, remove and update users.
6. To count word frequency in a text.
7. Make username-password verification. The dictionary will hold a series of key value pairs, where keys are users and values are passwords.
8. Partially translate phrases in portuguese to english by the use of the following dictionary:

```python
portuguese_to_english = {
    "casa": "house",
    "gato": "cat",
    "cachorro": "dog",
    "banana": "banana",
    "amigo": "friend",
    "livro": "book",
    "computador": "computer",
    "sol": "sun",
    "lua": "moon"
}
```

7.2 Sets

Implement programs to:

1. Given two lists, for example $[12, 13, 14, 14, 15, 16, 16]$, and $[12, 13, 14]$, make the following actions (use sets):
   - Discover the elements that belong to both lists;
   - Calculate the elements that are not common to both lists;
   - Determine whether a set is superset or subset of another one.
2. Obtain the **repeated** elements of a list;
3. Given a large set of elements, build disjoint sets for each of equivalence classes of those elements defined by the modulo operation. Example, two elements, x, y are of the same class of equivalence if $x\%n == y\%n$.
4. "The friend of my friend, my friend is". Given a dictionary, where keys and individuals and values are lists of friends, identify communities, disjoint sets of friends of friends.