CC - AI

# Programación para la Computación Científica - IA

# Python for Programmers

## Numpy

Universidad Sergio Arboleda

*Prof. John Corredor*

```python
import random

def display_intro():
    title = "** A Simple Math Quiz **"
    print("*" * len(title))
    print(title)
    print("*" * len(title))


def display_menu():
    menu_list = ["1. Addition", "2. Subtraction", "3. Multiplication", "4. Integer Division", "5. Exit"]
    print(menu_list[0])
    print(menu_list[1])
    print(menu_list[2])
    print(menu_list[3])
    print(menu_list[4])

def display_separator():
    print("-" * 24)
```

# Challenge 11

```
def menu_option(index, count):
    number_one = random.randrange(1, 21)
    number_two = random.randrange(1, 21)
    if index is 1:
        problem = str(number_one) + " + " + str(number_two)
        solution = number_one + number_two
        user_solution = get_user_solution(problem)
        count = check_solution(user_solution, solution, count)
        return count
    elif index is 2:
        problem = str(number_one) + " - " + str(number_two)
        solution = number_one - number_two
        user_solution = get_user_solution(problem)
        count = check_solution(user_solution, solution, count)
        return count
    elif index is 3:
        problem = str(number_one) + " * " + str(number_two)
        solution = number_one * number_two
        user_solution = get_user_solution(problem)
        count = check_solution(user_solution, solution, count)
        return count
```

# Challenge 11

```
def get_user_input():
    user_input = int(input("Enter your choice: "))
    while user_input > 5 or user_input <= 0:
        print("Invalid menu option.")
        user_input = int(input("Please try again: "))
    else:
        return user_input

def get_user_solution(problem):
    print("Enter your answer")
    print(problem, end="")
    result = int(input(" = "))
    return result

def check_solution(user_solution, solution, count):
    if user_solution == solution:
        count = count + 1
        print("Correct.")
        return count
    else:
        print("Incorrect.")
        return count
```

```
  else:
    problem = str(number_one) + " // " + str(number_two)
    solution = number_one // number_two
    user_solution = get_user_solution(problem)
    count = check_solution(user_solution, solution, count)
    return count
def display_result(total, correct):
  if total > 0:
    result = correct / total
    percentage = round((result * 100), 2)
  if total == 0:
    percentage = 0
  print("You answered", total, "questions with", correct, "correct.")
  print("Your score is ", percentage, "%. Thank you.", sep = "")
```

e 11

nge 11

```
def main():
    display_intro()
    display_menu()
    display_separator()

    option = get_user_input()
    total = 0
    correct = 0
    while option != 5:
        total = total + 1
        correct = menu_option(option, correct)
        option = get_user_input()

    print("Exit the quiz.")
    display_separator()
    display_result(total, correct)
main()
```

# Numpy Library

Para usar una biblioteca científica compilada, la memoria asignada en el intérprete de Python debe llegar de alguna manera a esta biblioteca como entrada.

Además, la salida de estas bibliotecas también debe volver al intérprete de Python. Este intercambio bidireccional de memoria es esencialmente la función central del módulo Numpy (matrices numéricas en Python). Numpy es el estándar de facto para matrices numéricas en Python.

Surgió como un esfuerzo de Travis Oliphant y otros para unificar las matrices numéricas en Python.

```
>>> import numpy as np # recommended convention
>>> x = np.array([1,2,3],dtype=np.float32)
>>> x
array([ 1., 2., 3.], dtype=float32)
>>> x.itemsize
4
```

```
>>np.sin(np.array([1,2,3],dtype=np.float32) )
array([ 0.84147096, 0.90929741, 0.14112 ], dtype=float32)
```

This computes the sine of the input array [1,2,3], using Numpy's unary function, np.sin.

```
>>> from math import sin
>>> [sin(i) for i in [1,2,3]] # list comprehension
[0.8414709848078965, 0.9092974268256817, 0.1411200080598672]
```

```
>>> x=np.array([ [1,2,3],[4,5,6] ])
>>> x.shape
(2, 3)
```

Numpy arrays come in many dimensions. Note that Numpy is limited to 32 dimensions unless you build it for more.

```
>>> x=np.array([ [1,2,3],[4,5,6] ])
>>> x[:,0] # 0th column
array([1, 4])
>>> x[:,1] # 1st column
array([2, 5])
>>> x[0,:] # 0th row
array([1, 2, 3])
>>> x[1,:] # 1st row
array([4, 5,6])
```

the : colon character selects all elements along a particular axis.

If the indexing object (i.e., the item between the brackets) is

- a non-tuple sequenceobject,
- another Numpy array (of type integer or boolean),
- or a tuple with at least one sequence object or Numpy array,

then indexing creates copies.

Because of advanced indexing, the variable y has its own memory because the relevant parts of x were copied.

```
>>> x = np.ones((3,3))
>>> x
array([[ 1., 1., 1.],
       [ 1., 1., 1.],
       [ 1., 1., 1.]])
>>> x[:,[0,1,2,2]] # notice duplicated last dimension
array([[ 1., 1., 1., 1.],
       [ 1., 1., 1., 1.],
       [ 1., 1., 1., 1.]])
>>> y=x[:,[0,1,2,2]]
```

```
>>> x[0,0]=999 # change element in x
>>> x                  # changed
array([[ 999., 1., 1.],
       [    1., 1., 1.],
       [    1., 1., 1.]]
>>> y                  # not changed!
array([[ 1., 1., 1., 1.],
       [ 1., 1., 1., 1.],
       [ 1., 1., 1., 1.]])
```

To prove it, we assign a new element to x and see that y is not updated.

```
>>> x = np.ones((3,3))
>>> y = x[:2,:2] # view of upper left piece
>>> x[0,0] = 999 # change value
>>> x
array([[ 999., 1.,  1.], # see the change?
       [    1., 1.,  1.],
       [    1., 1., 1.]])
>>> y
array([[ 999., 1.], # changed Y also!
       [    1., 1.]])
```

Sin embargo, si comenzamos de nuevo y construimos y cortamos (lo que lo convierte en una vista) como se muestra a continuación, entonces el cambio que realizamos afecta: “**una vista es solo una ventana a la misma memoria**”.

```
>>> from numpy.lib.stride_tricks import as_strided
>>> x = arange(16)
>>> y=as_strided ( x, (7,4), (8,4) ) # overlapped entries
>>> y
```

La manipulación de la memoria utilizando vistas es particularmente poderosa para los algoritmos de procesamiento de señales e imágenes que requieren fragmentos de memoria superpuestos. Ejemplo de cómo usar Numpy avanzado para crear bloques superpuestos que en realidad no consumen memoria adicional, Tenga en cuenta que as_strided no verifica que se mantenga dentro de los límites del bloque de memoria. Entonces, si el tamaño de la matriz de destino no se llena con los datos disponibles, **los elementos restantes provendrán de los bytes que estén en esa ubicación de memoria.** En otras palabras, no hay relleno predeterminado por ceros u otra estrategia que defienda los límites de bloque de memoria. Una defensa es controlar explícitamente las dimensiones:

```
>>> n = 8 # number of elements
>>> x = arange(n) # create array
>>> k = 5 # desired number of rows
>>> y = as_strided(x,(k,n-k+1),(x.itemsize,)*2)
>>> y
```

# Challenge 01

- 1.- Escriba un programa NumPy para redondear elementos de la matriz al entero más cercano.

Entrada: [-0.7 8.2 0.3 4.2 1.8 2.]
Salida:   [-1. 8. 0. 4 2. 2.]

- 2.- Programa para calcular el peso de una persona

Prueba
Masa = 60 kilos
Peso = 588 Newtons

# Numpy Matrices

Matrices in Numpy are similar to Numpy arrays but they can only have two dimensions. They implement **row-column matrix multiplication** as opposed to element-wise multiplication (np.array). If you have two matrices you want to multiply, you can either create them directly or convert them from Numpy arrays.

```
>>> import numpy as np
>>> A=np.matrix([[1,2,3],[4,5,6],[7,8,9]])
>>> x=np.matrix([[1],[0],[0]])
>>> A*x
```

```
>>> A=np.array([[1,2,3],[4,5,6],[7,8,9]])
>>> x=np.array([[1],[0],[0]])
>>> A.dot(x)
```

# Numpy Matrices (II)

No es necesario echar todo a las matrices para multiplicarlo. En el siguiente ejemplo, todo hasta la última línea es ***numpy array*** Y, a continuación, convertimos el **array** como una matriz con np.matrix que luego utiliza la multiplicación fila-columna.

```
>>> A=np.ones((3,3))
>>> type(A)                # array not matrix
<type 'numpy.ndarray'>
>>> x=np.ones((3,1))       # array not matrix
>>> A*x
>>> np.matrix(A)*x         # row-column multiplication
```

# Numpy Matrices (III)
## Access

```
import numpy as np
A = np.array([2, 4, 6, 8, 10])
print("A[0] =",  A[0])     # First element
print("A[2] =",  A[2])     # Third element
print("A[-1] =", A[-1])    # Last element
```

# Numpy Matrices (IV) Access

```python
import numpy as np
A = np.array([[1, 4, 5, 12],
              [-5, 8, 9, 0],
              [-6, 7, 11, 19]])
#  First element of first row
print("A[0][0] =", A[0][0])
# Third element of second row
print("A[1][2] =", A[1][2])
# Last element of last row
print("A[-1][-1] =", A[-1][-1])
```

## Numpy Matrices (V)
## Access

```
import numpy as np
A = np.array([[1, 4, 5, 12],
              [-5, 8, 9, 0],
              [-6, 7, 11, 19]])
#  First row
print("A[0]  =", A[0])
# Third row
print("A[2]  =", A[2])
# Last row
print("A[-1] =", A[-1])
```

# Numpy Matrices (VI)
## Slicing

```
import numpy as np
A = np.array([[1, 4, 5, 12, 14],
    [-5, 8, 9, 0, 17],
    [-6, 7, 11, 19, 21]])
# two rows, four columns
print(A[:2, :4])
# first row, all columns
print(A[:1,])
 # all rows, second column
print(A[:,2])
# all rows, third to the fifth column
print(A[:, 2:5])
```

```
************************
**    Matriz Calculator    **
************************
1. Addition
2. Subtraction
3. Multiplication
4. Inverse
5. Exit
------------------------
Enter your choice: 1
```

## Challenge 11

# References


- ★ Kernighan, Brian W., and Dennis M. Ritchie. The C Programming Language. Vol. 2. Englewood Cliffs: prentice-Hall, 1988.
- ★ Silberschatz, Abraham, Peter B. Galvin, and Greg Gagne. Operating System Concepts. Vol. 8. Wiley, 2013.
- ★ https://planningtank.com/computer-applications/data-processing-cycle
- ★ https://www.talend.com/resources/what-is-data-processing/
- ★ https://peda.net/kenya/ass/subjects2/computer-studies/form-3/data-processing/dpc2
- ★ https://www.academia.edu/38210518/What_is_Data_Processing
- ★ https://www.studocu.com/en/document/polytechnic-university-of-the-philippines/information-and-communication-technology/lecture-notes/data-processing-lectures-in-data-processing/3167716/view
- ★ http://download.nos.org/srsec330/330L2.pdf