Εφαρμογές Γραμμικής Άλγεβρας σε python με multiprocessing

ΟΜΑΔΙΚΗ ΕΡΓΑΣΙΑ 1ΟΥ ΕΞΑΜΗΝΟΥ

Βιβλιοθήκες που χρησιμοποιήθηκαν

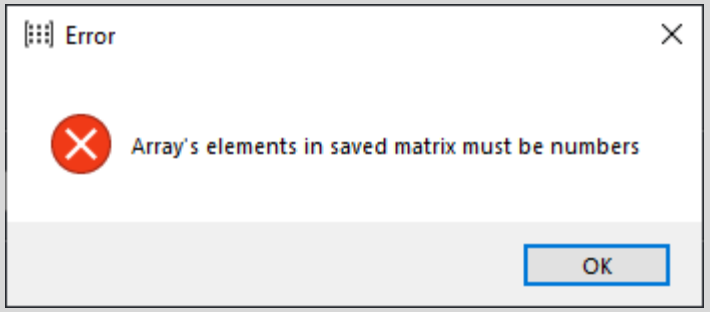
Python Multiprocessing

Βιβλιοθήκη με γραφικές διεπαφές

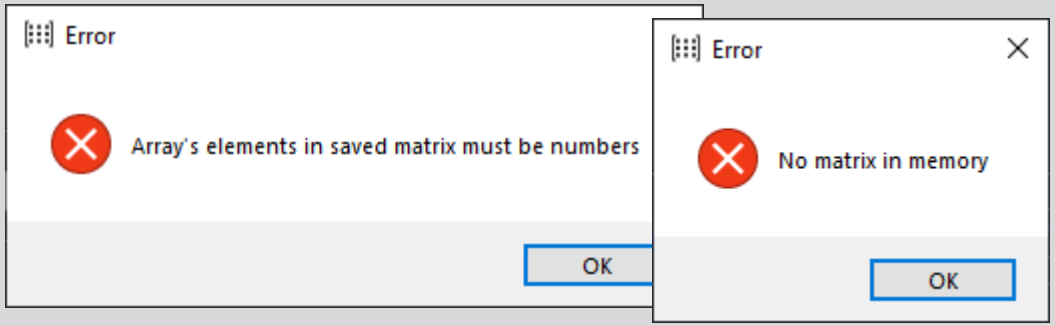
Βιβλιοθήκη σε Python και C, που περιέχει όλες τις πράξεις πινάκων

Αμυντικός προγραμματισμός

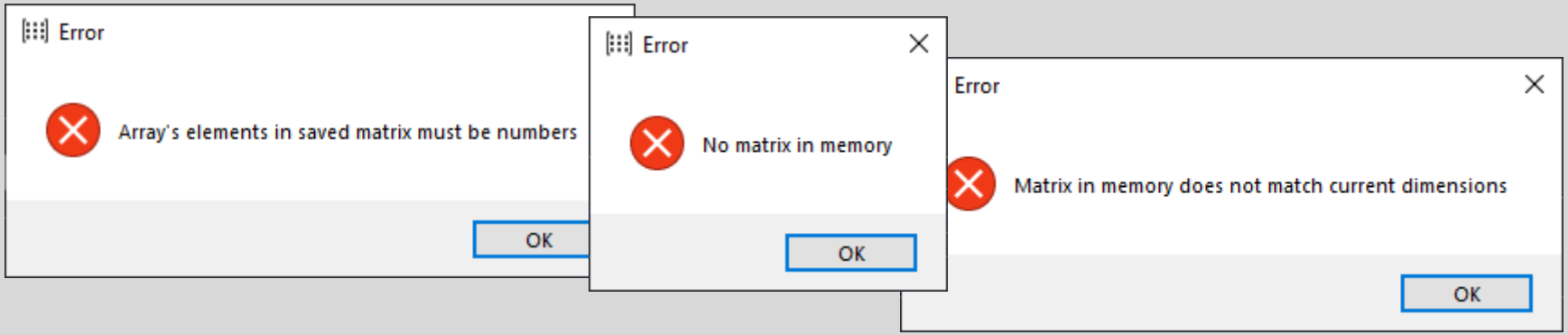
Αμυντικός προγραμματισμός



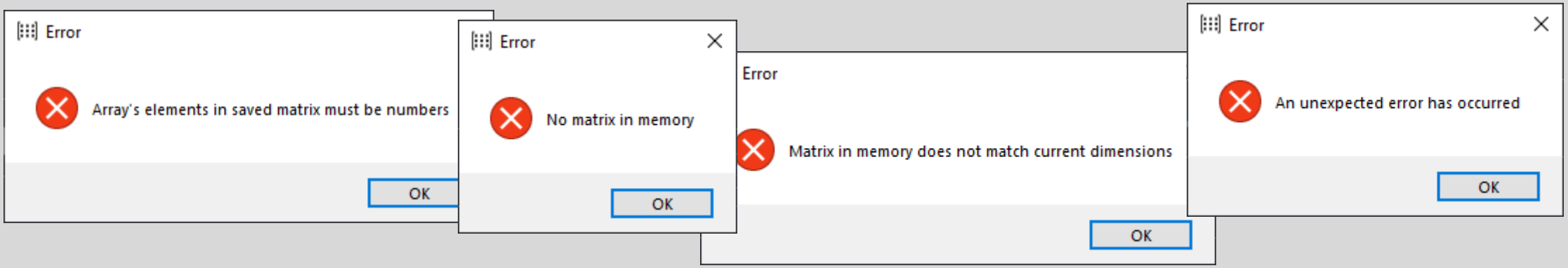
Αμυντικός προγραμματισμός



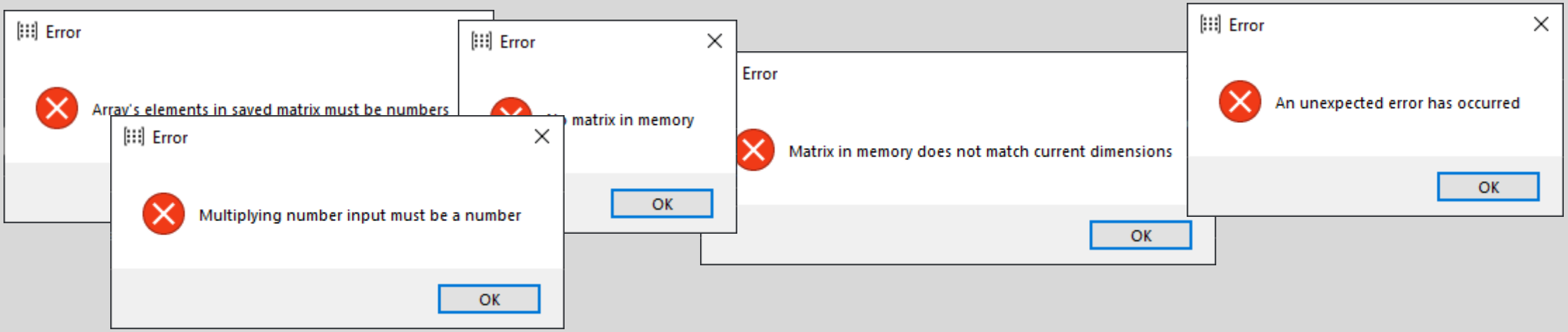
Αμυντικός προγραμματισμός



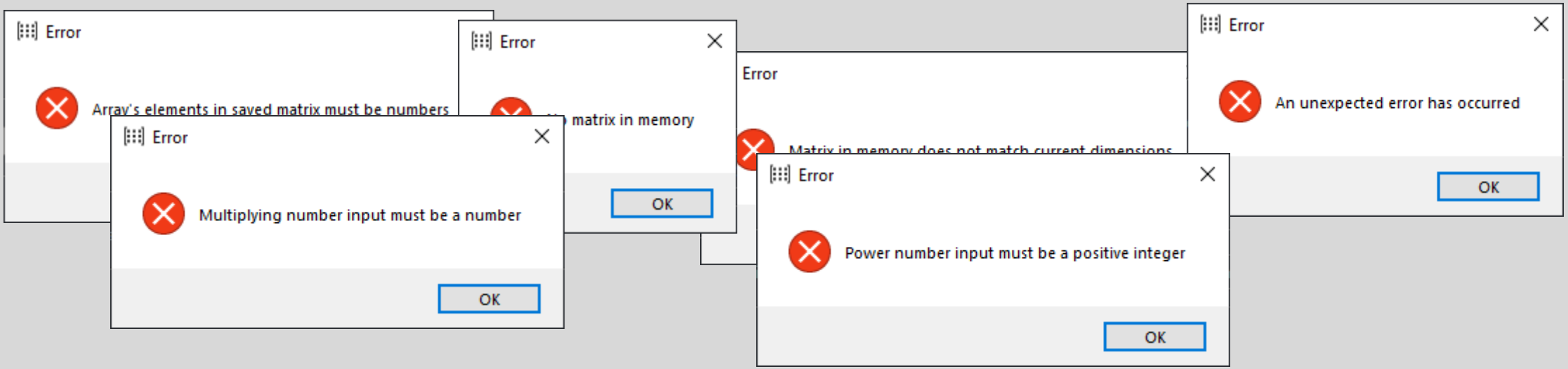
Αμυντικός προγραμματισμός



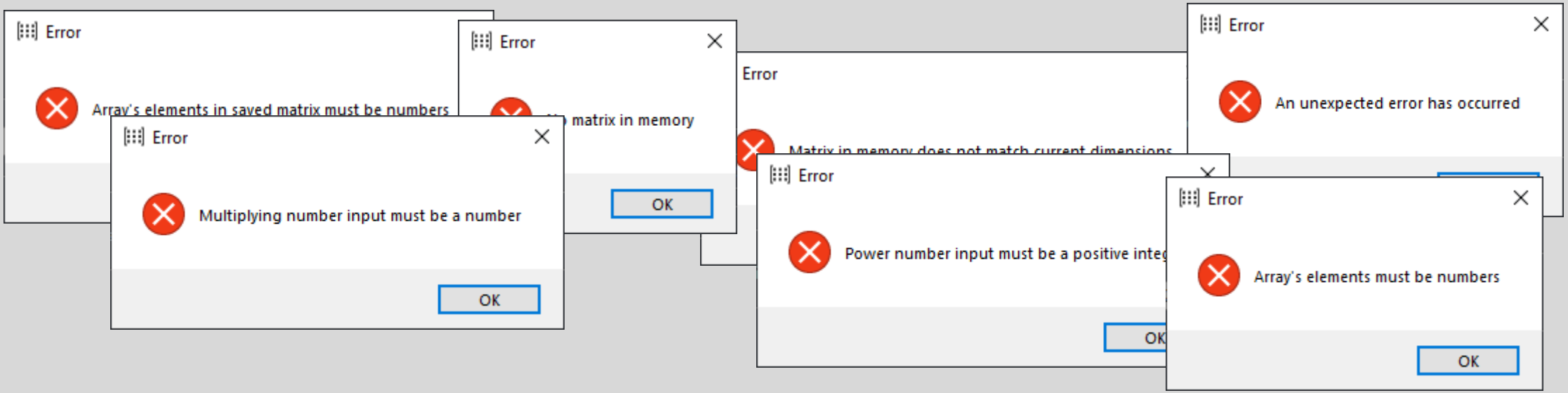
Αμυντικός προγραμματισμός



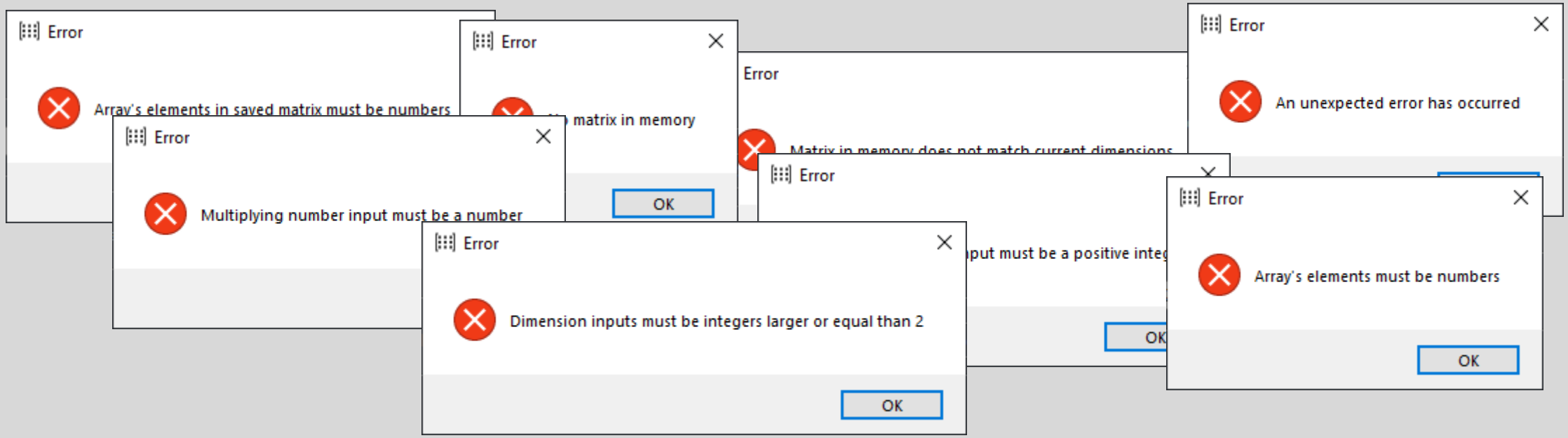
Αμυντικός προγραμματισμός



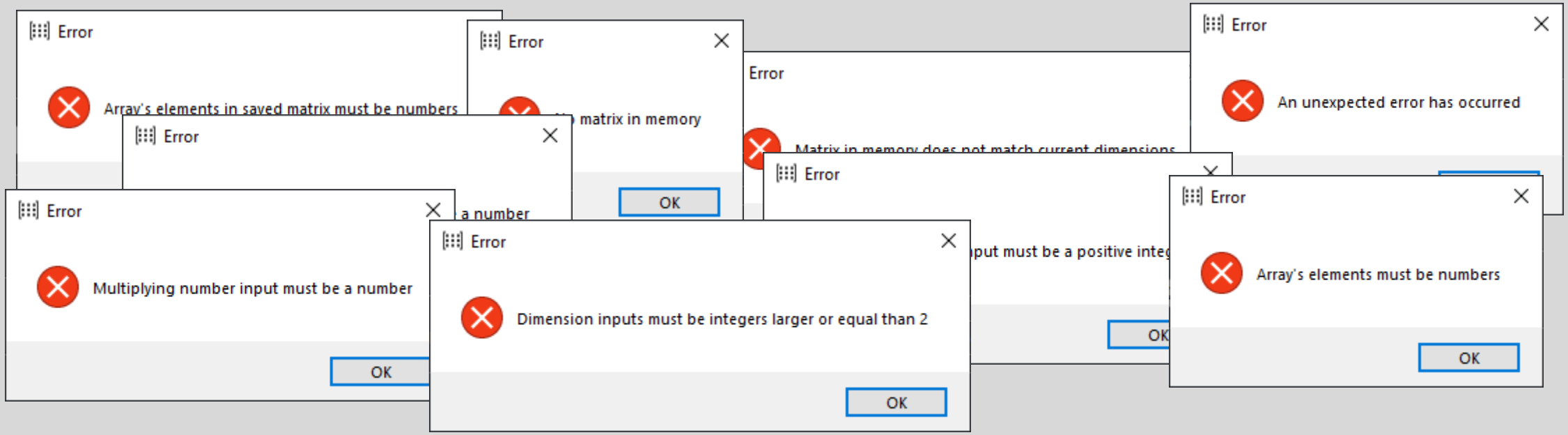
Αμυντικός προγραμματισμός



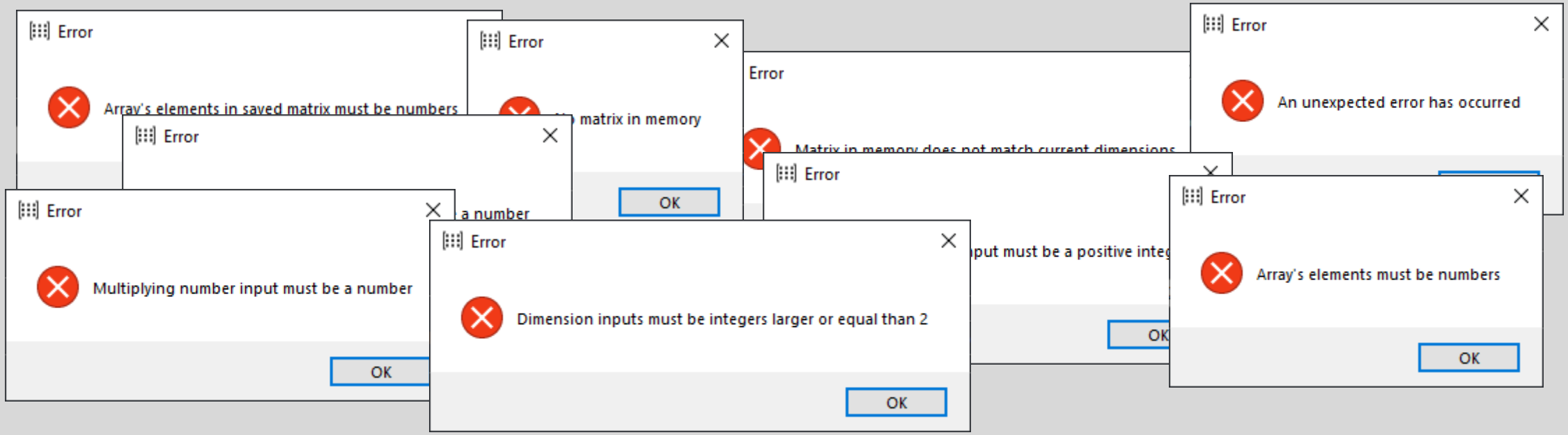
Αμυντικός προγραμματισμός



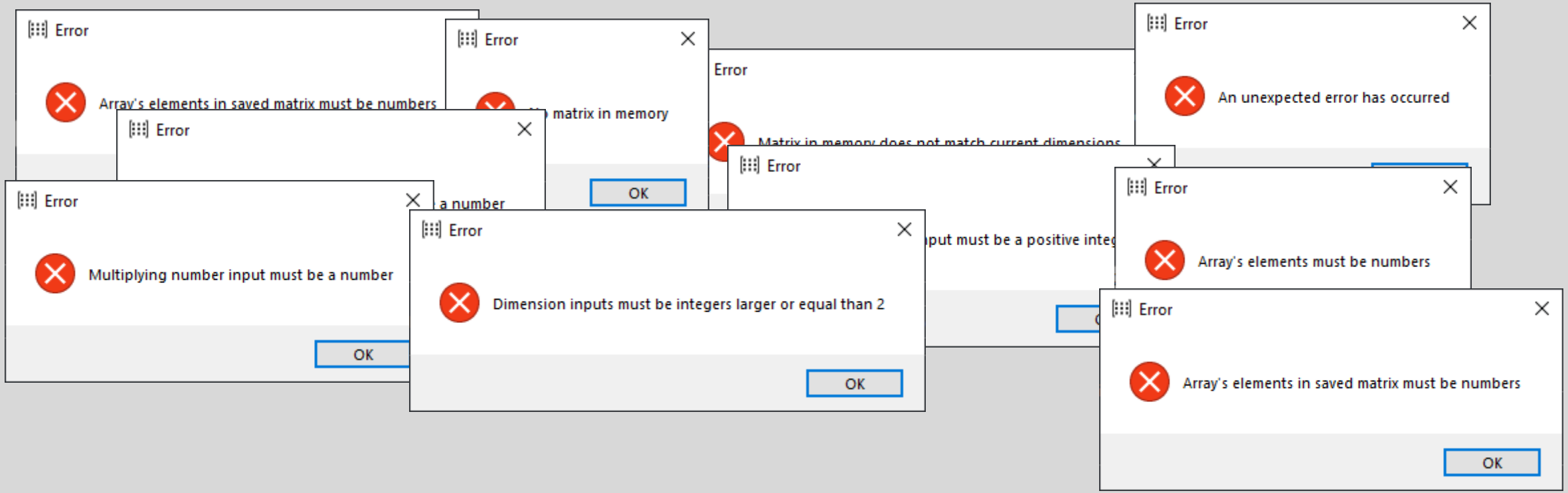
Αμυντικός προγραμματισμός



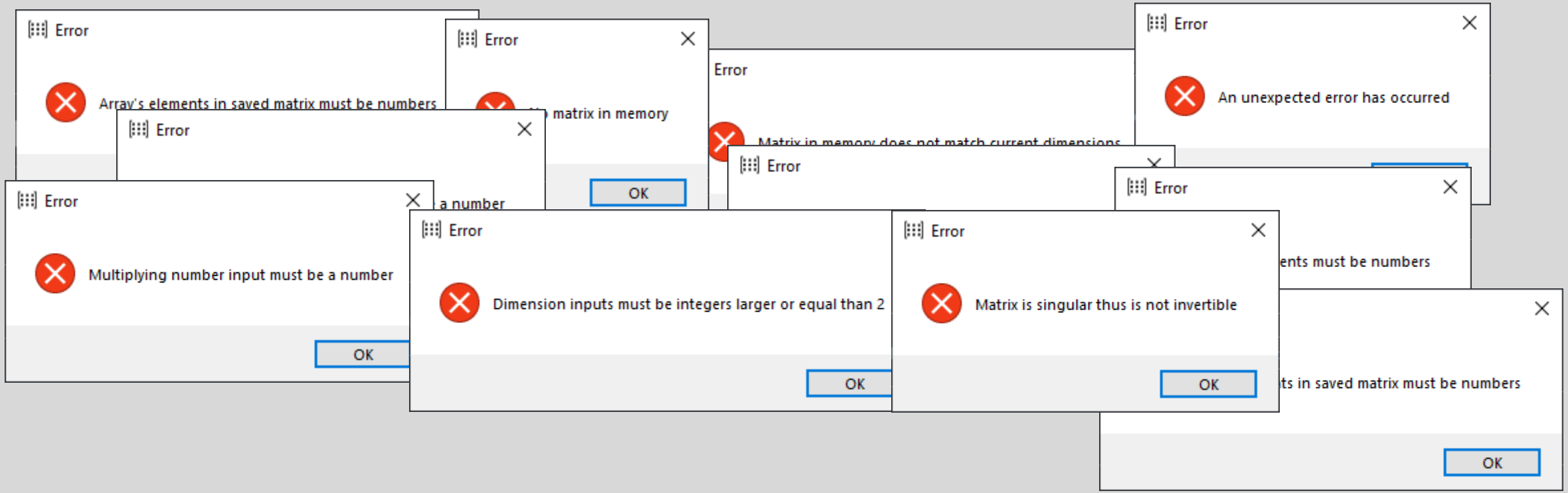
Αμυντικός προγραμματισμός



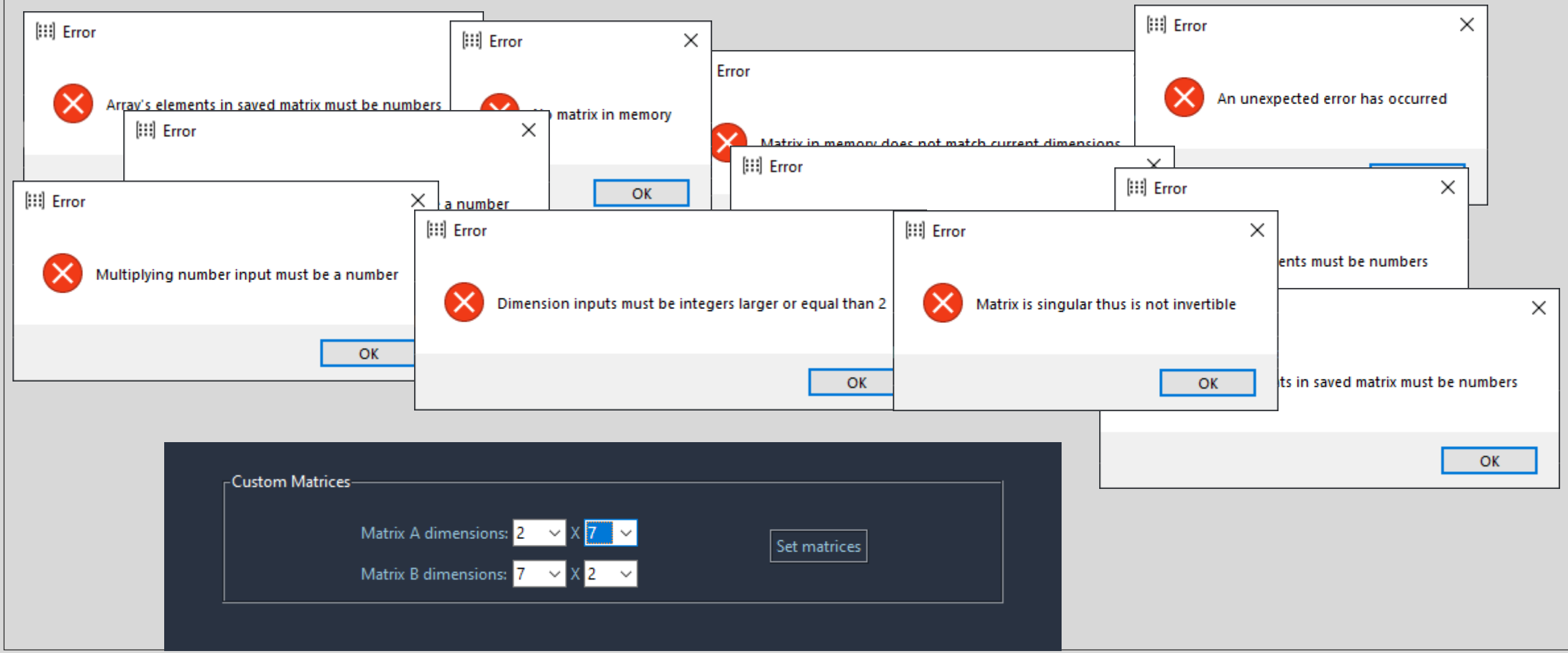
Αμυντικός προγραμματισμός



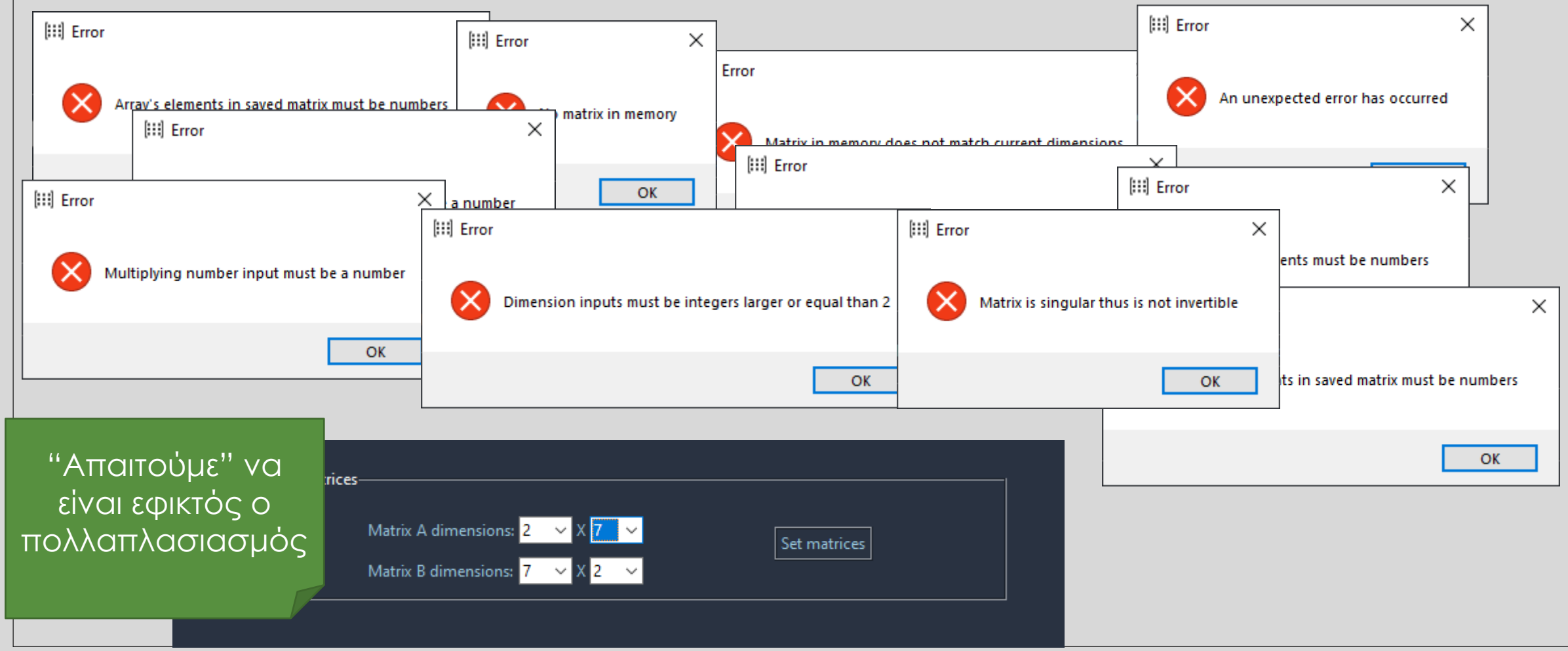
Αμυντικός προγραμματισμός



Αμυντικός προγραμματισμός

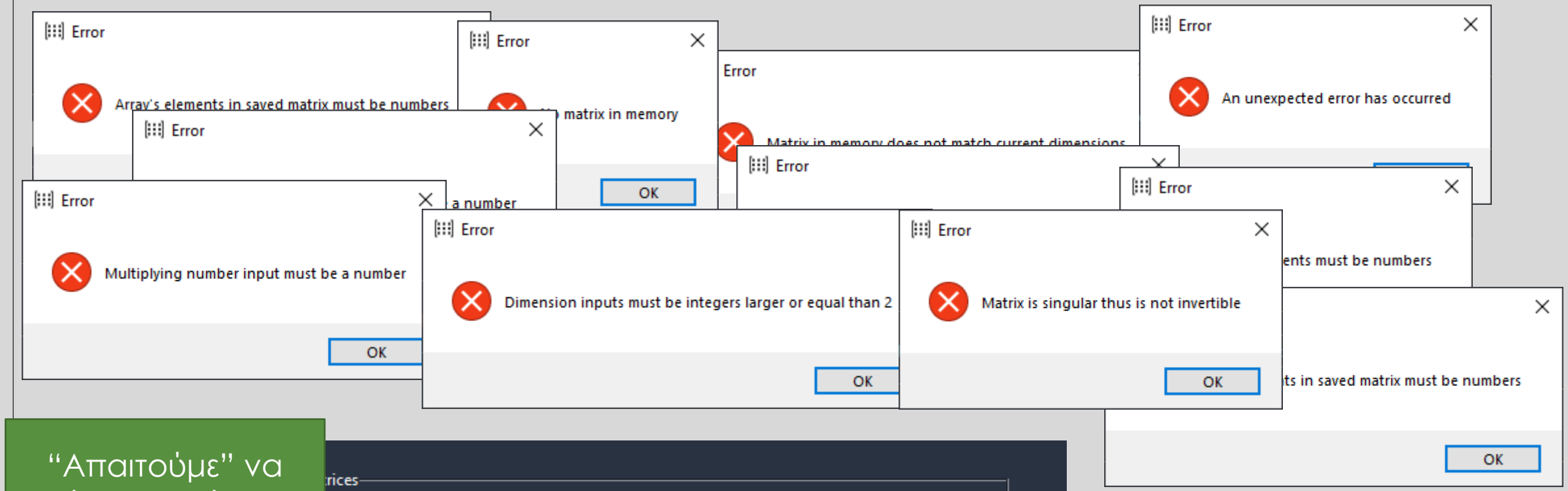


Αμυντικός προγραμματισμός

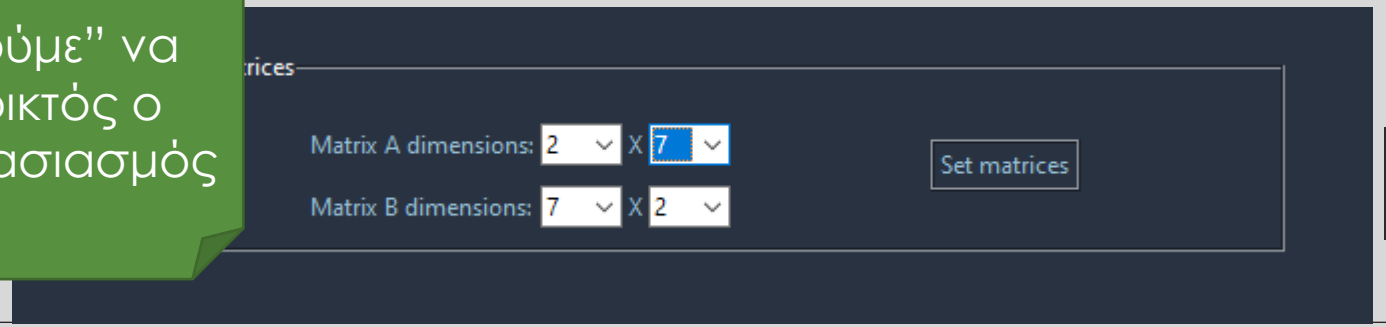


"Απαιτούμε" να είναι εφικτός ο πολλαπλασιασμός

Αμυντικός προγραμματισμός



"Απαιτούμε" να είναι εφικτός ο πολλαπλασιασμός




```
def callback(iv):
    iv.set(iv.get())
```

Κλάσεις του κώδικα

Κλάσεις του κώδικα

Κλάσεις του κώδικα

RandomMatrix

- Δημιουργία πίνακα προκαθορισμένων διαστάσεων και τυχαίων στοιχείων

SimpleCalculation

- Βασικές συναρτήσεις πράξεων
- Χρήση της NumPy

Multiprocessing Calculation

- Χρήση της Multiprocessing

Κλάση GUI

Όλες οι γραφικές διεπαφές

```
class GUI:
    def __init__(self, root):...

    def add_sub(self):...

    def mul_num(self):...

    def mul(self):...

    def power(self):...

    def det(self):...

    def inv(self):...

    def trans(self):...

    def rank(self):...

    def trace(self):...

    def clear_frame(self):...

    def set_matrix(self, func, a, b=0, c=0):...

    def calculate(self, func, a, b=0, c=0):...

    def rand_calculate(self, func, a, b=0, c=0):...

    def clear_cells(self, list):...

    def fill_zeros(self, list):...

    def fill_ones(self, list):...

    def mem_sv(self, list):...

    def mem_ld(self, list):...
```

Κλάση GUI

Όλες οι γραφικές διεπαφές

```
class GUI:
    def __init__(self, root):...

    def add_sub(self):...

    def mul_num(self):...

    def mul(self):...

    def power(self):...

    def det(self):...

    def inv(self):...

    def trans(self):...

    def rank(self):...

    def trace(self):...

    def clear_frame(self):...

    def set_matrix(self, func, a, b=0, c=0):...

    def calculate(self, func, a, b=0, c=0):...

    def rand_calculate(self, func, a, b=0, c=0):...

    def clear_cells(self, list):...

    def fill_zeros(self, list):...

    def fill_ones(self, list):...

    def mem_sv(self, list):...

    def mem_ld(self, list):...
```

Κλάση GUI

Όλες οι γραφικές διεπαφές

- Δημιουργία εκάστοτε παραθύρου

```
class GUI:
    def __init__(self, root):...

    def add_sub(self):...

    def mul_num(self):...

    def mul(self):...

    def power(self):...

    def det(self):...

    def inv(self):...

    def trans(self):...

    def rank(self):...

    def trace(self):...

    def clear_frame(self):...

    def set_matrix(self, func, a, b=0, c=0):...

    def calculate(self, func, a, b=0, c=0):...

    def rand_calculate(self, func, a, b=0, c=0):...

    def clear_cells(self, list):...

    def fill_zeros(self, list):...

    def fill_ones(self, list):...

    def mem_sv(self, list):...

    def mem_ld(self, list):...
```

Κλάση GUI

Όλες οι γραφικές διεπαφές

- Δημιουργία εκάστοτε παραθύρου

```
class GUI:
    def __init__(self, root):...

    def mul(self):
        GUI.clear_frame(self)

        self.title = Label(self.f_main, text='Matrix Multiplication Calculator', font=('Arial', 30, 'bold'),
                           bg=self.color_bg1, fg=self.color_text1)
        self.title.pack(pady=(30, 0))

        self.desc = Label(self.f_main, text='''Matrix Multiplication is a binary operation that produces
a matrix from two matrices. For matrix multiplication,
the number of columns in the first matrix must be equal
to the number of rows in the second matrix. The resulting matrix,
known as the matrix product, has the number of rows
of the first and the number of columns of the second matrix.''', font=('Arial', 15), bg=self.color_bg1,
                          fg=self.color_text1)
        self.desc.pack(pady=(30, 0))

        self.f_dims = LabelFrame(self.f_main, text='Custom Matrices', padx=100, pady=10, bg=self.color_bg1,
                                 fg=self.color_text3, relief=GROOVE)
        self.f_dims.pack(pady=(200, 50))

        self.dimA_text = Label(self.f_dims, text='Matrix A dimensions:', bg=self.color_bg1, fg=self.color_text2)
        self.dimA_text.grid(row=0, column=0, pady=10)
        self.dimB_text = Label(self.f_dims, text='Matrix B dimensions:', bg=self.color_bg1, fg=self.color_text2)
        self.dimB_text.grid(row=1, column=0)

    def fill_zeros(self, list):...

    def fill_ones(self, list):...

    def mem_sv(self, list):...

    def mem_ld(self, list):...
```

Κλάση GUI

Όλες οι γραφικές διεπαφές

- Δημιουργία εκάστοτε παραθύρου

```
class GUI:
    def __init__(self, root):...

    def mul(self):
        GUI.clear_frame(self)

        self.title = Label(self.f_main, text='Matrix Multiplication Calculator', font=('Arial', 30, 'bold'),
                           bg=self.color_bg1, fg=self.color_text1)
        self.title.pack(pady=(30, 0))

        self.desc = Label(self.f_main, text='''Matrix Multiplication is a binary operation that produces
a matrix from two matrices. For matrix multiplication,
the number of columns in the first matrix must be equal
to the number of rows in the second matrix. The resulting matrix,
known as the matrix product, has the number of rows
of the first and the number of columns of the second matrix.''', font=('Arial', 15), bg=self.color_bg1,
                          fg=self.color_text1)
        self.desc.pack(pady=(30, 0))

        self.f_dims = LabelFrame(self.f_main, text='Custom Matrices', padx=100, pady=10, bg=self.color_bg1,
                                 fg=self.color_text3, relief=GROOVE)
        self.f_dims.pack(pady=(200, 50))

        self.dimA_text = Label(self.f_dims, text='Matrix A dimensions:', bg=self.color_bg1, fg=self.color_text2)
        self.dimA_text.grid(row=0, column=0, pady=10)
        self.dimB_text = Label(self.f_dims, text='Matrix B dimensions:', bg=self.color_bg1, fg=self.color_text2)
        self.dimB_text.grid(row=1, column=0)

    def fill_zeros(self, list):...

    def fill_ones(self, list):...

    def mem_sv(self, list):...

    def mem_ld(self, list):...
```

Κλάση GUI

Όλες οι γραφικές διεπαφές

- Δημιουργία εκάστοτε παραθύρου
- Δημιουργία Πίνακα & κλήση συνάρτησης

```
elif func == 'mul':
    self.matrix_A = np.zeros((a, b))
    self.matrix_B = np.zeros((b, c))

    try:
        for i in range(a):
            for j in range(b):
                if self.matrix_A_entries[i][j].get() == '':
                    self.matrix_A_entries[i][j].insert(0, '0')

                self.matrix_A[i, j] = float(self.matrix_A_entries[i][j].get())
        for i in range(b):
            for j in range(c):
                if self.matrix_B_entries[i][j].get() == '':
                    self.matrix_B_entries[i][j].insert(0, '0')

                self.matrix_B[i, j] = float(self.matrix_B_entries[i][j].get())
    except ValueError:
        is_error = True
        self.errors('alpha')
    except:
        is_error = True
        self.errors('unexpected')

    if not is_error:
        print("Matrix A:", self.matrix_A, sep="\n")
        print()
        print("Matrix B:", self.matrix_B, sep="\n")
        print()

        start = time.perf_counter()
        calc = SimpleCalculation.matrix_mul(self.matrix_A, self.matrix_B)
        finish = time.perf_counter()

        self.time = round(finish - start, 3)
        print("Result:", calc, sep="\n")
        print()
        print("Time:", self.time)
        print()
```

```
font=('Arial', 30, 'bold'),

operation that produces

5), bg=self.color_bg1,

ady=10, bg=self.color_bg1,

color_bg1, fg=self.color_text2)

color_bg1, fg=self.color_text2)
```

Κλάση GUI

Όλες οι γραφικές διεπαφές

- Δημιουργία εκάστοτε παραθύρου
- Δημιουργία Πίνακα & κλήση συνάρτησης

Συνάρτηση calculate

```
elif func == 'mul':
    self.matrix_A = np.zeros((a, b))
    self.matrix_B = np.zeros((b, c))

    try:
        for i in range(a):
            for j in range(b):
                if self.matrix_A_entries[i][j].get() == '':
                    self.matrix_A_entries[i][j].insert(0, '0')

                self.matrix_A[i, j] = float(self.matrix_A_entries[i][j].get())
        for i in range(b):
            for j in range(c):
                if self.matrix_B_entries[i][j].get() == '':
                    self.matrix_B_entries[i][j].insert(0, '0')

                self.matrix_B[i, j] = float(self.matrix_B_entries[i][j].get())
    except ValueError:
        is_error = True
        self.errors('alpha')
    except:
        is_error = True
        self.errors('unexpected')

    if not is_error:
        print("Matrix A:", self.matrix_A, sep="\n")
        print()
        print("Matrix B:", self.matrix_B, sep="\n")
        print()

        start = time.perf_counter()
        calc = SimpleCalculation.matrix_mul(self.matrix_A, self.matrix_B)
        finish = time.perf_counter()

        self.time = round(finish - start, 3)
        print("Result:", calc, sep="\n")
        print()
        print("Time:", self.time)
        print()
```

```
-('Arial', 30, 'bold'),

operation that produces

5), bg=self.color_bg1,

ady=10, bg=self.color_bg1,

color_bg1, fg=self.color_text2)

color_bg1, fg=self.color_text2)
```

Κλάση GUI

Όλες οι γραφικές διεπαφές

- Δημιουργία εκάστοτε παραθύρου
- Δημιουργία Πίνακα & κλήση συνάρτησης

Συνάρτηση calculate

```
elif func == 'mul':
    self.matrix_A = np.zeros((a, b))
    self.matrix_B = np.zeros((b, c))

    try:
        for i in range(a):
            for j in range(b):
                if self.matrix_A_entries[i][j].get() == '':
                    self.matrix_A_entries[i][j].insert(0, '0')

                self.matrix_A[i, j] = float(self.matrix_A_entries[i][j].get())
        for i in range(b):
            for j in range(c):
                if self.matrix_B_entries[i][j].get() == '':
                    self.matrix_B_entries[i][j].insert(0, '0')

                self.matrix_B[i, j] = float(self.matrix_B_entries[i][j].get())
    except ValueError:
        is_error = True
        self.errors('alpha')
    except:
        is_error = True
        self.errors('unexpected')

    if not is_error:
        print("Matrix A:", self.matrix_A, sep="\n")
        print()
        print("Matrix B:", self.matrix_B, sep="\n")
        print()

        start = time.perf_counter()
        calc = SimpleCalculation.matrix_mul(self.matrix_A, self.matrix_B)
        finish = time.perf_counter()

        self.time = round(finish - start, 3)
        print("Result:", calc, sep="\n")
        print()
        print("Time:", self.time)
        print()
```

```
-('Arial', 30, 'bold'),

operation that produces

5), bg=self.color_bg1,

ady=10, bg=self.color_bg1,

color_bg1, fg=self.color_text2)

color_bg1, fg=self.color_text2)
```

Κλάση GUI

Όλες οι γραφικές διεπαφές

- Δημιουργία εκάστοτε παραθύρου
- Δημιουργία Πίνακα & κλήση συνάρτησης
- Εμφάνιση Αποτελέσματος

Συνάρτηση calculate

```
elif func == 'mul':
    self.matrix_A = np.zeros((a, b))
    self.matrix_B = np.zeros((b, c))

    try:
        for i in range(a):
            for j in range(b):
                if self.matrix_A_entries[i][j].get() == '':
                    self.matrix_A_entries[i][j].insert(0, '0')

                self.matrix_A[i, j] = float(self.matrix_A_entries[i][j].get())
        for i in range(b):
            for j in range(c):
                if self.matrix_B_entries[i][j].get() == '':
                    self.matrix_B_entries[i][j].insert(0, '0')

                self.matrix_B[i, j] = float(self.matrix_B_entries[i][j].get())
    except ValueError:
        is_error = True
        self.errors('alpha')
    except:
        is_error = True
        self.errors('unexpected')

    if not is_error:
        print("Matrix A:", self.matrix_A, sep="\n")
        print()
        print("Matrix B:", self.matrix_B, sep="\n")
        print()

        start = time.perf_counter()
        calc = SimpleCalculation.matrix_mul(self.matrix_A, self.matrix_B)
        finish = time.perf_counter()

        self.time = round(finish - start, 3)
        print("Result:", calc, sep="\n")
        print()
        print("Time:", self.time)
        print()
```

```
-('Arial', 30, 'bold'),

operation that produces

5), bg=self.color_bg1,

ady=10, bg=self.color_bg1,

color_bg1, fg=self.color_text2)

color_bg1, fg=self.color_text2)
```

Κλάση GUI

Όλες οι γραφικές διεπαφές

- Δημιουργία εκάστοτε παραθύρου
- Δημιουργία Πίνακα & κλήση συνάρτησης
- Εμφάνιση Αποτελέσματος

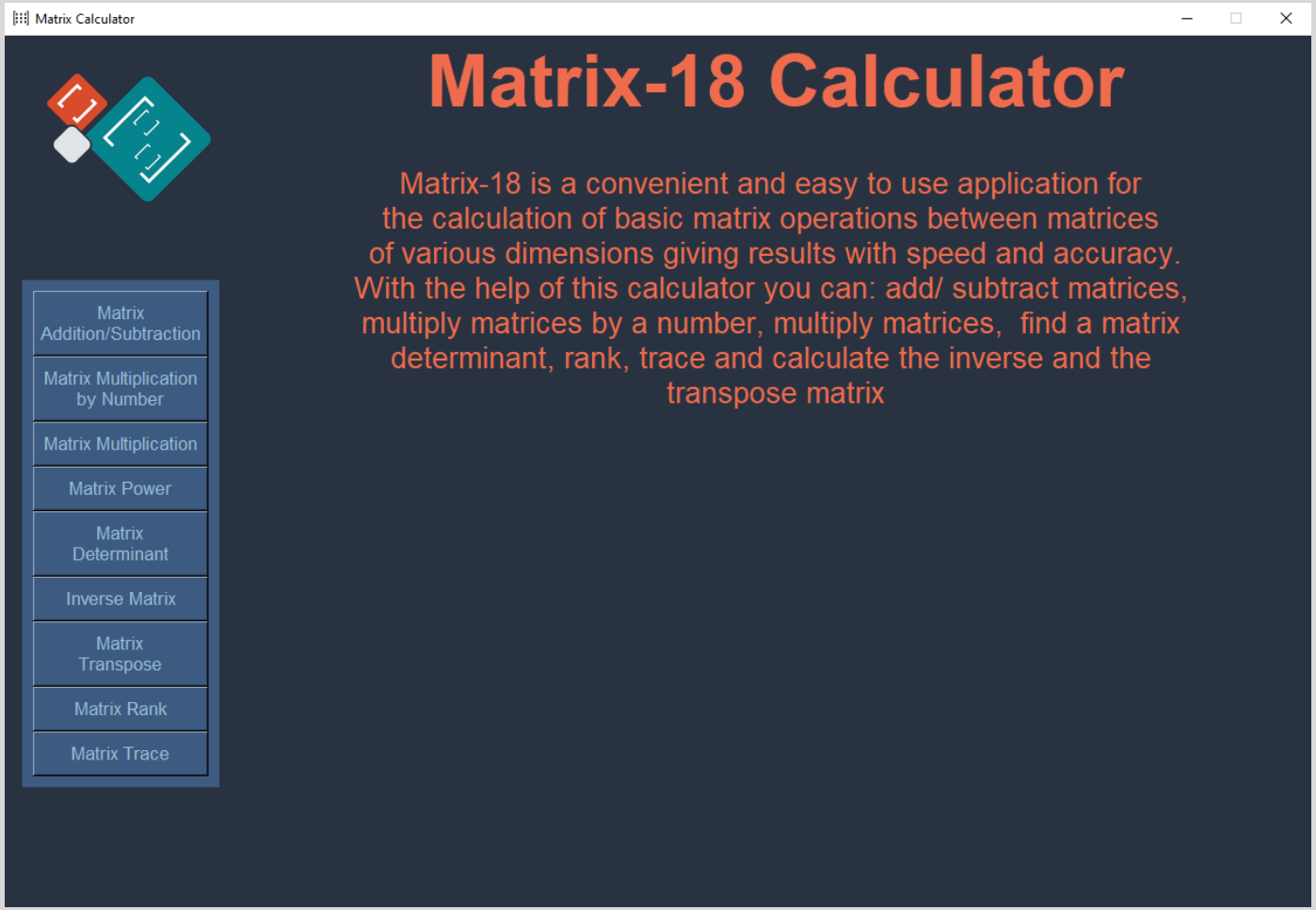
```
  elif func == 'mul':
def result(self, result, func=''):
    """This method creates a new window displaying the result"""
    self.result_win = Toplevel(root, bg=self.color_bg1)
    self.result_win.iconbitmap('matrix_ico.ico')
    self.result_win.title('Matrix Calculator')
    self.result_win.geometry("600x400")

    self.f_res_text = Frame(self.result_win, bg=self.color_bg1)
    self.f_res_text.pack(side='top', fill='x', padx=40, pady=(40, 20))

    self.f_main_res = Frame(self.result_win, bg=self.color_bg1, padx=5, pady=5)
    self.f_main_res.pack(side='top', padx=40)

    self.f_time = Frame(self.result_win, bg=self.color_bg1)
    self.f_time.pack(side='bottom', fill='x', padx=40, pady=(10, 40))

    self.time_text = Label(self.f_time, text=f'Computation time: {self.time} seconds', font=('Arial', 10),
                           bg=self.color_bg1, fg=self.color_text2)
    self.time_text.pack(anchor='e')

    self.result_text = Label(self.f_res_text, text='Result:', font=('Arial', 12), bg=self.color_bg1,
                             fg=self.color_text2)
    self.result_text.pack(anchor='w')

    labels = []
    width = 3

    if func == 'det':
        self.result_win.geometry("600x300")

        self.result = Label(self.f_main_res, text=f'Matrix determinant is {round(result, 2)}', font=('Arial', 15),
                            bg=self.color_bg1, fg=self.color_text2)
        self.result.pack(anchor='n')

         print()
```

Κλάση GUI

Όλες οι γραφικές διεπαφές

- Δημιουργία εκάστοτε παραθύρου
- Δημιουργία Πίνακα & κλήση συνάρτησης
- Εμφάνιση Αποτελέσματος

Το πρόγραμμά μας



Διευκολύνσεις

Διευκολύνσεις

@staticmethod

Διευκολύνσεις

@staticmethod

```
@staticmethod
def two_random_matrices(a, b=0, c=0):...
```

Διευκολύνσεις

@staticmethod

```
@staticmethod
def two_random_matrices(a, b=0, c=0):...
```

Διευκολύνσεις

@staticmethod

```
@staticmethod
def two_random_matrices(a, b=0, c=0):...
```

Με τη χρήση του επιτυγχάνεται η σύμπτυξη συναρτήσεων που δεν πραγματεύονται με παραμέτρους της κλάσης (self)

Διευκολύνσεις

@staticmethod

```
@staticmethod
def two_random_matrices(a, b=0, c=0):...
```

lambda

Με τη χρήση του επιτυγχάνεται η σύμπτυξη συναρτήσεων που δεν πραγματεύονται με παραμέτρους της κλάσης (self)

Διευκολύνσεις

@staticmethod

```
@staticmethod
def two_random_matrices(a, b=0, c=0):...
```

lambda

Με τη χρήση του επιτυγχάνεται η σύμπτυξη συναρτήσεων που δεν πραγματεύονται με παραμέτρους της κλάσης (self)

Διευκολύνσεις

@staticmethod

```
@staticmethod
def two_random_matrices(a, b=0, c=0):...
```

lambda

```
error = lambda: mb.showerror(title='Error',
                             message="Array's element can be only a number")
```

Με τη χρήση του επιτυγχάνεται η σύμπτυξη συναρτήσεων που δεν πραγματεύονται με παραμέτρους της κλάσης (self)

Διευκολύνσεις

@staticmethod

```
@staticmethod
def two_random_matrices(a, b=0, c=0):...
```

lambda

```
error = lambda: mb.showerror(title='Error',
                             message="Array's element can be only a number")
command=lambda: GUI.rand_calculate(self, 'det', self.rand_dim.get()))
```

Με τη χρήση του επιτυγχάνεται η σύμπτυξη συναρτήσεων που δεν πραγματεύονται με παραμέτρους της κλάσης (self)

Διευκολύνσεις

@staticmethod

```
@staticmethod
def two_random_matrices(a, b=0, c=0):...
```

lambda

```
error = lambda: mb.showerror(title='Error',
                             message="Array's element can be only a number")
command=lambda: GUI.rand_calculate(self, 'det', self.rand_dim.get()))
```

Με τη χρήση του επιτυγχάνεται η σύμπτυξη συναρτήσεων που δεν πραγματεύονται με παραμέτρους της κλάσης (self)

Διευκολύνσεις

@staticmethod

```
@staticmethod
def two_random_matrices(a, b=0, c=0):...
```

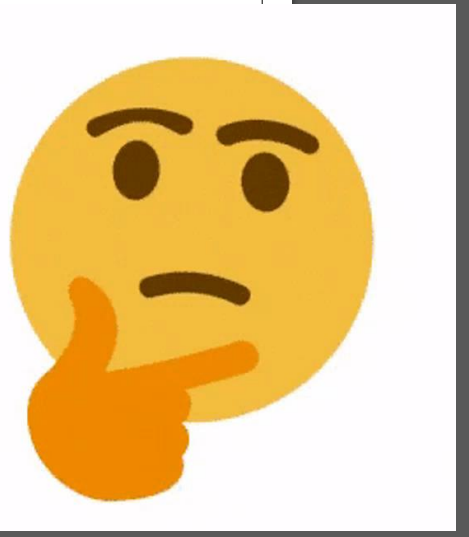
Με τη χρήση του επιτυγχάνεται η σύμπτυξη συναρτήσεων που δεν πραγματεύονται με παραμέτρους της κλάσης (self)

lambda

```
error = lambda: mb.showerror(title='Error',
                             message="Array's element can be only a number")
command=lambda: GUI.rand_calculate(self, 'det', self.rand_dim.get()))
```

Συντακτικά ελαφρύτερος προγραμματισμός
Ταχύτητα

Ερωτήσεις



Μια εργασία των:

Γιακουμέλου Αιμιλία με ΑΜ: up1083878

Ντάγκας Αλέξανδρος με ΑΜ: up1083874

Ντεν- Μπαρμπερ Βερνάρδος- Ανριανός με ΑΜ: up1083808

Παπουτσάς Γεώργιος με ΑΜ: up1083738

Ροδόπουλος Γεώργιος με ΑΜ: up1083876

Ψημμένος Επαμεινώνδας με ΑΜ: up1083815