

ΜΥΕ046 – Υπολογιστική Όραση: Άνοιξη 2023

1η Σειρά Ασκήσεων: 25% του συνολικού βαθμού

Διδάσκων: Άγγελος Γιώτης

- ΠΑΡΑΔΟΣΗ: Δευτέρα, 24 Απριλίου, 2023 23:59

Γενικές Οδηγίες

Απαντήστε στα παρακάτω ζητήματα χρησιμοποιώντας Python στο συνημμένο σημειωματάριο Jupyter και ακολουθήστε τις παρακάτω οδηγίες:

- Οι ασκήσεις είναι **ατομικές** - δεν επιτρέπεται η μεταξύ σας συνεργασία για την υλοποίηση/παράδοσή τους.
- Δεν επιτρέπεται να χρησιμοποιήσετε κώδικα που τυχόν θα βρείτε στο web. Η χρήση κώδικα τρίτων θα έχει σαν αποτέλεσμα τον αυτόματο μηδενισμό σας.
- Όλες οι λύσεις πρέπει να είναι γραμμένες σε αυτό το σημειωματάριο Jupyter notebook .
- Ο κώδικάς σας πρέπει να σχολιαστεί εκτενώς.
- Αφού ολοκληρώσετε (υλοποιήσετε και εκτελέσετε) τις απαντήσεις σας στο σημειωματάριο (notebook), εξαγάγετε το notebook ως PDF και υποβάλετε, τόσο το σημειωματάριο όσο και το PDF (δηλαδή τα αρχεία .ipynb και .pdf) στο turnin του μαθήματος, μαζί με ένα συνοδευτικό αρχείο onoma.txt που θα περιέχει το ον/μο σας και τον Α.Μ. σας.
- Οι απαντήσεις θα παραδοθούν με την εντολή: **turnin assignment_1@mye046 onoma.txt assignment1.ipynb assignment1.pdf**
- Μπορείτε να χρησιμοποιήσετε βασικά πακέτα γραμμικής άλγεβρας (π.χ. NumPy , SciPy κ.λπ.), αλλά δεν επιτρέπεται να χρησιμοποιείτε τα πακέτα/βιβλιοθήκες που επιλύουν άμεσα τα προβλήματα. Μη διστάσετε να ρωτήσετε τον διδάσκοντα εάν δεν είστε σίγουροι για τα πακέτα που θα χρησιμοποιήσετε.
- Συνιστάται ιδιαίτερα να αρχίσετε να εργάζεστε στις ασκήσεις σας το συντομότερο δυνατό!

Late Policy: Εργασίες που υποβάλλονται καθυστερημένα θα λαμβάνουν μείωση βαθμού 10% για κάθε 24 ώρες καθυστέρησης. Οι εργασίες δεν θα γίνονται δεκτές 72 ώρες (3 ημέρες) μετά την προθεσμία παράδοσης. Για παράδειγμα, παράδοση της εργασίας 2 ημέρες μετά την προθεσμία βαθμολογείται με άριστα το 20 (από 25).

Εισαγωγή

Καλώς ήρθατε στο μάθημα **ΜΥΕ046 – Υπολογιστική Όραση!**

Το μάθημα αυτό περιλαμβάνει μια ολοκληρωμένη εισαγωγή στην όραση υπολογιστών παρέχοντας ευρεία κάλυψη, συμπεριλαμβανομένης της υπολογιστικής όρασης χαμηλού επιπέδου (σχηματισμός εικόνας, φωτομετρία, χρώμα, ανίχνευση χαρακτηριστικών εικόνας - επεξεργασία εικόνας), συμπερασματολογία 3D ιδιοτήτων από εικόνες (σχήμα-από-σκιαση, στερεοσκοπία, ερμηνεία κίνησης) και αναγνώριση αντικειμένων (object recognition).

Θα χρησιμοποιήσουμε μια ποικιλία εργαλείων (π.χ. ορισμένα πακέτα και λειτουργίες) σε αυτό το μάθημα που μπορεί να απαιτούν κάποια αρχική παραμετροποίηση. Για να διασφαλίσουμε την ομαλή πρόοδο από εργασία σε εργασία, θα εγκαταστήσουμε τα περισσότερα από τα εργαλεία που θα χρησιμοποιηθούν σε αυτό το μάθημα σε αυτήν την σειρά ασκήσεων (**assignment1**). Θα εξασκηθείτε επίσης σε ορισμένες βασικές τεχνικές χειρισμού εικόνας.

Google Colab, Jupyter Notebook, JupyterLab and Python

ecourse

Οι ανακοινώσεις των σειρών ασκήσεων θα γίνονται στη σελίδα ecourse του μαθήματος. Κάθε σειρά ασκήσεων θα περιλαμβάνει το αρχείο .ipynb για προβολή και επεξεργασία σε περιβάλλον Jupyter Notebook ή JupyterLab, **είτε τοπικά** (local machine) στον υπολογιστή σας, **είτε μέσω της υπηρεσίας** νέφους [Google Colab](https://colab.research.google.com/) (<https://colab.research.google.com/>) ή [Colaboratory](https://colaboratory.google.com/) (<https://colaboratory.google.com/>).

Working remotely on Google Colaboratory

To [Google Colaboratory](https://colaboratory.google.com/) (<https://colaboratory.google.com/>) είναι βασικά ένας συνδυασμός σημειωματαρίου Jupyter και [Google Drive](https://www.google.com/drive/) (<https://www.google.com/drive/>). Εκτελείται εξ ολοκλήρου στο cloud και έρχεται προεγκατεστημένο με πολλά πακέτα (π.χ. PyTorch και Tensorflow), ώστε όλοι να έχουν πρόσβαση στις ίδιες εξαρτήσεις/βιβλιοθήκες. Ακόμη πιο ενδιαφέρον είναι το γεγονός ότι το Colab επωφελείται από την ελεύθερη πρόσβαση σε επιταχυντές υλικού (π.χ. κάρτες γραφικών) όπως οι GPU (K80, P100) και οι TPU που μπορεί να είναι ιδιαίτερα χρήσιμοι για τις σειρές ασκήσεων 2 και 3.

- Requirements:

Για να χρησιμοποιήσετε το Colab, πρέπει να έχετε λογαριασμό Google με συσχετισμένο Google Drive. Υποθέτοντας ότι έχετε και τα δύο (ο ακαδημαϊκός σας λογαριασμός είναι λογαριασμός google), μπορείτε να συνδέσετε το Colab στο Drive σας με τα ακόλουθα βήματα:

- Κάντε κλικ στον τροχό στην επάνω δεξιά γωνία (στο Google Drive) και επιλέξτε Ρυθμίσεις .
- Κάντε κλικ στην καρτέλα Διαχείριση εφαρμογών .
- Στο επάνω μέρος, επιλέξτε Σύνδεση περισσότερων εφαρμογών που θα εμφανίσουν ένα παράθυρο του GSuite Marketplace .
- Αναζητήστε το Colab και, στη συνέχεια, κάντε κλικ στην Προσθήκη (install).

- Workflow:

Κόθε σειρά ασκήσεων στη σελίδα ecourse του μαθήματος παρέχει έναν σύνδεσμο λήψης σε ένα αρχείο zip που περιέχει σημειωματάρια Colab, κώδικα Python και ενδεχομένως (αναλόγως τις απαιτήσεις της σειράς ασκήσεων) κάποιο φάκελο images με τις εικόνες που θα χρησιμοποιήσετε για την υλοποίησή σας. Μπορείτε να ανεβάσετε τον (αποσυμπιεσμένο) φάκελο στο Drive, να ανοίξετε τα σημειωματάρια στο Colab και να εργαστείτε πάνω τους, και στη συνέχεια, να αποθηκεύσετε την πρόοδό σας πίσω στο Drive.

- Βέλτιστες πρακτικές:

Υπάρχουν μερικά πρόγραμμα που πρέπει να γνωρίζετε όταν εργάζεστε με την υπηρεσία Colab. Το πρώτο πρόγραμμα που πρέπει να σημειωθεί είναι ότι οι πόροι δεν είναι εγγυημένοι (αυτό είναι το τίμημα της δωρεάν χρήσης). Εάν είστε σε αδράνεια για ένα συγκεκριμένο χρονικό διάστημα ή ο συνολικός χρόνος σύνδεσής σας υπερβαίνει τον μέγιστο επιτρεπόμενο χρόνο (~12 ώρες), το Colab VM θα αποσυνδεθεί. Αυτό σημαίνει ότι οποιαδήποτε μη αποθηκευμένη πρόοδος θα χαθεί. Έτσι, φροντίστε να αποθηκεύετε συχνά την υλοποίησή σας ενώ εργάζεστε.

- Χρήση GPU:

Η χρήση μιας GPU απαιτεί πολύ απλά την αλλαγή του τύπου εκτέλεσης (runtime) στο Colab. Συγκεκριμένα, κάντε κλικ Runtime -> Change runtime type -> Hardware Accelerator -> GPU και το στιγμιότυπο εκτέλεσής σας Colab θα υποστηρίζεται αυτόματα από επιταχυντή υπολογισμών GPU (αλλαγή τύπου χρόνου εκτέλεσης σε GPU ή TPU).

Working locally on your machine

- Σε περιβάλλον linux

Εάν θέλετε να εργαστείτε τοπικά στον Η/Υ σας, θα πρέπει να χρησιμοποιήσετε ένα εικονικό περιβάλλον. Μπορείτε να εγκαταστήσετε ένα μέσω του [Anaconda](https://www.anaconda.com/products/distribution) (<https://www.anaconda.com/products/distribution>) (συνιστάται) ή μέσω της native μονάδας venv της Python. Βεβαιωθείτε ότι χρησιμοποιείτε (τουλάχιστον) έκδοση Python 3.7.

- Εικονικό περιβάλλον Anaconda: Συνιστάται η χρήση της δωρεάν διανομής [Anaconda](https://www.anaconda.com/products/distribution) (<https://www.anaconda.com/products/distribution>), η οποία παρέχει έναν εύκολο τρόπο για να χειρίστετε τις εξαρτήσεις πακέτων. Μόλις εγκαταστήσετε το Anaconda, είναι εύχρηστο να δημιουργήσετε ένα εικονικό περιβάλλον για το μάθημα. Για να ρυθμίσετε ένα εικονικό περιβάλλον που ονομάζεται π.χ. mye046, εκτελέστε τα εξής στο τερματικό σας: conda create -n mye046 python=3.7 (Αυτή η εντολή θα δημιουργήσει το περιβάλλον mye046 στη διαδρομή 'path/to/anaconda3/envs/') Για να ενεργοποιήσετε και να εισέλθετε στο περιβάλλον, εκτελέστε το conda activate mye046. Για να απενεργοποιήσετε το περιβάλλον, είτε εκτελέστε conda deactivate mye046 είτε βγείτε από το τερματικό. Σημειώστε ότι κάθε φορά που θέλετε να εργαστείτε στην εργασία, θα πρέπει να εκτελείτε ξανά το conda activate mye046 .
- Εικονικό περιβάλλον Python venv: Για να ρυθμίσετε ένα εικονικό περιβάλλον που ονομάζεται mye046, εκτελέστε τα εξής στο τερματικό σας: python3.7 -m venv ~/mye046 Για να ενεργοποιήσετε και να εισέλθετε στο περιβάλλον, εκτελέστε το source ~/mye046 /bin/activate . Για να απενεργοποιήσετε το περιβάλλον, εκτελέστε: deactivate ή έξοδο από το τερματικό. Σημειώστε ότι κάθε φορά που θέλετε να εργαστείτε για την άσκηση, θα πρέπει να εκτελείτε ξανά το source ~/mye046/bin/activate .
- Installing packages: Αν και στην 1η σειρά ασκήσεων **δεν θα χρειαστεί** να εγκαταστήσετε επιπλέον πακέτα/βιβλιοθήκες λογισμικού, στη γενική περίπτωση συνιστάται το εξής: Αφού ρυθμίσετε και ενεργοποιήσετε το εικονικό σας περιβάλλον (μέσω conda ή venv), θα πρέπει να εγκαταστήσετε τις βιβλιοθήκες που απαιτούνται για την εκτέλεση των εργασιών χρησιμοποιώντας το pip . Για να το κάνετε αυτό, εκτελέστε:

```
# again, ensure your virtual env (either conda or venv)
# has been activated before running the commands below
cd assignment1 # cd to the assignment directory

# install assignment dependencies since the virtual env
# is activated, this pip is associated with the python
# binary of the environment
pip install -r requirements.txt
```

- Εκτέλεση Jupyter Notebook: Εάν θέλετε να εκτελέσετε το notebook τοπικά με το Jupyter, βεβαιωθείτε ότι το εικονικό σας περιβάλλον έχει εγκατασταθεί σωστά (σύμφωνα με τις οδηγίες εγκατάστασης που περιγράφονται παραπάνω για περιβάλλον linux), ενεργοποιήστε το και, στη συνέχεια, εκτελέστε pip install notebook για να εγκαταστήσετε το σημειωματάριο Jupyter . Στη συνέχεια, αφού κατεβάσετε και αποσυμπιέσετε το φάκελο της 1ης σειράς ασκήσεων από τη σελίδα ecourse σε κάποιο κατάλογο της επιλογής σας, εκτελέστε cd σε αυτόν το φάκελο και στη συνέχεια εκτελέστε το σημειωματάριο jupyter notebook . Αυτό θα πρέπει να εκκινήσει αυτόματα έναν διακομιστή notebook στη διεύθυνση http://localhost:8888 . Εάν όλα έγιναν σωστά, θα πρέπει να δείτε μια οθόνη που θα εμφανίζει όλα τα διαθέσιμα σημειωματάρια στον τρέχοντα κατάλογο, στην προκειμένη περίπτωση μόνο το assignment1.ipynb . Κάντε κλικ στο assignment1.ipynb και ακολουθήστε τις οδηγίες στο σημειωματάριο.

- Σε περιβάλλον Windows

Τα πρόγραμμα είναι πολύ πιο απλά στην περίπτωση που θέλετε να εργαστείτε τοπικά σε περιβάλλον Windows . Μπορείτε να εγκαταστήσετε την [Anaconda](https://www.anaconda.com/products/distribution) (<https://www.anaconda.com/products/distribution>) για Windows και στη συνέχεια να εκτελέσετε το [Anaconda Navigator](https://docs.anaconda.com/navigator/index.html) (<https://docs.anaconda.com/navigator/index.html>) αναζητώντας το απευθείας στο πεδίο αναζήτησης δίπλα από το κουμπί έναρξης των Windows. Το εργαλείο αυτό παρέχει επίσης άμεσα προεγκατεστημένα, τα πακέτα λογισμικού Jupyter Notebook και JupyterLab τα οποία επιτρέπουν την προβολή και υλοποίηση του σημειωματάριου Jupyter της 1ης εργασίας άμεσα και εύκολα (εκτελώντας το απευθείας από τη διαδρομή αρχείου που βρίσκεται). Ενδεχομένως, κατά την αποθήκευση/εξαγωγή του notebook assignment1.ipynb σε assignment1.pdf , να χρειαστεί η εγκατάσταση του πακέτου [Pandoc universal document converter](https://pandoc.org/installing.html) (<https://pandoc.org/installing.html>) (εκτέλεση: conda install -c conda-forge pandoc μέσω από το command prompt του "activated" anaconda navigator). Εναλλακτικά, μπορεί να εκτυπωθεί ως PDF αρχείο (βλ. Ενότητα: Οδηγίες υποβολής).

Python

Θα χρησιμοποιήσουμε τη γλώσσα προγραμματισμού Python για όλες τις εργασίες σε αυτό το μάθημα, με μερικές δημοφιλείς βιβλιοθήκες (NumPy , Matplotlib). Οι εργασίες θα δοθούν στη μορφή του σημειοματαρίου Jupyter notebook, όπως η εφαρμογή διακομιστή ιστοσελίδας που βλέπετε αυτήν τη στιγμή. Αναμένεται ότι πολλοί από εσάς έχετε κάποια εμπειρία σε Python και NumPy . Και αν έχετε πρότερη εμπειρία

Getting Started with NumPy

NumPy είναι μια θεμελιώδης βιβλιοθήκη για επιστημονικούς υπολογισμούς με Python. Παρέχει ένα ισχυρό τρόπο αναπαράστασης N-διάστατων πινάκων ως αντικείμενα ndarray και συναρτήσεις που ενεργούν πάνω σε αυτούς τους πινάκες. Μερικές βασικές χρήσεις αυτών των πακέτων παρουσιάζονται παρακάτω. Τα ακόλουθα ΔΕΝ είναι ζητούμενα της 1ης σειράς ασκήσεων, αλλά συνιστάται να εκτελέσετε τον ακόλουθο κώδικα με κάποια από τα στοιχεία εισόδου, τροποποιημένα, ώστε να κατανοήσετε καλύτερα το νόημα των λειτουργιών.

Arrays

In [5]:

```
1 import numpy as np          # Import the NumPy package
2 v = np.array([1, 2, 3])      # A 1D array
3 print(v, "\n")
4
5 print(v.shape, "\n")         # Print the size / shape of v
6
7 print("1D array:", v, "Shape:", v.shape, "\n")
8
9 v = np.array([[1], [2], [3]]) # A 2D array
10
11 print("2D array:", v, "Shape:", v.shape, "\n") # Print the size of v and check the difference.
12
13 # You can also attempt to compute and print the following values and their size.
14
15 v = v.T                     # Transpose of a 2D array
16 print(v, "\n")
17 print(v.shape, "\n")
18 m = np.zeros([3, 4])          # A 3x4 array (i.e. matrix) of zeros
19 print(m, "\n")
20 v = np.ones([1, 3])           # A 1x3 array (i.e. a row vector) of ones
21 print(v, "\n")
22 v = np.ones([3, 1])           # A 3x1 array (i.e. a column vector) of ones
23 print(v, "\n")
24 m = np.eye(4)                # Identity matrix
25 print(m, "\n")
26 m = np.random.rand(2, 3)      # A 2x3 random matrix with values in [0, 1] (sampled from uniform distribution)
27 print(m, "\n")
28 m_sum=np.sum(m, axis=None)    # element-wise sum of all matrix elements
29 print("m_sum:", m_sum, "\n")
```

[1 2 3]

(3,)

1D array: [1 2 3] Shape: (3,)

2D array: [[1]

[2]

[3]] Shape: (3, 1)

[[1 2 3]]

(1, 3)

[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]

[[1. 1. 1.]]

[[1.]
 [1.]
 [1.]])

[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

[[0.54747727 0.63900492 0.61172527]
 [0.53488006 0.83970118 0.40510157]]

m_sum: 3.577890278438314

Array Indexing

In [6]:

```
1 import numpy as np
2
3 print("Matrix")
4 m = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) # Create a 3x3 array.
5 print(m, "\n")
6
7 print("\nAccess a single element")
8 print(m[0, 1], "\n")                                # Access an element
9 m[1, 1] = 100                                         # Modify an element
10 print("\nModify a single element")
11 print(m, "\n")
12
13 print("\nAccess a subarray")
14 m = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) # Create a 3x3 array.
15 print(m[1, :], "\n")                                # Access a row (to 1D array)
16 print(m[1:2, :], "\n")                             # Access a row (to 2D array)
17 print(m[1:3, :], "\n")                             # Access a sub-matrix
18 print(m[1:, :], "\n")                            # Access a sub-matrix
19
20 print("\nModify a subarray")
21 m = np.array([[1,2,3], [4,5,6], [7,8,9]]) # Create a 3x3 array.
22 v1 = np.array([1,1,1])
23 m[0] = v1
24 print(m, "\n")
25 m = np.array([[1,2,3], [4,5,6], [7,8,9]]) # Create a 3x3 array.
26 v1 = np.array([1,1,1])
27 m[:,0] = v1
28 print(m, "\n")
29 m = np.array([[1,2,3], [4,5,6], [7,8,9]]) # Create a 3x3 array.
30 m1 = np.array([[1,1],[1,1]])
31 m[:2,:2] = m1
32 print(m, "\n")
33
34 print("\nTranspose a subarray")
35 m = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) # Create a 3x3 array.
36 print(m[1, :].T, "\n")                           # Notice the difference of the dimension of result
37 print(m[1:2, :].T, "\n")
38 print(m[1:, :].T, "\n")
39 print(np.transpose(m[1:, :, axes=(1,0))), "\n")    # np.transpose() can be used to transpose according
40
41 print("\nReverse the order of a subarray")
42 print(m[1, ::-1])                                # Access a row with reversed order (to 1D array)
43
44 # Boolean array indexing
45 # Given a array m, create a new array with values equal to m
46 # if they are greater than 2, and equal to 0 if they less than or equal to 2
47 m = np.array([[1, 2, 3], [4, 5, 6]])
48 m[m > 2] = 0
49 print("\nBoolean array indexing: Modify with a scalar")
50 print(m)
51
52 # Given a array m, create a new array with values equal to those in m
53 # if they are greater than 0, and equal to those in n if they less than or equal 0
54 m = np.array([[1, 2, -3], [4, -5, 6]])
55 n = np.array([[1, 10, 100], [1, 10, 100]])
56 n[m > 0] = m[m > 0]
57 print("\nBoolean array indexing: Modify with another array")
58 print(n)
```

Matrix

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Access a single element

```
2
```

Modify a single element

```
[[ 1   2   3]
 [ 4 100   6]
 [ 7   8   9]]
```

Access a subarray

```
[4 5 6]
```

```
[[4 5 6]]
```

```
[[4 5 6]
 [7 8 9]]
```

```
[[4 5 6]
 [7 8 9]]
```

```
Modify a subarray
```

```
[[1 1 1]
 [4 5 6]
 [7 8 9]]
```

```
[[1 2 3]
 [1 5 6]
 [1 8 9]]
```

```
Transpose a subarray
```

```
[4 5 6]
```

```
[[4]
 [5]
 [6]]
```

```
[[4 7]
 [5 8]
 [6 9]]
```

```
[[4 7]
 [5 8]
 [6 9]]
```

```
Reverse the order of a subarray
```

```
[6 5 4]
```

```
Boolean array indexing: Modify with a scalar
```

```
[[1 2 0]
 [0 0 0]]
```

```
Boolean array indexing: Modify with another array
```

```
[[ 1   2 100]
 [ 4  10   6]]
```

Array Dimension Operation

In [7]:

```
1 import numpy as np
2
3 print("Matrix")
4 m = np.array([[1, 2], [3, 4]]) # Create a 2x2 array.
5 print(m, m.shape)
6
7 print("\nReshape")
8 re_m = m.reshape(1,2,2) # Add one more dimension at first.
9 print(re_m, re_m.shape)
10 re_m = m.reshape(2,1,2) # Add one more dimension in middle.
11 print(re_m, re_m.shape)
12 #print('rows:',re_m[0:4, :])
13 re_m = m.reshape(2,2,1) # Add one more dimension at last.
14 print(re_m, re_m.shape)
15 #print('rows:',re_m[1:, :])
16
17 print("\nStack")
18 m1 = np.array([[1, 2], [3, 4]]) # Create a 2x2 array.
19 m2 = np.array([[1, 1], [1, 1]]) # Create a 2x2 array.
20 print(np.stack((m1,m2)))
21
22 print("\nConcatenate")
23 m1 = np.array([[1, 2], [3, 4]]) # Create a 2x2 array.
24 m2 = np.array([[1, 1], [1, 1]]) # Create a 2x2 array.
25 print(np.concatenate((m1,m2)))
26 print(np.concatenate((m1,m2), axis=0))
27 print(np.concatenate((m1,m2), axis=1))
```

Matrix
[[1 2]
 [3 4]] (2, 2)

Reshape
[[[1 2]
 [3 4]]] (1, 2, 2)
[[[1 2]]

 [[3 4]]] (2, 1, 2)
[[[1]
 [2]]

 [[3]
 [4]]] (2, 2, 1)

Stack
[[[1 2]
 [3 4]]

 [[1 1]
 [1 1]]]

Concatenate
[[1 2]
 [3 4]
 [1 1]
 [1 1]]
[[1 2]
 [3 4]
 [1 1]
 [1 1]]
[[1 2 1 1]
 [3 4 1 1]]

Math Operations on Array

Element-wise Operations

In [8]:

```

1 import numpy as np
2
3 a = np.array([[1, 2, 3], [4, 5, 6]], dtype=np.float64)
4 print(a * 3, "\n")                                # Scalar multiplication
5 print(a / 2, "\n")                                # Scalar division
6 print(np.round(a / 2), "\n")
7 print(np.power(a, 2), "\n")
8 print(np.log(a), "\n")
9 print(np.exp(a), "\n")
10
11 b = np.array([[1, 1, 1], [2, 2, 2]], dtype=np.float64)
12 print(a + b)                                     # Elementwise sum
13 print(a - b)                                     # Elementwise difference
14 print(a * b)                                     # Elementwise product
15 print(a / b)                                     # Elementwise division
16 print(a == b)                                    # Elementwise comparison

```

[[3. 6. 9.]
 [12. 15. 18.]]

[[0.5 1. 1.5]
 [2. 2.5 3.]]

[[0. 1. 2.]
 [2. 2. 3.]]

[[1. 4. 9.]
 [16. 25. 36.]]

[[0. 0.69314718 1.09861229]
 [1.38629436 1.60943791 1.79175947]]

[[2.71828183 7.3890561 20.08553692]
 [54.59815003 148.4131591 403.42879349]]

[[2. 3. 4.]
 [6. 7. 8.]]

[[0. 1. 2.]
 [2. 3. 4.]]

[[1. 2. 3.]
 [8. 10. 12.]]

[[1. 2. 3.]
 [2. 2.5 3.]]

[[True False False]
 [False False False]]

Broadcasting

In [9]:

```

1 # Note: See https://numpy.org/doc/stable/user/basics.broadcasting.html
2 # for more details.
3 import numpy as np
4 a = np.array([[1, 1, 1], [2, 2, 2]], dtype=np.float64)
5 b = np.array([1, 2, 3])
6 print(a*b)

```

[[1. 2. 3.]
 [2. 4. 6.]]

Sum and Mean

In [10]:

```

1 import numpy as np
2
3 a = np.array([[1, 2, 3], [4, 5, 6]])
4 print("Sum of array")
5 print(np.sum(a))                                # Sum of all array elements
6 print(np.sum(a, axis=0))                         # Sum of each column (row sum)
7 print(np.sum(a, axis=1))                         # Sum of each row (column sum)
8
9 print("\nMean of array")
10 print(np.mean(a))                               # Mean of all array elements
11 print(np.mean(a, axis=0))                        # Mean of each column
12 print(np.mean(a, axis=1))                        # Mean of each row

```

Sum of array

21

[5 7 9]

[6 15]

Mean of array

3.5

[2.5 3.5 4.5]

[2. 5.]

Vector and Matrix Operations

In [11]:

```

1 import numpy as np
2
3 a = np.array([[1, 2], [3, 4]])
4 b = np.array([[1, 1], [1, 1]])
5 print("Matrix-matrix product")
6 print(a.dot(b))                      # Matrix-matrix product
7 print(a.T.dot(b.T))
8
9 x = np.array([3, 4])
10 print("\nMatrix-vector product")
11 print(a.dot(x))                     # Matrix-vector product
12
13 x = np.array([1, 2])
14 y = np.array([3, 4])
15 print("\nVector-vector product")
16 print(x.dot(y))                   # Vector-vector product

```

Matrix-matrix product

```

[[3 3]
 [7 7]]
[[4 4]
 [6 6]]

```

Matrix-vector product

```
[11 25]
```

Vector-vector product

```
11
```

Matplotlib

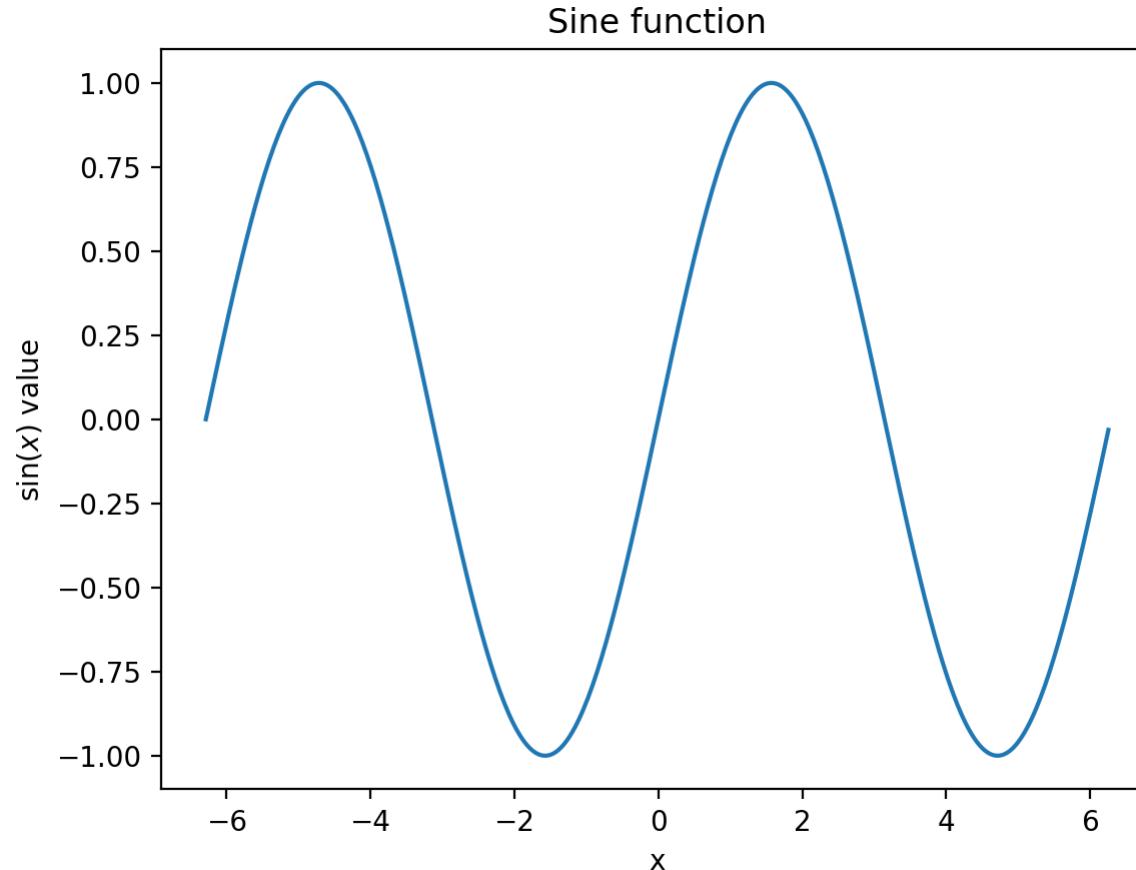
Η Matplotlib είναι μια βιβλιοθήκη σχεδίασης και αναπαράστασης. Θα τη χρησιμοποιήσουμε για να δείξουμε το αποτέλεσμα σε αυτήν την εργασία.

In [12]:

```

1 %config InlineBackend.figure_format = 'retina' # For high-resolution.
2 %matplotlib inline
3
4 import numpy as np
5 import matplotlib.pyplot as plt
6
7 x = np.arange(-2., 2., 0.01) * np.pi
8 plt.plot(x, np.sin(x))
9 plt.xlabel('x')
10 plt.ylabel('$\sin(x)$ value') # '$...$' for a LaTeX formula.
11 plt.title('Sine function')
12
13 plt.show()

```



Αυτή η σύντομη επισκόπηση εισάγει πολλές βασικές λειτουργίες από NumPy και Matplotlib, αλλά απέχει πολύ από την πλήρη εικόνα των βιβλιοθηκών. Δείτε περισσότερες λειτουργίες και τη χρήση τους στους συνδέσμους [NumPy](https://docs.scipy.org/doc/numpy/reference/) (<https://docs.scipy.org/doc/numpy/reference/>) and [Matplotlib](https://matplotlib.org/) (<https://matplotlib.org/>).

Άσκηση 1: Στοιχειώδεις λειτουργίες επεξεργασίας εικόνας και διανυσματοποίηση (10 μονάδες)

Οι πράξεις μεταξύ διανυσμάτων αξιοποιώντας τη βιβλιοθήκη NumPy μπορούν να παρέχουν μια σημαντική επιτόχυνση για την εποναληπτική εκτέλεση μιας λειτουργίας σε μια εικόνα. Το παρακάτω πρόβλημα παρουσιάζει το χρόνο που χρειάζονται δύο διαφορετικές προσεγγίσεις για να αλλάξουν το χρώμα των τεταρτημορίων μιας εικόνας.

Το πρόβλημα διαβάζει μια εικόνα `uo_i_entrance.jpg` που θα βρείτε στο φάκελο `images` της 1ης σειράς ασκήσεων. Στη συνέχεια παρέχονται δύο συναρτήσεις ως διαφορετικές προσεγγίσεις για την εκτέλεση μιας λειτουργίας στην εικόνα.

Η συνάρτηση `iterative()` απεικονίζει την εικόνα, χωρισμένη σε 4 περιοχές:

- (Top Left) Η αρχική εικόνα
(Top Right) Η πράσινη χρωματική συνιστώσα.
(Bottom Left) Η αναδιόταξη των καναλιών της αρχικής εικόνας, ως (B,G,R) έγχρωμη εικόνα.
(Bottom Right) Η `Grayscale` εικόνα (μονοχρωματική εικόνα σε κλίμακα του γκρι).

Για την υλοποίησή σας:

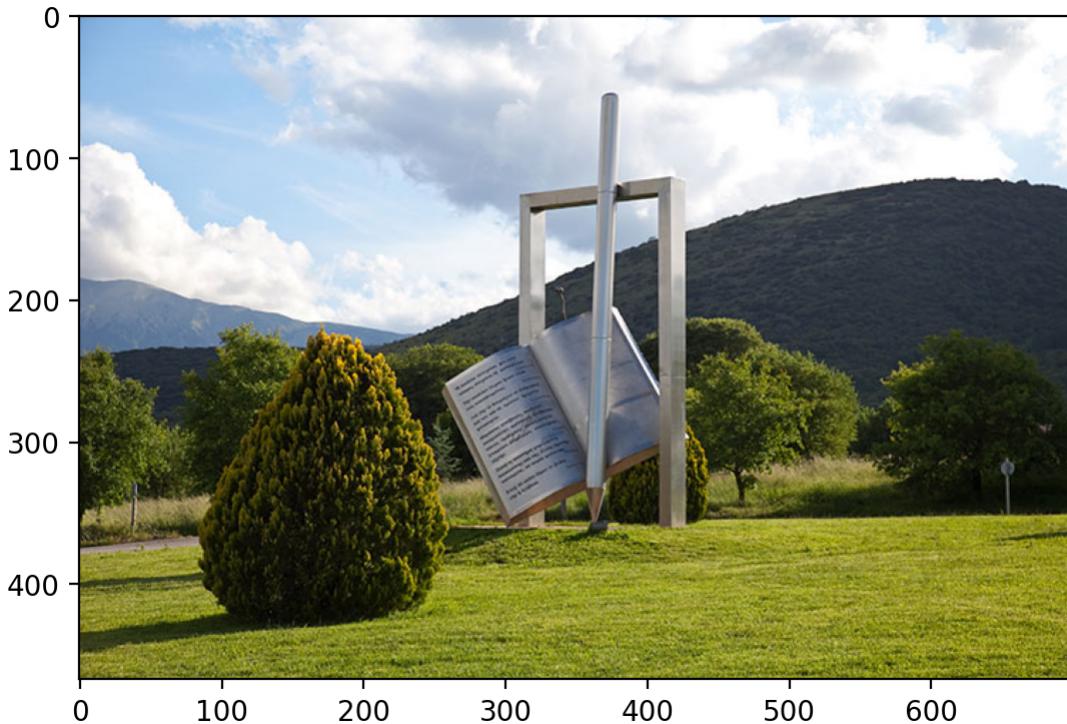
- (1) Για την πράσινη χρωματική συνιστώσα της εικόνας, γράψτε την υλοποίησή σας (`get_channel()`) για να εξαγάγετε ένα μόνο κανάλι από μια έγχρωμη εικόνα. Αυτό σημαίνει ότι από την $H \times W \times 3$ διάσταση της εικόνας, θα πρέπει να μπορείτε να εξάγετε τρεις (2Δ) πίνακες $H \times W$.
- (2) Για την (B,G,R) έγχρωμη εικόνα, γράψτε την υλοποίηση της συνάρτησης `merge_channels()` η οποία συγχωνεύει αυτές τις εικόνες ενός καναλιού ξανά σε μια τρισδιάστατη έγχρωμη εικόνα με την αντίστροφη σειρά καναλιών (B,G,R).
- (3) Για την (`grayscale`) εικόνα σε κλίμακα του γκρι, γράψτε μια συνάρτηση για τη διεξαγωγή λειτουργιών με τα εξαγόμενα κανάλια. (Υπόδειξη: γράψτε μια συνάρτηση που συνδυάζει τα 3 εξαγόμενα κανάλια με τρόπο αντίστοιχο της `iterative()`, δηλαδή $0.2989 * r + 0.5870 * g + 0.1140 * b$).

Θα πρέπει να ακολουθήσετε τον κώδικα και να συμπληρώσετε τα κενά στην `vectorized()` συνάρτηση προκειμένου να συγκρίνετε τη διαφορά στην ταχύτητα εκτέλεσης μεταξύ των συναρτήσεων `iterative()` και `vectorized()`. Βεβαιωθείτε ότι η τελική εικόνα που δημιουργείται ιέστια της `vectorized()` είναι ίδια με αυτή που δημιουργείται από την `iterative()`!

In [13]:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 img = plt.imread('images/uo_i_entrance.jpg') # Read an image
5 print("Image shape:", img.shape)           # Print image size and color depth. The shape should be (H,W,C)
6
7 plt.imshow(img)                          # Show the original image
8 plt.show()
```

Image shape: (467, 700, 3)



In [14]:

```

1 import copy
2 import time
3 def iterative(img):
4     """ Iterative operation. """
5     image = copy.deepcopy(img) # Create a copy of the image matrix
6     for y in range(image.shape[0]):
7         for x in range(image.shape[1]):
8             #Top Right
9             if y < image.shape[0]/2 and x > image.shape[1]/2:
10                 image[y,x] = image[y,x] * np.array([0,1,0]) # Keep the green channel
11             #Bottom Left
12             elif y > image.shape[0]/2 and x < image.shape[1]/2:
13                 image[y,x] = [image[y,x][2], image[y,x][1], image[y,x][0]] #(B,G,R) image
14             #Bottom Right
15             elif y > image.shape[0]/2 and x > image.shape[1]/2:
16                 r,g,b = image[y,x]
17                 image[y,x] = 0.2989 * r + 0.5870 * g + 0.1140 * b
18     return image
19
20 def get_channel(img, channel):
21     """ Function to extract 2D image corresponding to a channel index from a color image.
22     This function should return a H*W array which is the corresponding channel of the input image. """
23     img = copy.deepcopy(img) # Create a copy so as to not change the original image
24     channel_img = img[:, :, channel] #take all the elements of the first two dimensions of img,
25                                     #but only the channel in the third dimension
26     return channel_img #return a 2-dimensional array representing the image with only the specified chan
27
28
29
30 def merge_channels(img0, img1, img2):
31     """ Function to merge three single channel images to form a color image.
32     This function should return a H*W*3 array which merges all three single channel images
33     (i.e. img0, img1, img2) in the input."""
34     ##### Write your code here. #####
35     # Hint: There are multiple ways to implement it.
36     #       1. For example, create a H*W*C array with all values as zero and
37     #          fill each channel with given single channel image.
38     #          You may refer to the "Modify a subarray" section in the brief NumPy tutorial above.
39     #       2. You may find np.stack() / np.concatenate() / np.reshape() useful in this problem.
40     merged_img = np.zeros((img0.shape[0], img0.shape[1], 3)) # Create a zero array of the same size
41                                         #as the input image
42     merged_img[:, :, 0] = img2 #assigns img2 to the first channel of merged_img red
43     merged_img[:, :, 1] = img1 #assigns img1 to the second channel of merged_img blue
44     merged_img[:, :, 2] = img0 #assigns img0 to the third channel of merged_img green
45     return merged_img # return the specified channel color image
46
47
48 def vectorized(img):
49     """ Vectorized operation. """
50     image = copy.deepcopy(img) # create a deep copy of the input image
51     a = int(image.shape[0]/2) #divide image into two parts
52     b = int(image.shape[1]/2) #divide image into another two parts
53     # Please also keep the red / green / blue channel respectively in the corresponding part of image
54     # with the vectorized operations. You need to make sure your final generated image in this
55     # vectorized() function is the same as the one generated from iterative().
56
57     #Top Right: keep the green channel
58     image[:a,b:,0] = 0 #set the red to zero
59     image[:a,b:,2] = 0 #set the blue to zero
60
61     #Bottom Left: (B,G,R) image
62     image[a:,:b] = np.flip(image[a:,:b], axis=2) #Flip the image along the RGB axis to get a BGR image
63
64
65     #With this approach i had problems with the dimensions of the image
66     #gray = np.dot(image[a:,b,:], [0.2989, 0.5870, 0.1140])
67     #gray = gray.reshape(gray.shape[0], 1, 1)
68     #image[a:,b] = np.transpose([gray, gray, gray])
69     #image[a:, b:, :] = gray
70
71     #Bottom Right: Grayscale image
72     gray = np.mean(image[a:, b:], axis=2, keepdims=True) # take the mean of RGB channels which gives the
73                                         # by taking third axis of the subarray image[a:, b:]
74                                         # keepdims=True keeps the right dimensions
75
76     image[a:, b:] = gray
77
78     return image

```

Τώρα, εκτελέστε το παρακάτω κελί κώδικα για να συγκρίνετε τη διαφορά μεταξύ επαναληπτικής (iterative) και διανυσματικής (vectorized) λειτουργίας.

In [15]:

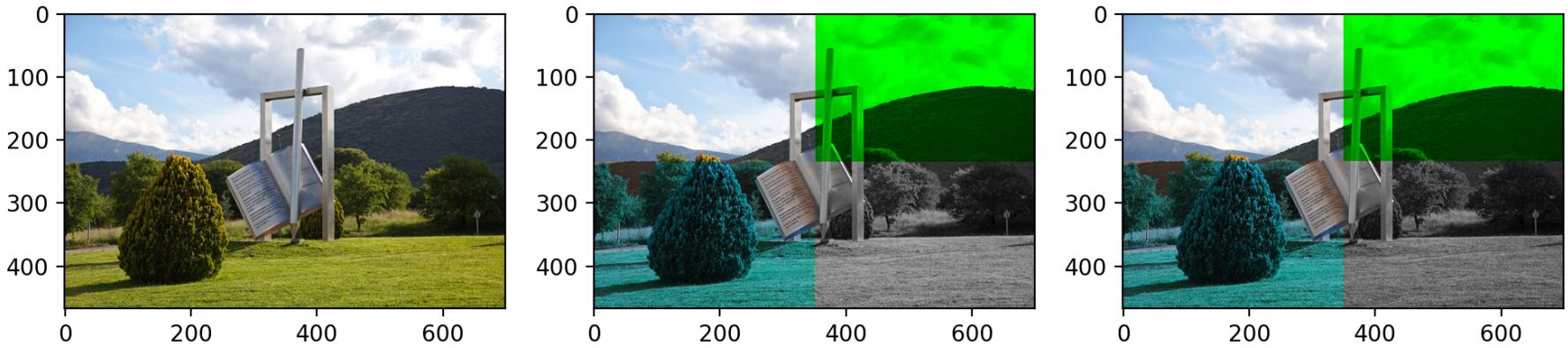
```

1 import time
2
3 def compare():
4     img = plt.imread('images/uoi_entrance.jpg')
5     cur_time = time.time()
6     image_iterative = iterative(img)
7     print("Iterative operation (sec):", time.time() - cur_time)
8
9     cur_time = time.time()
10    image_vectorized = vectorized(img)
11    print("Vectorized operation (sec):", time.time() - cur_time)
12
13    return image_iterative, image_vectorized
14
15 # Test your implemented get_channel()
16 assert len(get_channel(img, 0).shape) == 2 # Index 0
17
18 # Run the function
19 image_iterative, image_vectorized = compare()
20
21 # Plotting the results in sepearate subplots.
22 plt.figure(figsize=(12,4)) # Adjust the figure size.
23 plt.subplot(1, 3, 1) # Create 1x3 subplots, indexing from 1
24 plt.imshow(img) # Original image.
25
26 plt.subplot(1, 3, 2)
27 plt.imshow(image_iterative) # Iterative operations on the image.
28
29 plt.subplot(1, 3, 3)
30 plt.imshow(image_vectorized) # Vectorized operations on the image.
31
32 plt.show() # Show the figure.
33
34 # Note: The shown figures of image_iterative and image_vectorized should be identical!

```

Iterative operation (sec): 1.1297614574432373

Vectorized operation (sec): 0.003428220748901367



Άσκηση 2: Επεξεργασία-χειρισμός εικόνων (15 μονάδες)

Σε αυτή την άσκηση θα χρησιμοποιήσετε την εικόνα `bear.png` στο φάκελο `images`. Όντας έγχρωμη εικόνα έχει τρία κανάλια, τα οποία αντιστοιχούν στα κύρια χρώματα του κόκκινου, του πράσινου και του μπλε.

(1) "Διαβάστε" την εικόνα. (Read the image).

(2) Γράψτε μια συνάρτηση για να αναστρέψετε (`flip`) την αρχική εικόνα από πάνω προς τα κάτω. Για αυτήν τη συνάρτηση, χρησιμοποιήστε μόνο τεχνικές της ενότητας **Array Indexing** για την υλοποίησή της και **μην** χρησιμοποιείτε απευθείας τις συναρτήσεις (π.χ. `np.flip()`) που αναστρέφουν απευθείας τον πίνακα.

(3) Στη συνέχεια, γράψτε μια άλλη συνάρτηση για να περιστρέψετε την αρχική εικόνα 90° αριστερόστροφα (counterclockwise). Για αυτήν τη συνάρτηση, χρησιμοποιήστε μόνο **Array Indexing** για την υλοποίησή της και **μην** χρησιμοποιείτε απευθείας τις συναρτήσεις (π.χ. `np.rot90()`) που περιστρέφουν απευθείας τον πίνακα. Εμφανίστε τρεις εικόνες, εφαρμόζοντας για κάθε περίπτωση τη λειτουργία περιστροφής μία φορά (δηλαδή περιστροφή 90°), δύο φορές (δηλαδή περιστροφή 180°) και τρεις φορές (δηλαδή περιστροφή 270°), αντίστοιχα.

(4) Διαβάστε (read) την εικόνα `face-mask.png` και την αντίστοιχη εικόνα δυαδικής μάσκας `face-mask-binary.png` στο φάκελο `images`.

(5) Δεδομένου του `start_x` και `start_y` στην εικόνα της αρκούδας που υποδεικνύει την αρχική θέση (πάνω-αριστερή γωνία) της μάσκας προσώπου, πρέπει να γράψετε μια συνάρτηση για να βοηθήσετε την αρκούδα να "φορέσει" τη μάσκα προσώπου (Υπόδειξη: τιμές εικονοστοιχιών 1 της δυαδικής μάσκας υποδεικνύουν ποια εικονοστοιχία της **εικόνας** θα εμφανίζονται).

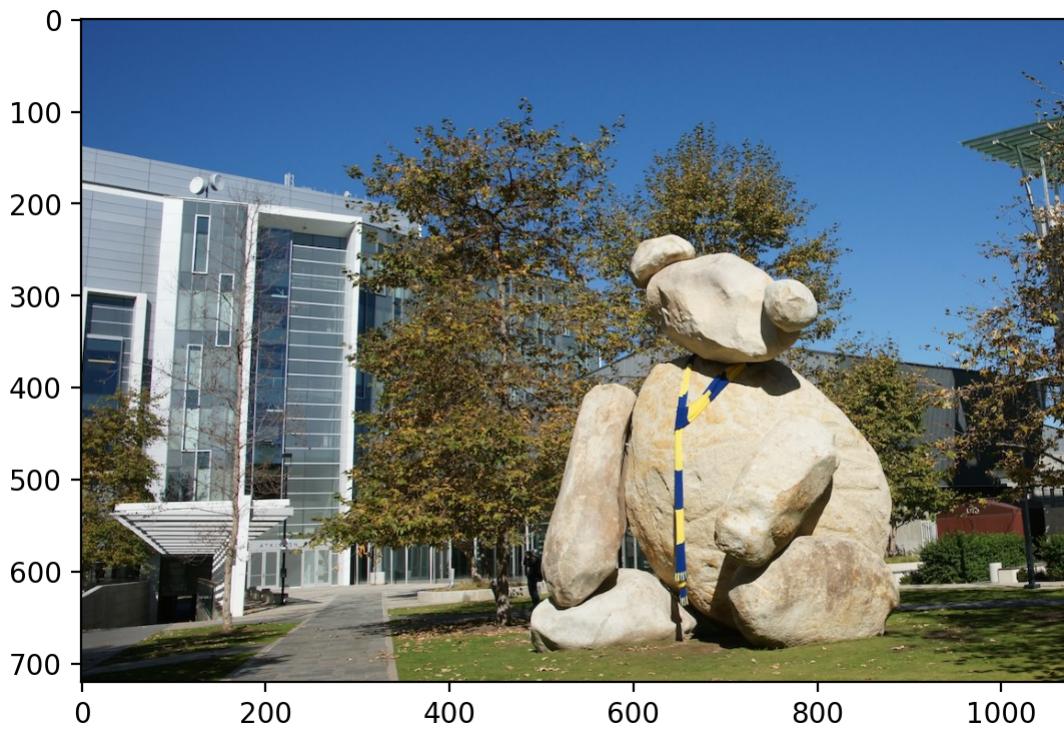
(6) Τέλος, λαμβάνοντας υπόψη τις **4 έγχρωμες εικόνες** που έχετε πάρει ως αποτέλεσμα των ανωτέρω: 1 αρχική εικόνα αρκούδας, 1 από αναστροφή (πάνω προς τα κάτω), 1 από περιστροφή (180° μοιρών) και 1 για την αρκούδα που φορά τη μάσκα προσώπου, δημιουργήστε μία μεμονωμένη εικόνα, συνδυάζοντας αυτές τις 4 εικόνες μαζί (image tiling) **χωρίς να χρησιμοποιήσετε βρόχους επανάληψης**. Η συνολική εικόνα θα έχει πλακίδια (`tiles`) 2×2 που θα διαμορφώνουν το σχήμα της τελικής εικόνας σε $2H \times 2W \times 3$. Η σειρά με την οποία τοποθετούνται οι εικόνες στα πλακίδια δεν έχει σημασία. Εμφανίστε τη συνολική εικόνα.

In [16]:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import copy
```

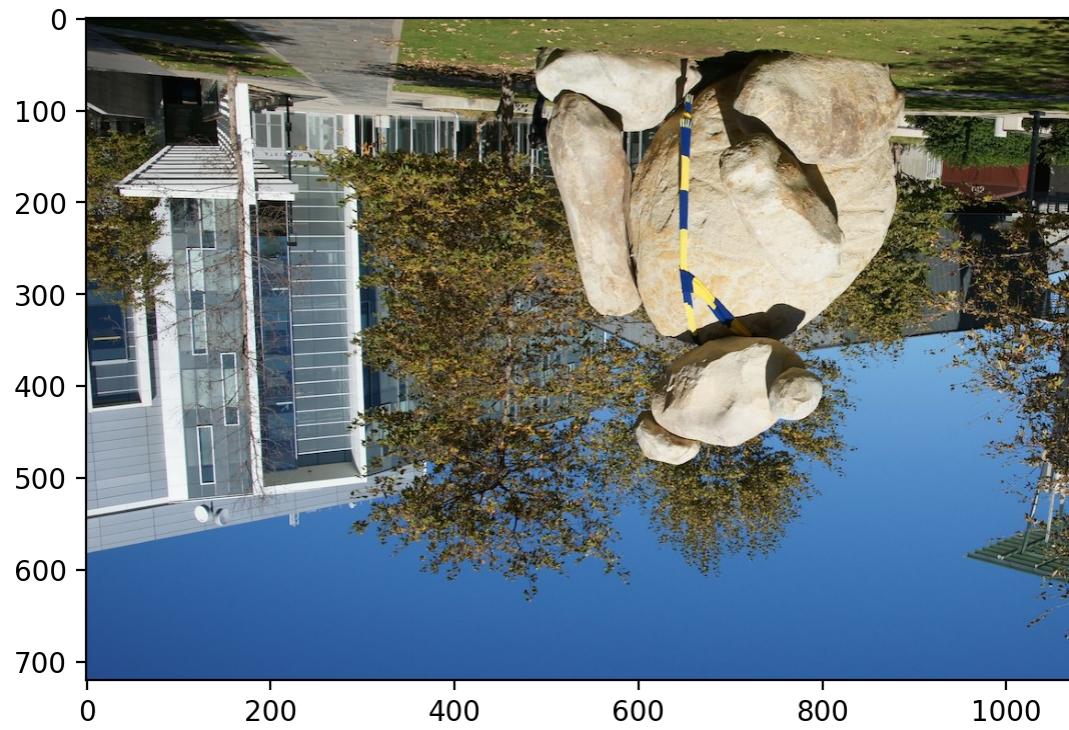
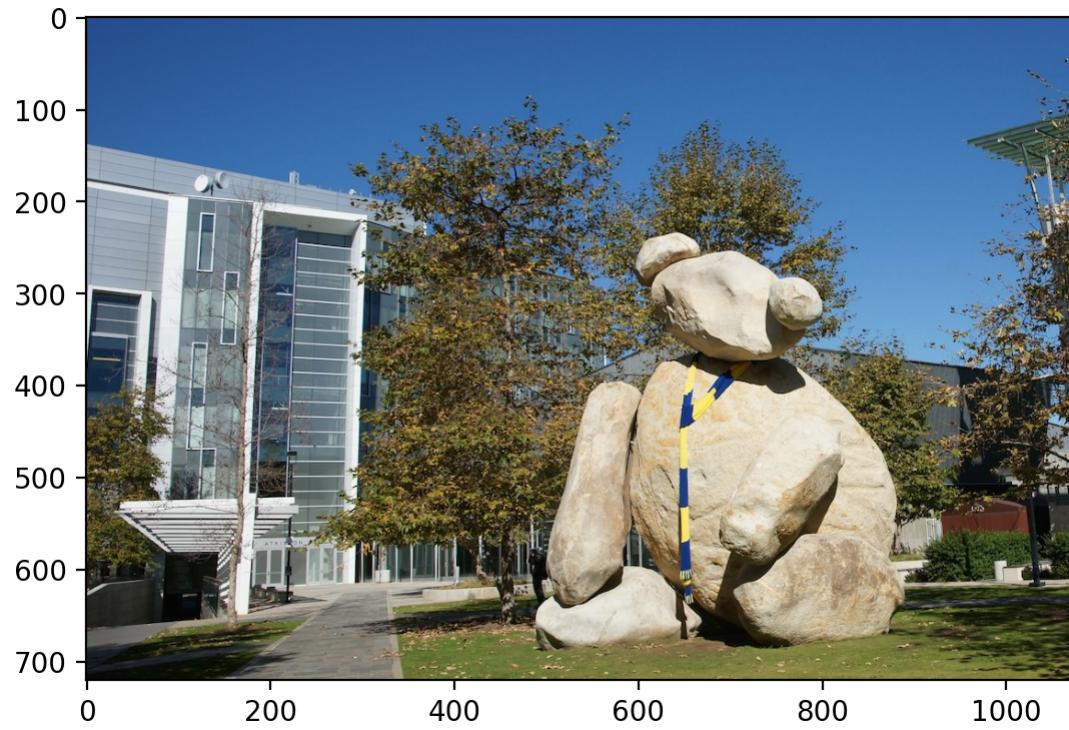
In [17]:

```
1 # (1) Read the image.
2 ##### Write your code here. #####
3 img = plt.imread('images/bear.png') # Read an image
4
5 plt.imshow(img) # Show the image after reading.
6 plt.show()
```



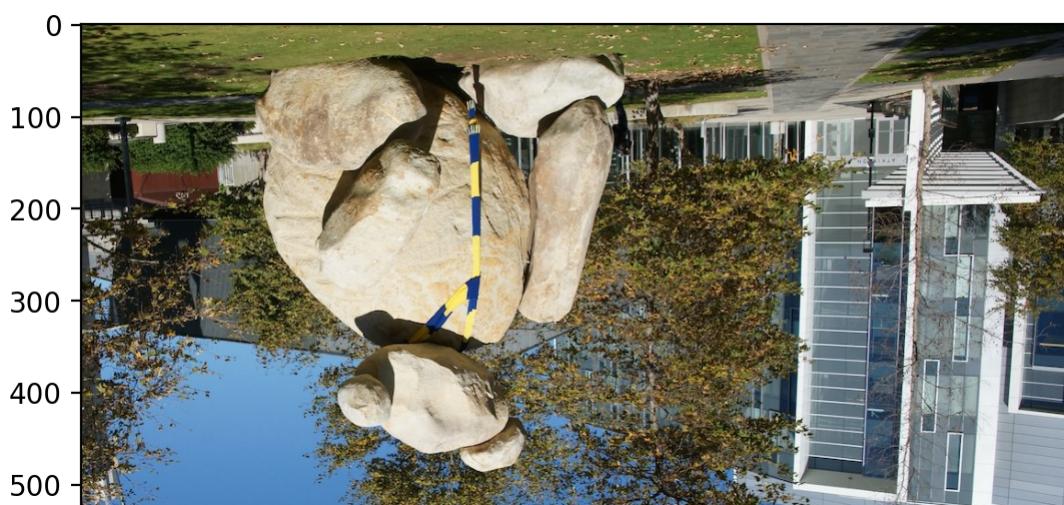
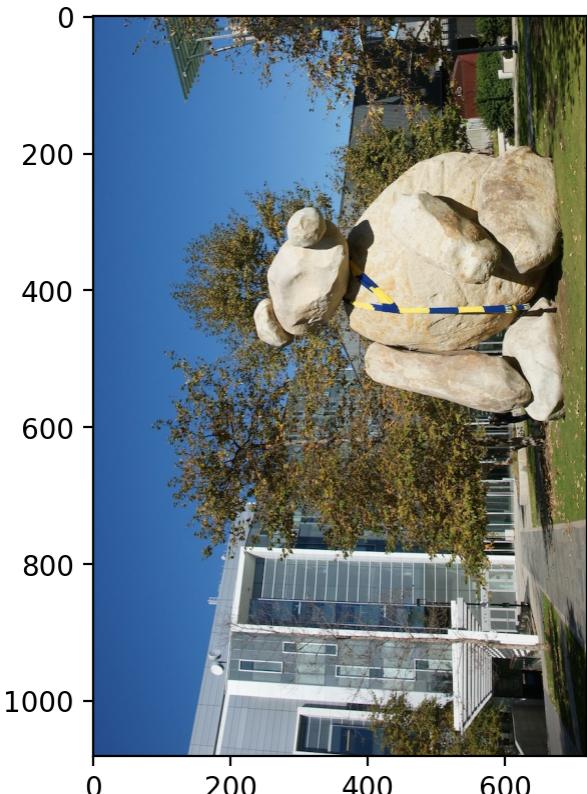
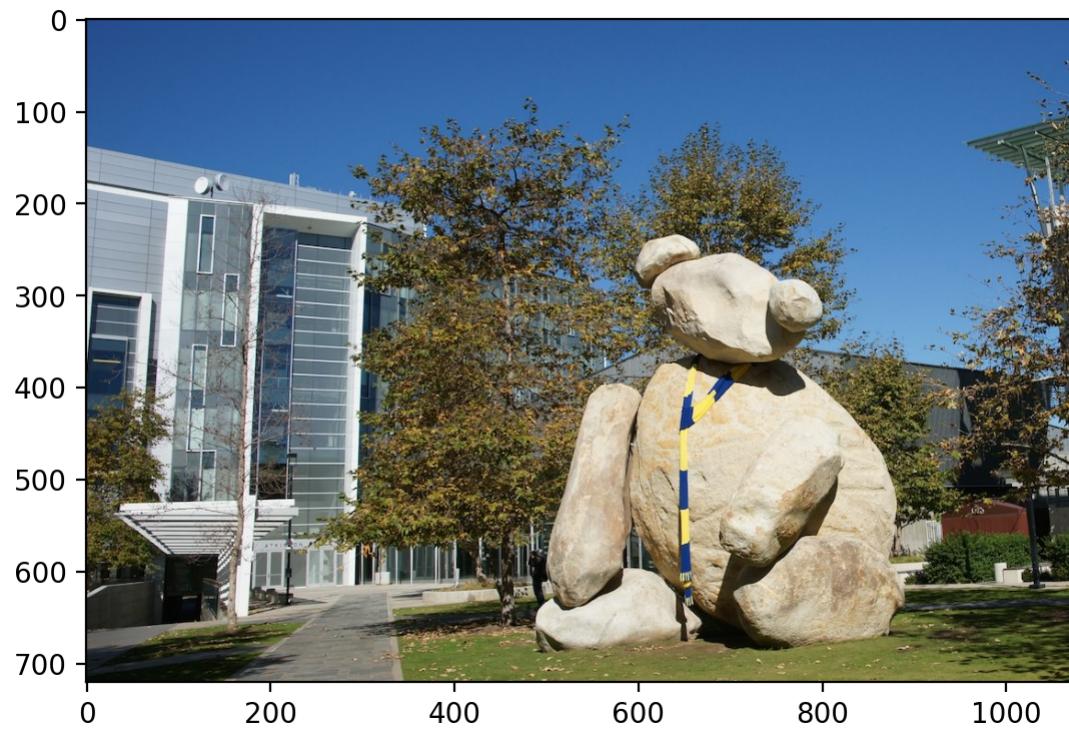
In [18]:

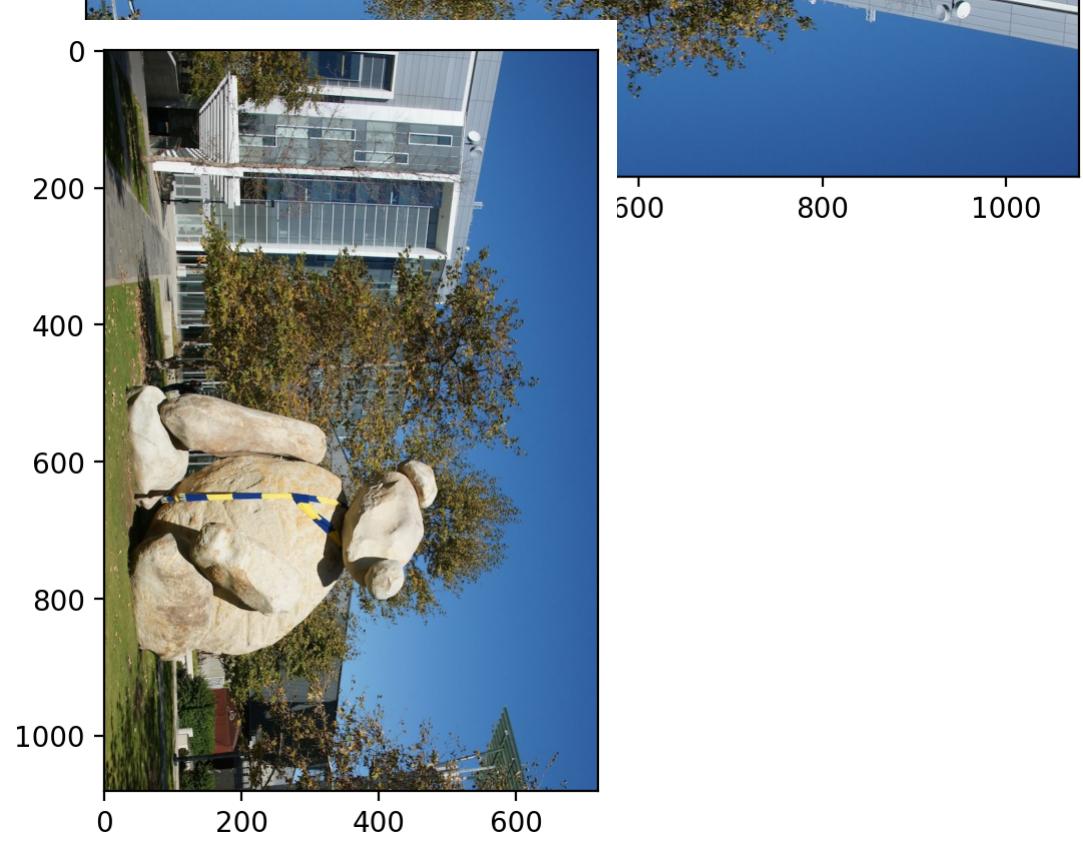
```
1 # (2) Flip the image from top to bottom.
2 def flip_img(img):
3     """ Function to mirror image from top to bottom.
4     This function should return a H*W*3 array which is the flipped version of original image.
5     """
6     flipped_img = img[::-1, :, :] #use slicing to reverse the order of the first axis (rows of the image)
7     return flipped_img           # resulting in the flipping of the image
8
9
10 plt.imshow(img)
11 plt.show()
12 flipped_img = flip_img(img)
13 plt.imshow(flipped_img)
14 plt.show()
```



In [19]:

```
1 # (3) Rotate image.
2 def rotate_90(img):
3     """ Function to rotate image 90 degrees counter-clockwise.
4     This function should return a W*H*3 array which is the rotated version of original image. """
5     ##### Write your code here. #####
6     h, w, c = img.shape # height, width, color
7     rotated_img = np.zeros((w, h, c), dtype=img.dtype) # change height with width and width with height
8     for i in range(w): # Loop through each column of the rotated image and
9         #copy the corresponding row from the original image
10        rotated_img[i, :, :] = img[:, w-i-1, :]
11    return rotated_img
12
13 plt.imshow(img)
14 plt.show()
15 rot90_img = rotate_90(img)
16 plt.imshow(rot90_img)
17 plt.show()
18 rot180_img = rotate_90(rotate_90(img))
19 plt.imshow(rot180_img)
20 plt.show()
21 rot270_img = rotate_90(rotate_90(rotate_90(img)))
22 plt.imshow(rot270_img)
23 plt.show()
```





In [20]:

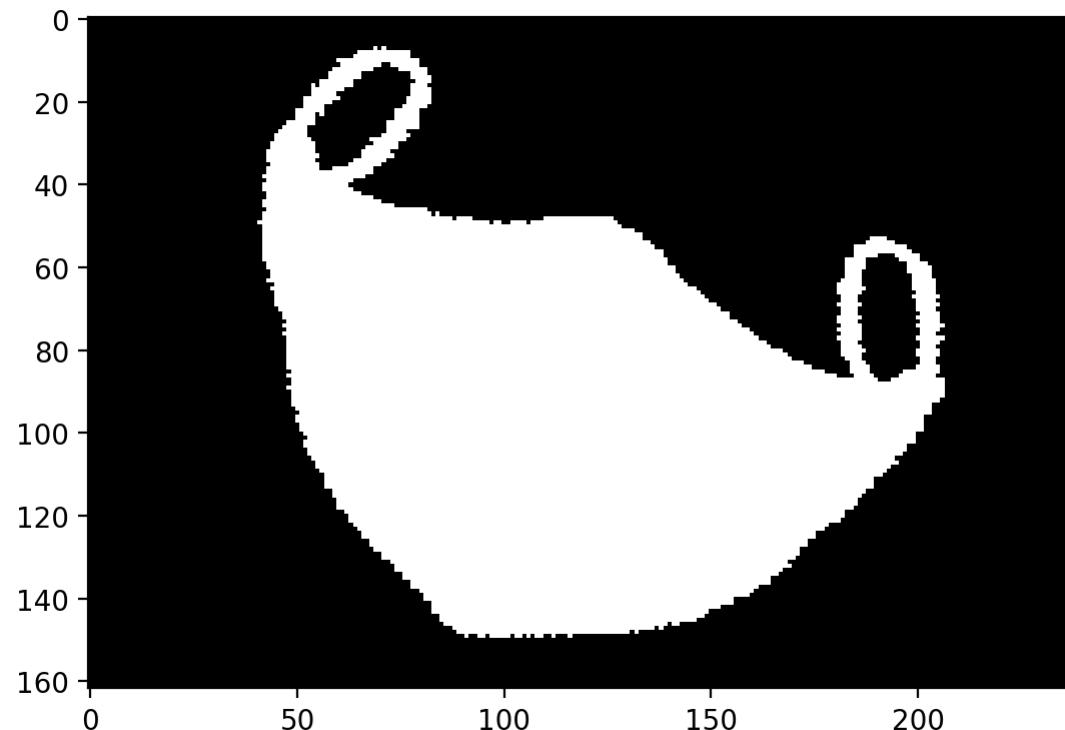
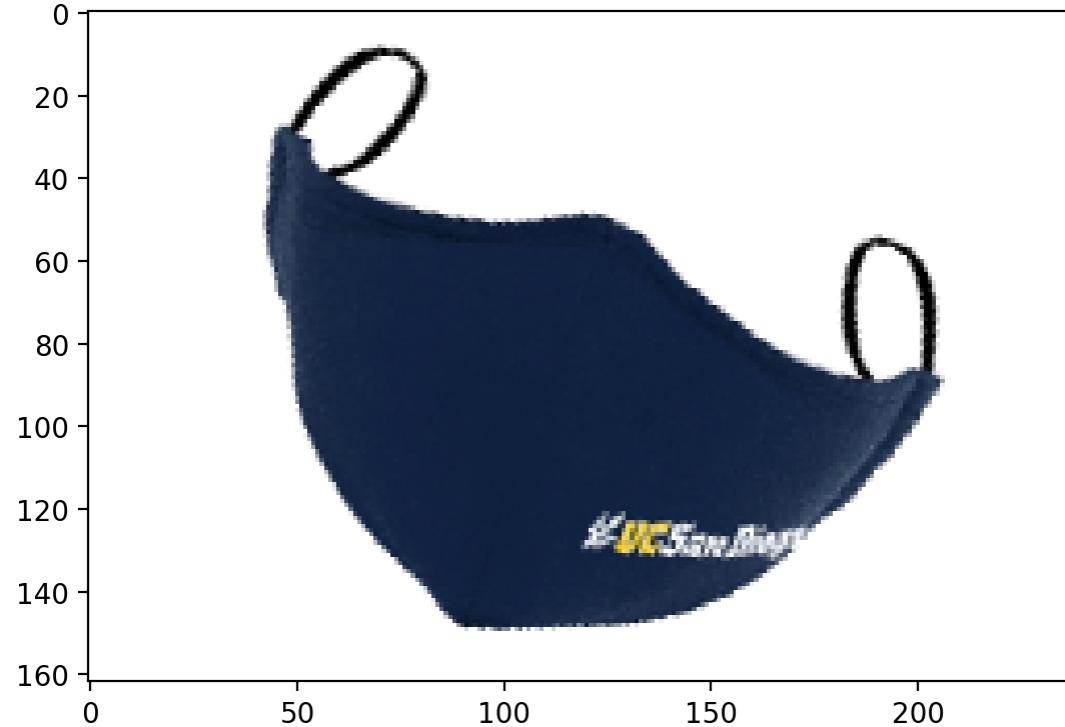
```
1 # (4)Read the face mask image and the face mask binary image
2
3 ##### Write your code here. #####
4
5 mask_img = plt.imread('images/face-mask.png') # Read an image
6 bi_mask_img = plt.imread('images/face-mask-binary.png') # Read an image
7
8 print("Face Mask Image Size: ")
9 print(mask_img.shape) #dimensions
10 print("Face Mask Binary Mask Image Size: ")
11 print(bi_mask_img.shape) #dimensions
12
13 plt.imshow(mask_img)
14 plt.show()
15 plt.imshow(bi_mask_img)
16 plt.show()
```

Face Mask Image Size:

(162, 240, 3)

Face Mask Binary Mask Image Size:

(162, 240, 3)

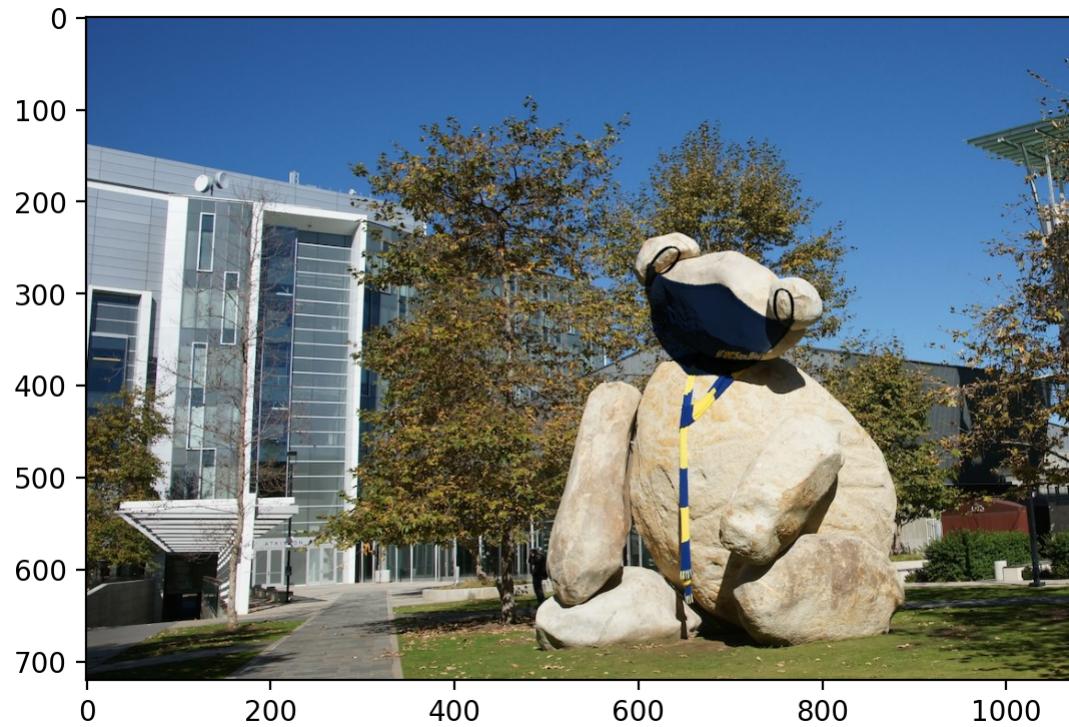


In [21]:

```

1 # (5) Put the face mask on the bear's face
2 start_x = 565 #location x of the mask
3 start_y = 240 #location y of the mask
4
5 maskon_img = copy.deepcopy(img) # Make a copy
6
7 end_x = start_x + mask_img.shape[1] # Calculate the end coordinate x of the mask based on its shape
8 end_y = start_y + bi_mask_img.shape[0] # Calculate the end coordinate y of the mask based on its shape
9
10 maskon_img[start_y:end_y, start_x:end_x, :] *= mask_img # Apply the mask to the image
11                                         #by setting the corresponding pixels to zero
12
13 # Show the resulting image
14 plt.imshow(maskon_img)
15 plt.show()
16
17
18

```

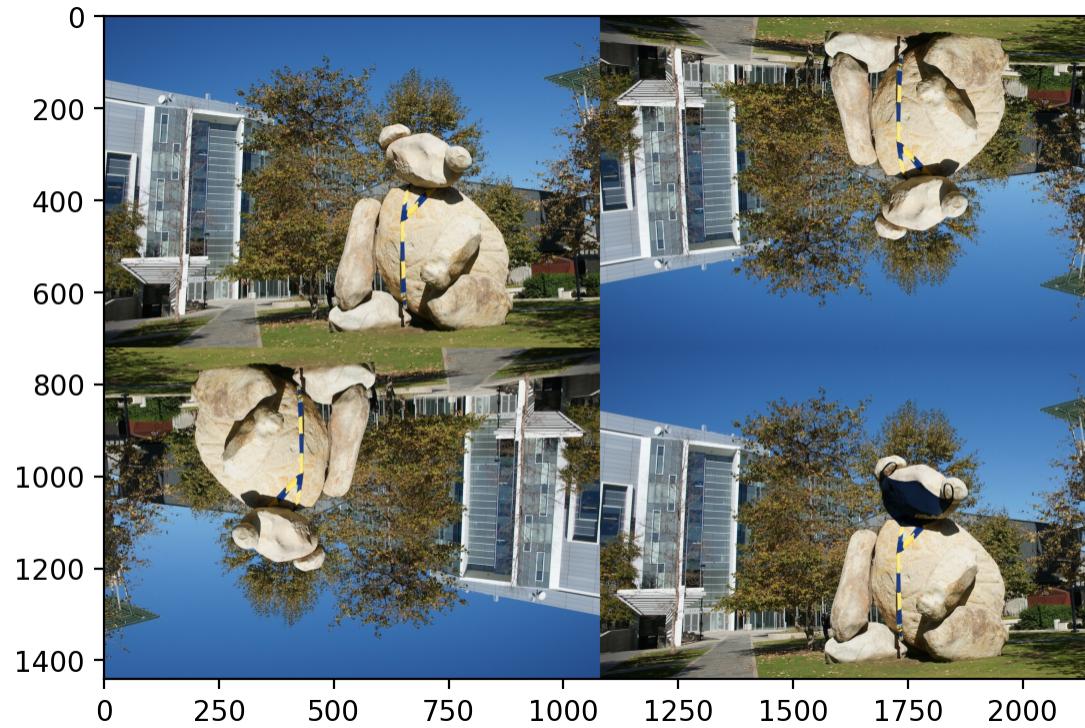


In [22]:

```

1 # (6) Write your code here to tile the four images and make a single image.
2 # You can use the img, flipped_img, rot180_img, maskon_img to represent the four images.
3 # After tiling, please display the tiled image.
4
5 tiled_img_top = np.concatenate((img, flipped_img), axis=1) #produce two horizontally concatenated images
6                                         #with same height but double width
7 tiled_img_bottom = np.concatenate((rot180_img, maskon_img), axis=1) #produce two horizontally concatenated images
8                                         #with same height but double width
9 tiled_img = np.concatenate((tiled_img_top, tiled_img_bottom), axis=0) #produce two vertically concatenated images
10                                         #with double height and double width
11
12 # Display the tiled image
13 plt.imshow(tiled_img)
14 plt.show()
15

```



Οδηγίες υποβολής

Μην ξεχάσετε να κάνετε turnin **τόσο** το αρχείο Jupyter notebook όσο και το PDF αρχείο αυτού του notebook μαζί με το συνοδευτικό αρχείο

onoma.txt : turnin assignment_1@mye046 onoma.txt assignment1.ipynb assignment1.pdf

Βεβαιωθείτε ότι το περιεχόμενο σε **κάθε κελί εμφανίζεται** καθαρά στο τελικό σας αρχείο PDF. Για να μετατρέψετε το σημειωματάριο σε PDF, μπορείτε να επιλέξετε **έναν** από τους παρακάτω τρόπους:

1. Google Collab (Συνιστάται): You can print the web page and save as PDF (e.g. Chrome: Right click the web page → Print... → Choose "Destination: Save as PDF" and click "Save"). Προσοχή στην περίπτωση όπου κώδικας/σχόλια εμφανίζονται εκτός των ορίων της σελίδας. Μια λύση είναι η αλλαγή γραμμής π.χ. σε σχόλια που υπερβαίνουν το πλάτος της σελίδας.
2. Local Jupyter/JupyterLab(Συνιστάται): You can print the web page and save as PDF (File → Print... → Choose "Destination: Save as PDF" and click "Save"). Προσοχή στην περίπτωση όπου κώδικας/σχόλια εμφανίζονται εκτός των ορίων της σελίδας. Μια λύση είναι η αλλαγή γραμμής π.χ. σε σχόλια που υπερβαίνουν το πλάτος της σελίδας.
3. Local Jupyter: You can find the export option in the header: File → Download as → "PDF via LaTeX" (Ενδέχεται να παρουσιάσει πρόβλημα στην απόδοση κειμένου στα Ελληνικά)
4. Local Jupyter: You can find the export option in the header: File → Download as → "LaTeX" file and then compile download .tex file (using e.g. TexMaker) to create the PDF file (Απαιτεί τη μετατροπή του notebook σε αρχείο "latex" και μετά μετατροπή του .tex αρχείου σε PDF).
5. Local JupyterLab: You can find the Save and Export option in the header: File → Save and Export Notebook as → "LaTeX" file and then compile download .tex file (using e.g. TexMaker) to create the PDF file (Απαιτεί τη μετατροπή του notebook σε αρχείο "latex" και μετά μετατροπή του .tex αρχείου σε PDF).