

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 ονόμα 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 3Δ ιδιοτήτων από εικόνες (σχήμα-από-σκίαση, 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](#) ή [Colaboratory](#).

Working remotely on Google Colaboratory

To [Google Colaboratory](#) είναι βασικά ένας συνδυασμός σημειωματαρίου Jupyter και [Google 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:

1. Κάντε κλικ στον τροχό στην επάνω δεξιά γωνία (στο Google Drive) και επιλέξτε **Ρυθμίσεις**.

2. Κάντε κλικ στην καρτέλα **Διαχείριση εφαρμογών**.
3. Στο επάνω μέρος, επιλέξτε **Σύνδεση περισσότερων εφαρμογών** που θα εμφανίσουν ένα παράθυρο του **GSuite Marketplace**.
4. Αναζητήστε το **Colab** και, στη συνέχεια, κάντε κλικ στην **Προσθήκη (install)**.

- Workflow:

Κάθε 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 πράγματα που πρέπει να γνωρίζετε όταν εργάζεστε με την υπηρεσία Colab. Το πρώτο πράγμα 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 στιγμότυπο 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** (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**, η οποία παρέχει έναν εύκολο τρόπο για να χειριστείτε 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 πράγματα είναι πολύ πιο απλά στην περίπτωση που θέλετε να εργαστείτε τοπικά σε περιβάλλον `Windows`. Μπορείτε να εγκαταστήσετε την `Anaconda` για Windows και στη συνέχεια να εκτελέσετε το `Anaconda Navigator` 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 σημειωματαρίου 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 (εκτέλεση: `conda install -c conda-forge pandoc` μέσα 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 iStoSeLίδας που βλέπετε αυτήν τη στιγμή. Αναμένεται ότι πολλοί από εσάς έχετε κάποια εμπειρία σε Python και NumPy . Και αν έχετε πρότερη εμπειρία σε MATLAB , μπορείτε να δείτε επίσης το σύνδεσμο NumPy for MATLAB users. Η παρακάτω ενότητα θα χρησιμεύσει ως μια γρήγορη εισαγωγή στη NumPy και σε ορισμένες άλλες βιβλιοθήκες.

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 πίνακες. 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 [14]: import numpy as np          # Import the NumPy package

v = np.array([1, 2, 3])           # A 1D array
print(v)
print(v.shape)                  # Print the size / shape of v
print("1D array:", v, "Shape:", v.shape)
v = np.array([[1], [2], [3]])    # A 2D array
print("2D array:", v, "Shape:", v.shape) # Print the size of v and check the difference

# You can also attempt to compute and print the following values and their size.

v = v.T                         # Transpose of a 2D array
print(v)
print(v.shape)
m = np.zeros([3, 4])              # A 2x3 array (i.e. matrix) of zeros
print(m)
v = np.ones([1, 3])               # A 1x3 array (i.e. a row vector) of ones
print(v)
v = np.ones([3, 1])               # A 3x1 array (i.e. a column vector) of ones
print(v)
m = np.eye(4)                     # Identity matrix
print(m)
m = np.random.rand(2, 3)          # A 2x3 random matrix with values in [0, 1] (sampled from a uniform distribution)
print(m)
```

```
m_sum=np.sum(m, axis=None)      # element-wise sum of all matrix elements
print("m_sum:",m_sum)

[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.]]
[[0.0715393  0.90481965  0.3782381 ]
 [0.78193775  0.52431769  0.72735703]]
m_sum: 3.3882095219292534
```

Array Indexing

```
In [15]: import numpy as np

print("Matrix")
m = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) # Create a 3x3 array.
print(m)

print("\nAccess a single element")
print(m[0, 1])                                # Access an element
m[1, 1] = 100                                 # Modify an element
print("\nModify a single element")
print(m)

print("\nAccess a subarray")
m = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) # Create a 3x3 array.
print(m[1, :])                                  # Access a row (to 1D array)
print(m[1:2, :])                               # Access a row (to 2D array)
print(m[1:3, :])                               # Access a sub-matrix
print(m[1:, :])                                # Access a sub-matrix

print("\nModify a subarray")
m = np.array([[1,2,3], [4,5,6], [7,8,9]]) # Create a 3x3 array.
v1 = np.array([1,1,1])
m[0] = v1
print(m)
m = np.array([[1,2,3], [4,5,6], [7,8,9]]) # Create a 3x3 array.
v1 = np.array([1,1,1])
m[:,0] = v1
print(m)
m = np.array([[1,2,3], [4,5,6], [7,8,9]]) # Create a 3x3 array.
m1 = np.array([[1,1],[1,1]])
m[1:2,:2] = m1
print(m)

print("\nTranspose a subarray")
m = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) # Create a 3x3 array.
```

```
print(m[1, :].T)                                # Notice the difference of the dimension  
print(m[1:2, :].T)  
print(m[1:, :].T)  
print(np.transpose(m[1:, :], axes=(1,0)))      # np.transpose() can be used to transpose arrays  
  
print("\nReverse the order of a subarray")  
print(m[1, ::-1])                                # Access a row with reversed order  
  
# Boolean array indexing  
# Given a array m, create a new array with values equal to m  
# if they are greater than 2, and equal to 0 if they less than or equal to 2  
m = np.array([[1, 2, 3], [4, 5, 6]])  
m[m > 2] = 0  
print("\nBoolean array indexing: Modify with a scaler")  
print(m)  
  
# Given a array m, create a new array with values equal to those in m  
# if they are greater than 0, and equal to those in n if they less than or equal to 0  
m = np.array([[1, 2, -3], [4, -5, 6]])  
n = np.array([[1, 10, 100], [1, 10, 100]])  
n[m > 0] = m[m > 0]  
print("\nBoolean array indexing: Modify with another array")  
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]]
[[1 1 3]
 [1 1 6]
 [7 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 [16]: import numpy as np

print("Matrix")
m = np.array([[1, 2], [3, 4]]) # Create a 2x2 array.
print(m, m.shape)
```

```

print("\nReshape")
re_m = m.reshape(1,2,2) # Add one more dimension at first.
print(re_m, re_m.shape)
re_m = m.reshape(2,1,2) # Add one more dimension in middle.
print(re_m, re_m.shape)
#print('rows:',re_m[0:4, :])
re_m = m.reshape(2,2,1) # Add one more dimension at last.
print(re_m, re_m.shape)
#print('rows:',re_m[1:, :])

print("\nStack")
m1 = np.array([[1, 2], [3, 4]]) # Create a 2x2 array.
m2 = np.array([[1, 1], [1, 1]]) # Create a 2x2 array.
print(np.stack((m1,m2)))

print("\nConcatenate")
m1 = np.array([[1, 2], [3, 4]]) # Create a 2x2 array.
m2 = np.array([[1, 1], [1, 1]]) # Create a 2x2 array.
print(np.concatenate((m1,m2)))
print(np.concatenate((m1,m2), axis=0))
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 [17]: `import numpy as np`

```
a = np.array([[1, 2, 3], [4, 5, 6]], dtype=np.float64)
print(a * 3)                                     # Scalar multiplication
print(a / 2)                                     # Scalar division
print(np.round(a / 2))
print(np.power(a, 2))
print(np.log(a))
print(np.exp(a))

b = np.array([[1, 1, 1], [2, 2, 2]], dtype=np.float64)
print(a + b)                                     # Elementwise sum
print(a - b)                                     # Elementwise difference
print(a * b)                                     # Elementwise product
print(a / b)                                     # Elementwise division
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 [18]:

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

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

Sum and Mean

In [19]:

```
import numpy as np

a = np.array([[1, 2, 3], [4, 5, 6]])
print("Sum of array")
print(np.sum(a))                                # Sum of all array elements
print(np.sum(a, axis=0))                          # Sum of each column (row sum)
print(np.sum(a, axis=1))                          # Sum of each row (column sum)

print("\nMean of array")
print(np.mean(a))                               # Mean of all array elements
print(np.mean(a, axis=0))                        # Mean of each column
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 [20]: import numpy as np
```

```
a = np.array([[1, 2], [3, 4]])
b = np.array([[1, 1], [1, 1]])
print("Matrix-matrix product")
print(a.dot(b))                      # Matrix-matrix product
print(a.T.dot(b.T))
```

```
x = np.array([3, 4])
print("\nMatrix-vector product")
print(a.dot(x))                      # Matrix-vector product
```

```
x = np.array([1, 2])
y = np.array([3, 4])
print("\nVector-vector product")
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

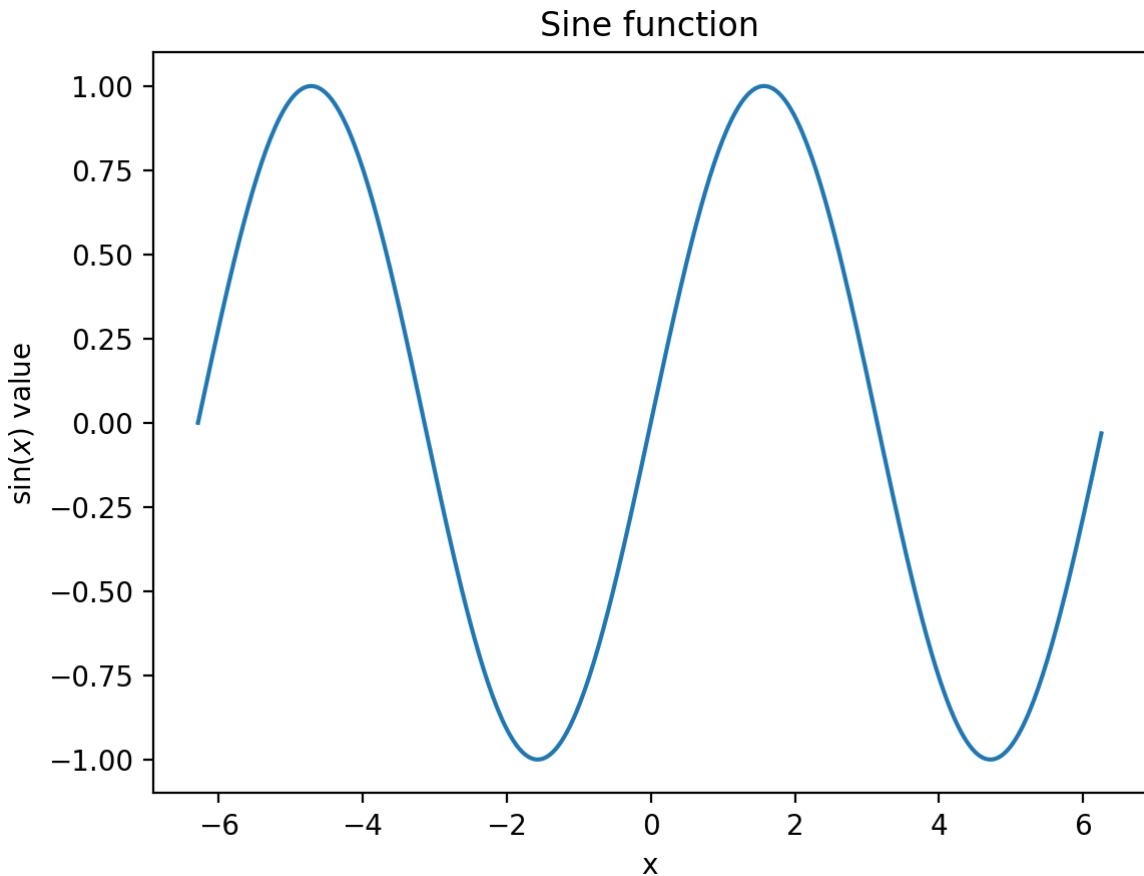
H `Matplotlib` είναι μια βιβλιοθήκη σχεδίασης και αναπαράστασης. Θα τη χρησιμοποιήσουμε για να δείξουμε το αποτέλεσμα σε αυτήν την εργασία.

```
In [22]: %config InlineBackend.figure_format = 'retina' # For high-resolution.
%matplotlib inline
```

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(-2., 2., 0.01) * np.pi
plt.plot(x, np.sin(x))
plt.xlabel('x')
plt.ylabel('$\sin(x)$ value') # '$...$' for a LaTeX formula.
plt.title('Sine function')

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](#) and [Matplotlib](#).

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

Οι πράξεις μεταξύ διανυσμάτων αξιοποιώντας τη βιβλιοθήκη `NumPy` μπορούν να παρέχουν μια σημαντική επιτάχυνση για την επαναληπτική 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 αναδιάταξη 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 μέσω της `vectorized()` είναι ίδια με αυτή που δημιουργείται από την `iterative()`!

```
In [55]: import numpy as np
import matplotlib.pyplot as plt

img = plt.imread('images/uoi_entrance.jpg') # Read an image
print("Image shape:", img.shape)           # Print image size and color depth. The

plt.imshow(img)                          # Show the original image
plt.show()
```

Image shape: (467, 700, 3)



```
In [49]: import copy
import time
def iterative(img):
```

```

    """ Iterative operation. """
    image = copy.deepcopy(img)                      # Create a copy of the image matrix
    for y in range(image.shape[0]):
        for x in range(image.shape[1]):
            #Top Right
            if y < image.shape[0]/2 and x > image.shape[1]/2:
                image[y,x] = image[y,x] * np.array([0,1,0])      # Keep the green channel
            #Bottom Left
            elif y > image.shape[0]/2 and x < image.shape[1]/2:
                image[y,x] = [image[y,x][2], image[y,x][1], image[y,x][0]]  #(B,G,R)
            #Bottom Right
            elif y > image.shape[0]/2 and x > image.shape[1]/2:
                r,g,b = image[y,x]
                image[y,x] = 0.2989 * r + 0.5870 * g + 0.1140 * b
    return image

def get_channel(img, channel):
    """ Function to extract 2D image corresponding to a channel index from a color image.
    This function should return a H*W array which is the corresponding channel of the image.
    img = copy.deepcopy(img)      # Create a copy so as to not change the original
    ##### Write your code here. #####
    # We take only the channel we need depending which number user will give (0,1,2)
    img = img[:, :, channel]
    #print(np.shape(img))
    return img

def merge_channels(img0, img1, img2):
    """ Function to merge three single channel images to form a color image.
    This function should return a H*W*3 array which merges all three single channels (i.e. img0, img1, img2) in the input."""
    ##### Write your code here. #####
    # Hint: There are multiple ways to implement it.
    #       1. For example, create a H*W*C array with all values as zero and
    #          fill each channel with given single channel image.
    #          You may refer to the "Modify a subarray" section in the brief NumPy
    #       2. You may find np.stack() / np.concatenate() / np.reshape() useful in
    # Firstly we make an array with zeros and the
    # correct shape and then in each channel
    # we assing the channels we got from the
    # get_channel function but in this order: B G R
    mergeimg = np.zeros([img0.shape[0],img0.shape[1],3])
    mergeimg[:, :, 0] = img2
    mergeimg[:, :, 1] = img1
    mergeimg[:, :, 2] = img0
    return mergeimg

# A function to make an image in grayscale mode
def change_rgb_to_gray(img):
    # We assign in each channel the value we got from the equation bellow
    temp = img[:, :, 0] * 0.2989 + img[:, :, 1] * 0.5870 + img[:, :, 2] * 0.1140
    img[:, :, 0] = temp
    img[:, :, 1] = temp
    img[:, :, 2] = temp
    return img

def vectorized(img):
    """ Vectorized operation. """
    image = copy.deepcopy(img)
    a = int(image.shape[0]/2)
    b = int(image.shape[1]/2)
    # Please also keep the red / green / blue channel respectively in the corresponding
    # with the vectorized operations. You need to make sure your final generated image

```

```
# vectorized() function is the same as the one generated from iterative().

#Top Right: keep the green channel
#image[:, :, 1] = get_channel(image[:, :, 1], 1) # Keep the green channel
# We want only the green channel so we use the
# get_channel function with the parameters being
# the image and number 1 (the second channel - green).
# We need to make sure that the remaining channels must
# be 0
image[:, :, 0] = 0
image[:, :, 1] = get_channel(image[:, :, 1], 1) # Keep the green channel
image[:, :, 2] = 0

#Bottom Left: (B, G, R) image
# We use the merge_channels function with the parameters being the results
# from get_channel function for each channel separate.
image[:, :, :] = merge_channels(get_channel(image[:, :, 0], 0), \
                                 get_channel(image[:, :, 1], 1), \
                                 get_channel(image[:, :, 2], 2));

#Bottom Right: Grayscale image
# Calling the change_rgb_to_gray function to get rid of the colors
image[:, :, :] = change_rgb_to_gray(image[:, :, :])

return image
```

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

```
In [50]: import time

def compare():
    img = plt.imread('images/uoi_entrance.jpg')
    cur_time = time.time()
    image_iterative = iterative(img)
    print("Iterative operation (sec):", time.time() - cur_time)

    cur_time = time.time()
    image_vectorized = vectorized(img)
    print("Vectorized operation (sec):", time.time() - cur_time)

    return image_iterative, image_vectorized

# Test your implemented get_channel()
assert len(get_channel(img, 0).shape) == 2 # Index 0

# Run the function
image_iterative, image_vectorized = compare()

# Plotting the results in sepearate subplots.
plt.figure(figsize=(12,4)) # Adjust the figure size.
plt.subplot(1, 3, 1) # Create 1x3 subplots, indexing from 1
plt.imshow(img) # Original image.

plt.subplot(1, 3, 2)
plt.imshow(image_iterative) # Iterative operations on the image.

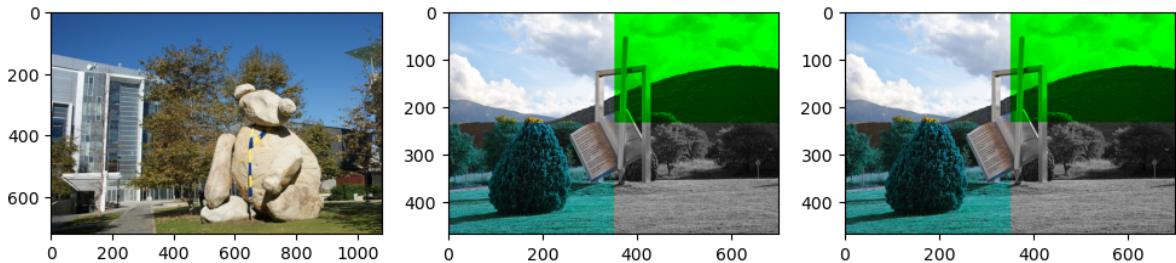
plt.subplot(1, 3, 3)
plt.imshow(image_vectorized) # Vectorized operations on the image.

plt.show() # Show the figure.
```

Note: The shown figures of `image_iterative` and `image_vectorized` should be identical.

Iterative operation (sec): 1.331303596496582

Vectorized operation (sec): 0.008204460144042969



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 εικονοστοιχείων 1 της δυαδικής **μάσκας** υποδεικνύουν ποια εικονοστοιχεία της **εικόνας** θα εμφανίζονται).

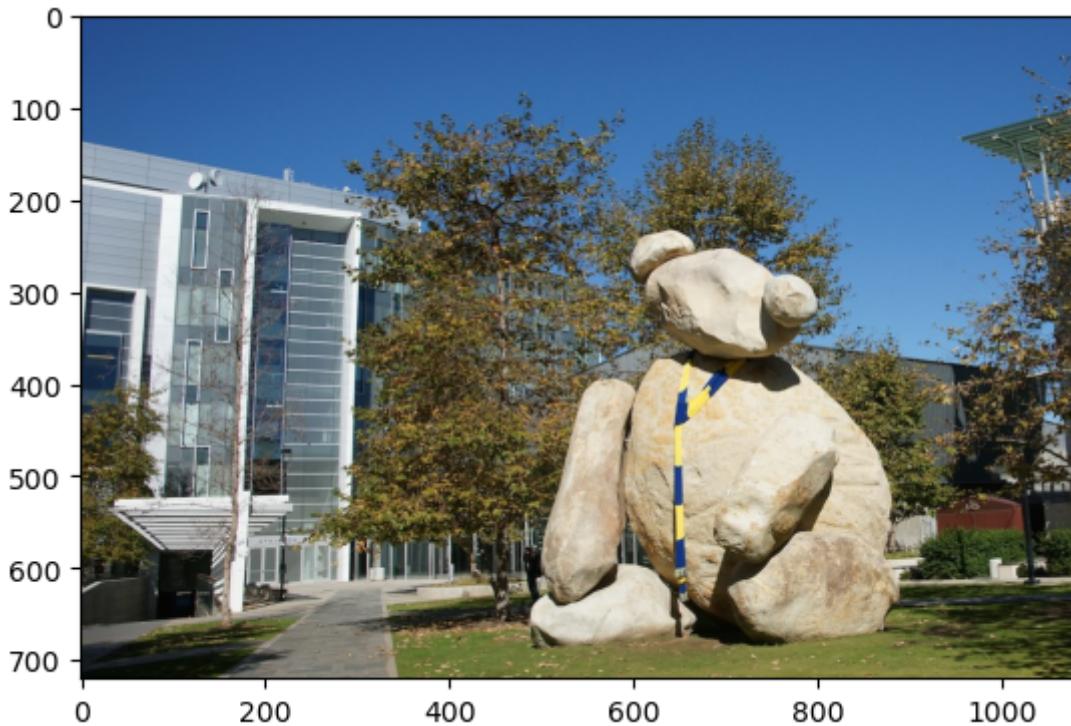
(6) Τέλος, λαμβάνοντας υπόψη τις **4 έγχρωμες εικόνες** που έχετε πάρει ως αποτέλεσμα των ανωτέρω: 1 αρχική εικόνα αρκούδας, 1 από αναστροφή (πάνω προς τα κάτω), 1 από περιστροφή (180 μοιρών) και 1 για την αρκούδα που φορά τη μάσκα προσώπου, δημιουργήστε μία μεμονωμένη εικόνα, συνδυάζοντας αυτές τις 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 [2]: import numpy as np
import matplotlib.pyplot as plt
import copy
```

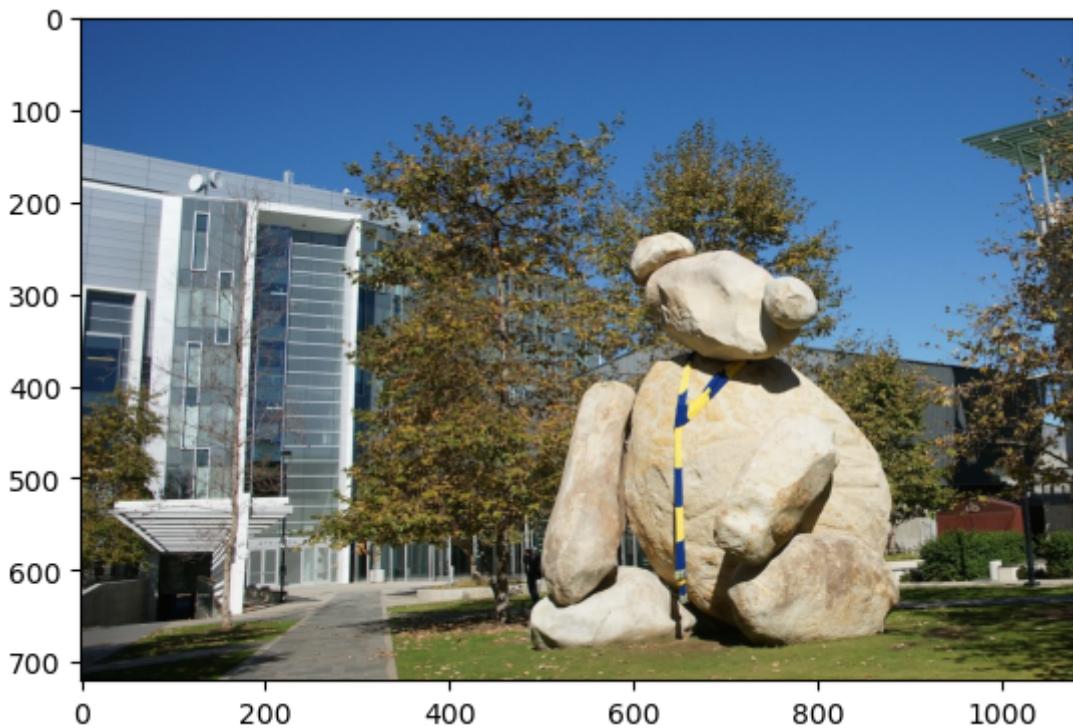
```
In [36]: # (1) Read the image.
##### Write your code here. #####
# We use plt.imread to read the image
img = plt.imread('images/bear.png')

plt.imshow(img) # Show the image after reading.
plt.show()
```



```
In [47]: # (2) Flip the image from top to bottom.
def flip_img(img):
    """ Function to mirror image from top to bottom.
    This function should return a H*W*3 array which is the flipped version of original.
    """
    ##### Write your code here. #####
    # We are reading the rows upside down
    # (reversed order) and we save them
    # in a new array so we get the flipped image
    flipedimg = img[::-1]
    return flipedimg

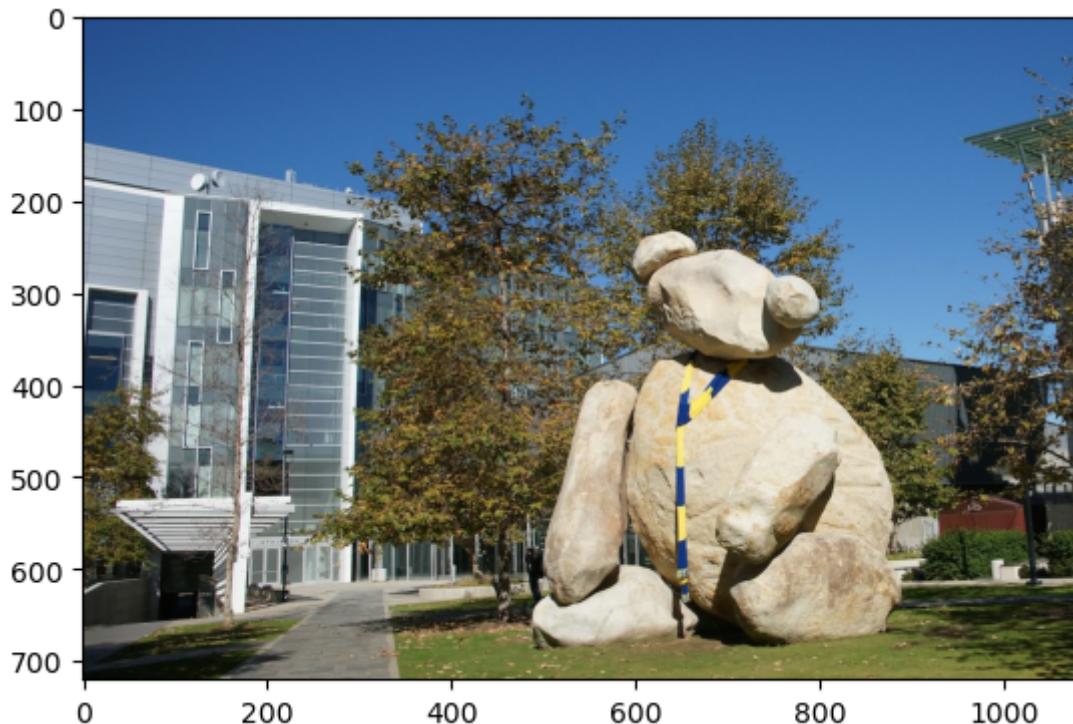
plt.imshow(img)
plt.show()
flipped_img = flip_img(img)
plt.imshow(flipped_img)
plt.show()
```

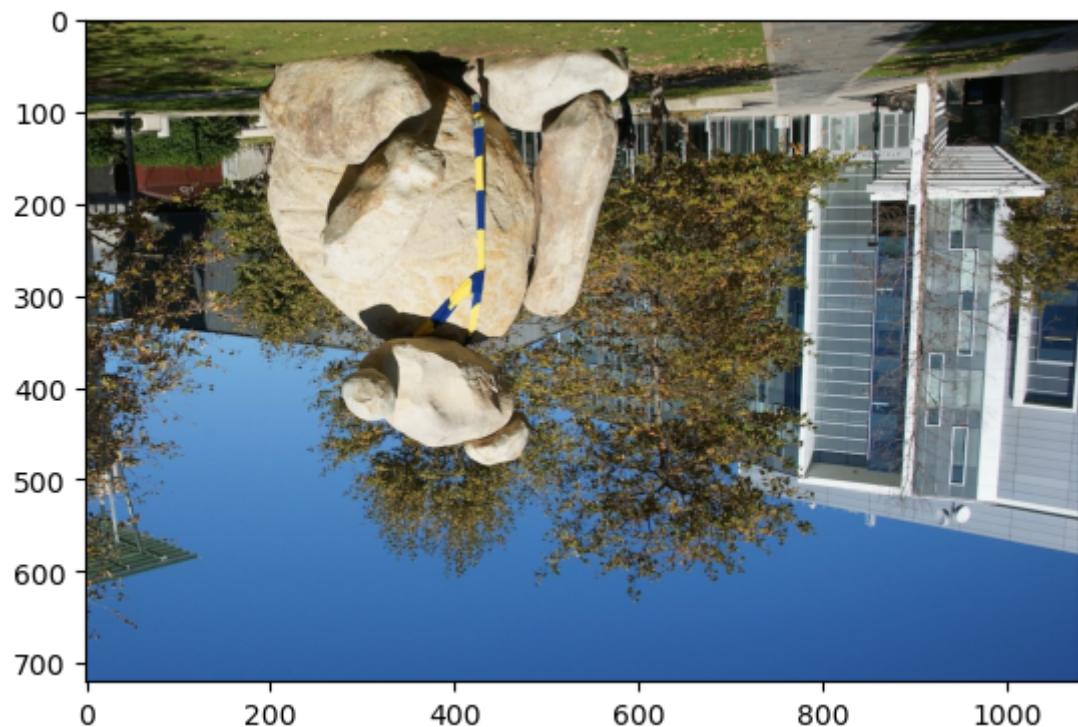
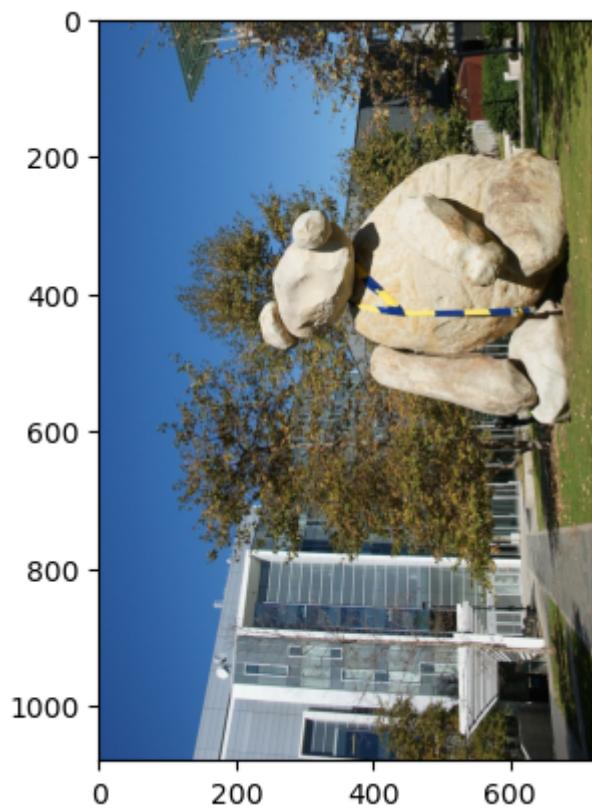


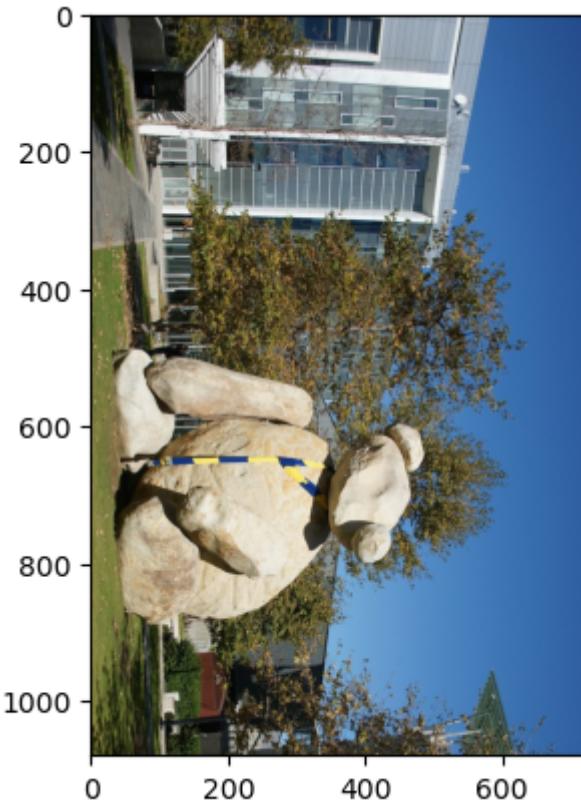
```
In [6]: # (3) Rotate image.
def rotate_90(img):
    """ Function to rotate image 90 degrees counter-clockwise.
    This function should return a W*H*3 array which is the rotated version of original.
    #### Write your code here. ####
    #flpimg = flip_img(img)
    # Like the cell before that we used flip_img function
    # here we read the
    # columns in reverse order and then each channel we
    # transpose it
    # (its like changing the rows with the columns and
    # this will rotate
    # the image 90 degrees to the left). We also use
    # a new array to save
    # the results and return it
    rotatedimg = img[:,::-1]
```

```
rotatedimg1 = np.zeros([img.shape[1],img.shape[0],3])
rotatedimg1[:, :, 0] = rotatedimg[:, :, 0].T
rotatedimg1[:, :, 1] = rotatedimg[:, :, 1].T
rotatedimg1[:, :, 2] = rotatedimg[:, :, 2].T
return rotatedimg1

plt.imshow(img)
plt.show()
rot90_img = rotate_90(img)
plt.imshow(rot90_img)
plt.show()
rot180_img = rotate_90(rotate_90(img))
plt.imshow(rot180_img)
plt.show()
rot270_img = rotate_90(rotate_90(rotate_90(img)))
plt.imshow(rot270_img)
plt.show()
```







```
In [38]: # (4)Read the face mask image and the face mask binary image
```

```
##### Write your code here. #####
```

```
# Reading the images the same way as before with the plt.imread
mask_img = plt.imread('images/face-mask.png')
bi_mask_img = plt.imread('images/face-mask-binary.png')

print("Face Mask Image Size: ")
print(mask_img.shape)
print("Face Mask Binary Mask Image Size: ")
print(bi_mask_img.shape)

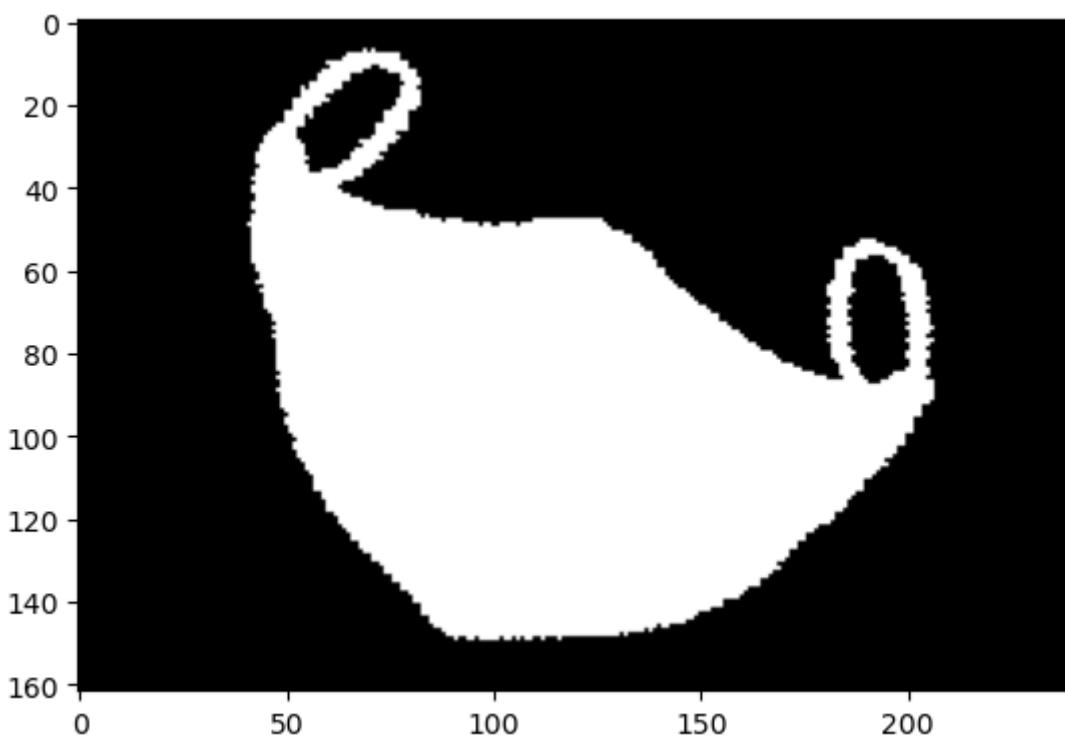
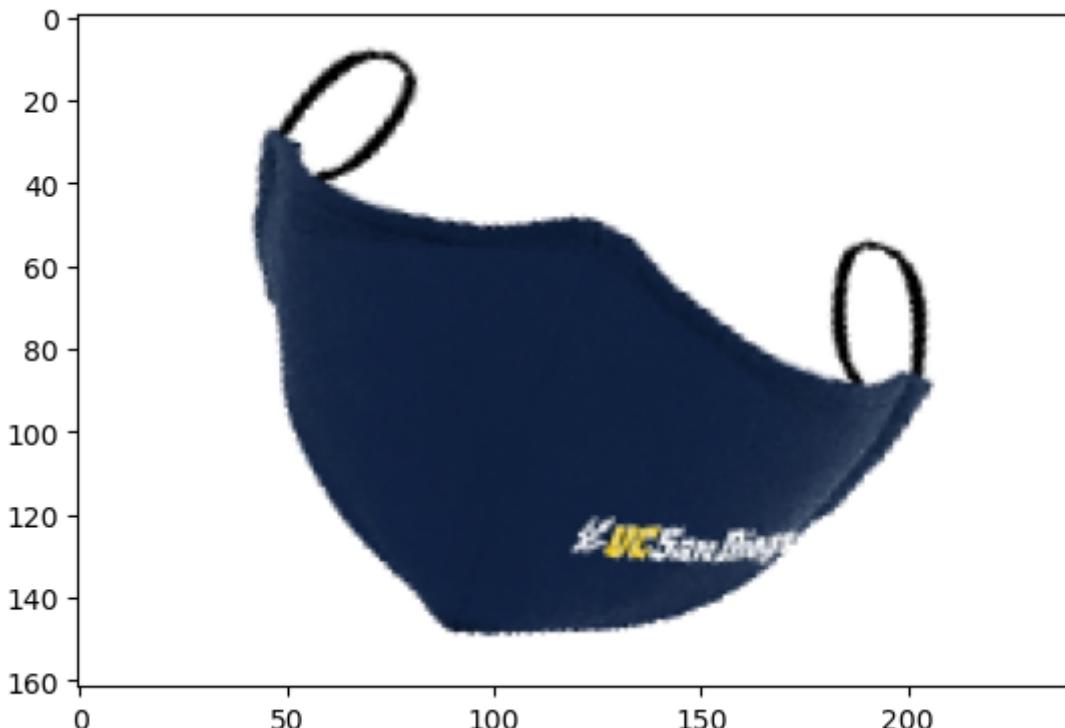
plt.imshow(mask_img)
plt.show()
plt.imshow(bi_mask_img)
plt.show()
```

```
Face Mask Image Size:
```

```
(162, 240, 3)
```

```
Face Mask Binary Mask Image Size:
```

```
(162, 240, 3)
```



```
In [48]: # (5) Put the face mask on the bear's face
start_x = 565
start_y = 240

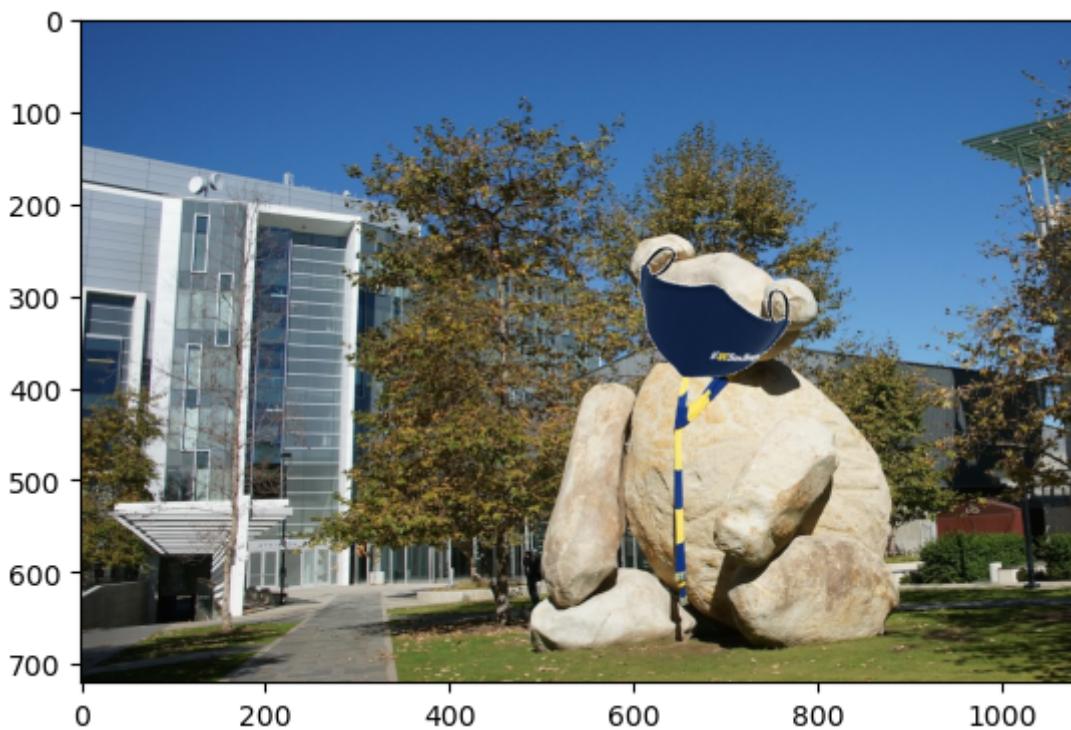
maskon_img = copy.deepcopy(img)

##### Write your code here. #####
# We use some helping variables so that we know where the indexing from maskon img
# corresponds to the mask image
tempx = 0
tempy = 0
flag = 0
for i in range(maskon_img.shape[0]):
    tempx = 0
    for y in range(maskon_img.shape[1]):
```

```
# We need to make sure that the indexes we are looking
# for are between the ranges of the mask image(shape)
if i >= start_y and i < start_y + mask_img.shape[0] \
and y >= start_x and y < start_x + mask_img.shape[1]:
    # We check the binary image to see the cells with true
    # value and then we change the bear image
    # in those specific cells with the cells from the mask image
    if bi_mask_img[tempy,tempx].any():
        maskon_img[i,y] = mask_img[tempy,tempx]
        tempx += 1
        flag = 1
    if (flag):
        tempy += 1
        flag = 0

plt.imshow(maskon_img)
```

Out[48]: <matplotlib.image.AxesImage at 0x2020440b820>

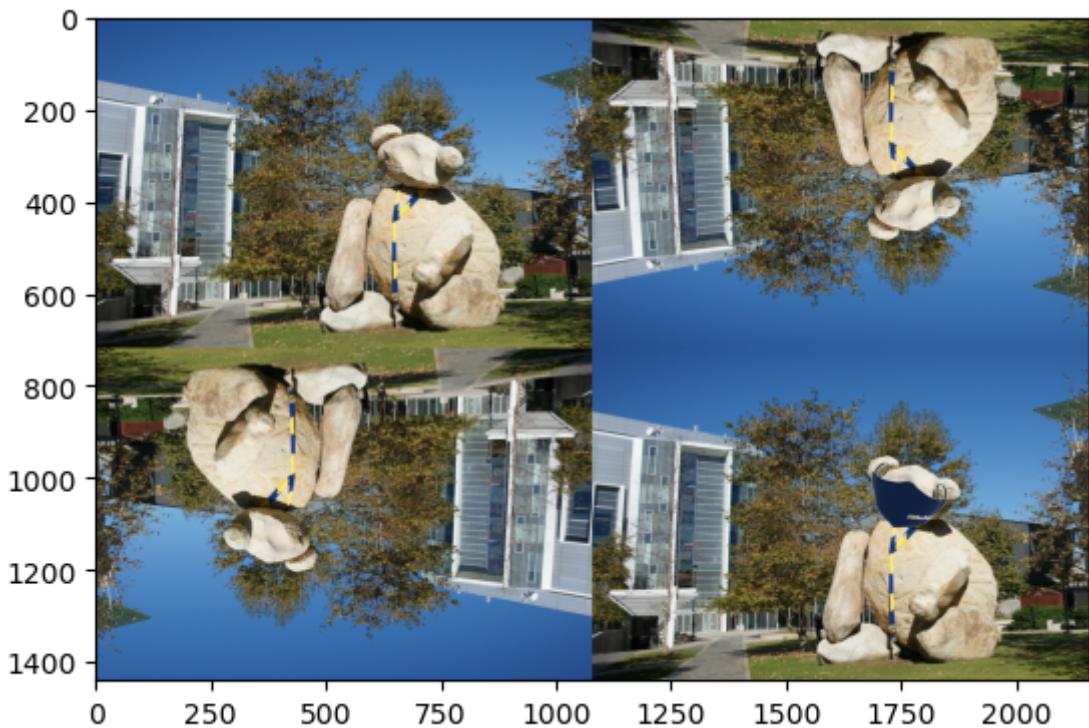


In [40]:

```
# (6) Write your code here to tile the four images and make a single image.
# You can use the img, flipped_img, rot180_img, maskon_img to represent the four images.
# After tiling, please display the tiled image.
##### Write your code here. #####
# We make a new image at first with all values zero but with the
# correct shape (2*H,2*W,3) and in each corner of the new image
# we put the tiles (the images we made)
a = img.shape[0]
b = img.shape[1]
final_img = np.zeros([2*a,2*b,3])
final_img[:a,:b] = img
final_img[:a,b:] = flipped_img
final_img[a:,:b] = rot180_img
final_img[a:,b:] = maskon_img

plt.imshow(final_img)
```

Out[40]: <matplotlib.image.AxesImage at 0x202042f1390>



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 \rightarrow Print... \rightarrow 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 \rightarrow Print... \rightarrow 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 \rightarrow Download as \rightarrow "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 \rightarrow Download as \rightarrow "LaTeX" file and then compile downloaded .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 \rightarrow Save and Export Notebook as \rightarrow "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).