**Mean**

|  |
| --- |
| **def** calculate\_mean**(**numbers**):**  S **=** **sum(**numbers**)**  N **=** **len(**numbers**)**  mean **=** S**/**N  **return** mean  donations**=** **[**100**,** 200**,** 60 **,**70 **,**50 **,**20**,** 80 **,**90**]**  mean\_value **=** calculate\_mean**(**donations**)**  **print(**'Trung bình số tiền quyên góp là: '**,** mean\_value) |
|  |

**Load image and blurring using mask-simple**

|  |
| --- |
| # load image and blurring using mask-simple  **import** numpy **as** np  **import** cv2  # load image in grayscale mode  image **=** cv2**.**imread**(**'mrbean.jpeg'**,**0**)**  # create kernel  kernel **=** np**.**ones**((**5**,**5**),** np**.**float32**)** **/** 25.0  # select ROI (top\_y, top\_x, height, width)  # image[y1:y2, x1:x2]  roi **=** image**[**22**:**225**,** 269**:**449**]**  # compute mean for each pixel  roi **=** cv2**.**filter2D**(**roi**,** cv2**.**CV\_8U**,** kernel**)**  image**[**22**:**225**,** 269**:**449**]** **=** roi  # show images  cv2**.**imshow**(**'roi'**,** roi**)**  cv2**.**imshow**(**'image'**,** image**)**  # waiting for any keys pressed and close windows  cv2**.**waitKey**(**0**)**  cv2**.**destroyAllWindows**()** |
|  |

**load image and blurring using face detection**

|  |
| --- |
| # load image and blurring using face detection  **import** numpy **as** np  **import** cv2  # face detection setup  face\_cascade **=** cv2**.**CascadeClassifier**(**cv2**.**data**.**haarcascades **+**'haarcascade\_frontalface\_default.xml'**)**  # load image in grayscale mode  image **=** cv2**.**imread**(**'mrbean.jpeg'**,**1**)**  # Convert to grayscale  gray **=** cv2**.**cvtColor**(**image**,** cv2**.**COLOR\_BGR2GRAY**)**  # face detection  faces **=** face\_cascade**.**detectMultiScale**(**gray**,** 1.1**,** 4**)**  # Create kernel  kernel **=** np**.**ones**((**7**,**7**),** np**.**float32**)** **/** 49.0  # Draw the retangle around each face  **for** **(**x**,**y**,**w**,**h**)** **in** faces**:**  cv2**.**rectangle**(**image**,** **(**x**,**y**),** **(**x**+**w**,** y**+**h**),** **(**0**,** 255**,** 0**),**1**)**  roi **=** image**[**y**:**y**+**h**,** x**:**x**+**w**]**    # compute mean for each pixel  roi **=** cv2**.**filter2D**(**roi**,** cv2**.**CV\_8U**,** kernel**)**  roi **=** cv2**.**filter2D**(**roi**,** cv2**.**CV\_8U**,** kernel**)**  roi **=** cv2**.**filter2D**(**roi**,** cv2**.**CV\_8U**,** kernel**)**    # update  image**[**y**:**y**+**h**,** x**:**x**+**w**]=**roi      # show images  cv2**.**imshow**(**'image'**,** image**)**  # waiting for any keys pressed and close windows  cv2**.**waitKey**(**0**)**  cv2**.**destroyAllWindows**()** |
|  |

**Median**

|  |
| --- |
| **def** calculate\_median**(**numbers**):**  N **=** **len(**numbers**)**  numbers**.**sort**()**  **if(**N**%**2 **==** 0**):**  m1 **=** N**/**2  m2 **=** **(**N**/**2**)+**1  m1 **=** **int(**m1**)** **-**1  m2 **=** **int(**m2**)** **-**1  median **=** **(**numbers**[**m1**]** **+**numbers**[**m2**])/**2  **else:**  m **=** **(**N**+**1**)/**2  m **=** **int(**m**)-**1  median **=** numbers**[**m**]**  **return** median    numbers **=** **[**8**,**4**,**1**,**2**,**5**,**8**]**  median\_value **=** calculate\_median**(**numbers**)**  **print(**'Median: '**,** median\_value**)** |
|  |

**Image denoising**

|  |
| --- |
| **import** numpy **as** np  **import** cv2  img1 **=** cv2**.**imread**(**'mrbean.jpeg'**)**  img2 **=** cv2**.**medianBlur**(**img1**,** 5**)**  # show images  cv2**.**imshow**(**'img1'**,** img1**)**  cv2**.**imshow**(**'img2'**,** img2**)**  # waiting for any keys pressed and close windows  cv2**.**waitKey**(**0**)**  cv2**.**destroyAllWindows**()**  median\_value **=** calculate\_median**(**numbers**)**  **print(**'Median: '**,** median\_value**)** |
|  |

**Variance**

|  |
| --- |
| # variance  **def** calculate\_mean**(**numbers**):**  S **=** **sum(**numbers**)**  N **=** **len(**numbers**)**  mean **=** S**/**N  **return** mean  **def** calculate\_variance**(**numbers**):**  mean **=** calculate\_mean**(**numbers**)**    diff **=** **[]**  **for** num **in** numbers**:**  diff**.**append**(**num**-**mean**)**    squared\_diff **=** **[]**  **for** d **in** diff**:**  squared\_diff**.**append**(**d**\*\***2**)**    sum\_squared\_diff**=sum(**squared\_diff**)**  variance **=** sum\_squared\_diff**/len(**numbers**)**    **return** variance  numbers **=** **[**5**,** 3**,** 6**,** 7**,** 4**]**  variance\_value **=** calculate\_variance**(**numbers**)**  **print(**'Variance: '**,** variance\_value**)** |
|  |

**Variance application to find texture of image**

|  |
| --- |
| **import** numpy **as** np  **import** cv2  **import** math  **from** scipy**.**ndimage**.**filters **import** generic\_filter  img **=** cv2**.**imread**(**'mrbean.jpeg'**)**  gray **=** cv2**.**cvtColor**(**img**,** cv2**.**COLOR\_BGR2GRAY**)**  cv2**.**imwrite**(**'edge\_s1.jpeg'**,** gray**)**  x**=** gray**.**astype**(**'float'**)**  x\_filt **=** generic\_filter**(**x**,** np**.**std**,** size **=** 7**)**  cv2**.**imwrite**(**'edge\_s2.jpeg'**,** x\_filt**)**  x\_filt**[**x\_filt **<** 20**]** **=** 0  cv2**.**imwrite**(**'edge\_s3.jpeg'**,** x\_filt**)**  maxv **=** np**.max(**x\_filt**)**  **print(**maxv**)**  x\_filt **=** x\_filt**\***2.5  cv2**.**imwrite**(**'edge\_s4.jpeg'**,** x\_filt**)** |
| edge\_s1.jpeg  edge\_s2.jpeg  edge\_s3.jpeg  edge\_s4.jpeg |

**Correlation coefficient**

|  |
| --- |
| **def** find\_corr\_x\_y**(**x**,**y**):**  n**=** **len(**x**)**  prod **=** **[]**  **for** xi**,** yi **in** **zip(**x**,**y**):**  prod**.**append**(**xi**\***yi**)**    sum\_prod\_x\_y **=sum(**prod**)**    sum\_x**=sum(**x**)**  sum\_y**=sum(**y**)**    Squared\_sum\_x **=** sum\_x**\*\***2  Squared\_sum\_y **=** sum\_y**\*\***2    x\_square **=** **[]**  **for** xi **in** x**:**  x\_square**.**append**(**xi**\*\***2**)**  x\_square\_sum **=** **sum(**x\_square**)**    y\_square **=** **[]**  **for** yi **in** y**:**  y\_square**.**append**(**yi**\*\***2**)**  y\_square\_sum **=** **sum(**y\_square**)**    # Use formula to calculate correlation  numerator **=** n**\***sum\_prod\_x\_y**-**sum\_x**\***sum\_y  denominator\_term1 **=** n**\***x\_square\_sum **-** Squared\_sum\_x  denominator\_term2 **=** n**\***y\_square\_sum **-** Squared\_sum\_y  denominator **=** **(**denominator\_term1**\*** denominator\_term2**)\*\***0.5  correlation **=** numerator**/**denominator    **return** correlation  x **=** **[**7**,**18**,**29**,**2**,**10**,**9**,**9**]**  y **=** **[**1**,**6**,**12**,**8**,**6**,**21**,**10**]**  corr\_x\_y **=**find\_corr\_x\_y**(**x**,** y**)**  **print(**'Correlation: '**,** corr\_x\_y**)** |
|  |

**Patch matching**

|  |
| --- |
| **import** numpy **as** np  **from** PIL **import** Image  **def** find\_corr\_x\_y**(**x**,**y**):**  n**=** **len(**x**)**  prod **=** **[]**  **for** xi**,** yi **in** **zip(**x**,**y**):**  prod**.**append**(**xi**\***yi**)**    sum\_prod\_x\_y **=sum(**prod**)**    sum\_x**=sum(**x**)**  sum\_y**=sum(**y**)**    Squared\_sum\_x **=** sum\_x**\*\***2  Squared\_sum\_y **=** sum\_y**\*\***2    x\_square **=** **[]**  **for** xi **in** x**:**  x\_square**.**append**(**xi**\*\***2**)**  x\_square\_sum **=** **sum(**x\_square**)**    y\_square **=** **[]**  **for** yi **in** y**:**  y\_square**.**append**(**yi**\*\***2**)**  y\_square\_sum **=** **sum(**y\_square**)**    # Use formula to calculate correlation  numerator **=** n**\***sum\_prod\_x\_y**-**sum\_x**\***sum\_y  denominator\_term1 **=** n**\***x\_square\_sum **-** Squared\_sum\_x  denominator\_term2 **=** n**\***y\_square\_sum **-** Squared\_sum\_y  denominator **=** **(**denominator\_term1**\*** denominator\_term2**)\*\***0.5  correlation **=** numerator**/**denominator    **return** correlation  # load ảnh và chuyển về kiểu list  image1 **=** Image**.open(**'image1.png'**)**  image2 **=** Image**.open(**'image2.png'**)**  image3 **=** Image**.open(**'image3.png'**)**  image1\_list **=** np**.**asarray**(**image1**).**flatten**().**tolist**()**  image2\_list **=** np**.**asarray**(**image2**).**flatten**().**tolist**()**  image3\_list **=** np**.**asarray**(**image3**).**flatten**().**tolist**()**  # tính correlation coefficient  corr\_1\_2 **=** find\_corr\_x\_y**(**image1\_list**,** image2\_list**)**  corr\_1\_3 **=** find\_corr\_x\_y**(**image1\_list**,** image3\_list**)**  **print(**'corr\_1\_2:'**,** corr\_1\_2**)**  **print(**'corr\_1\_3:'**,** corr\_1\_3**)** |
| **C:\Users\ADMIN\hoc python\image1.PNG**  Image1  **C:\Users\ADMIN\hoc python\image2.PNG**  Image2  C:\Users\ADMIN\hoc python\image3.PNG  Image3 |

**Range**

|  |
| --- |
| **def** find\_range**(**numbers**):**  lowest **=** **min(**numbers**)**  highest **=** **max(**numbers**)**  r **=** highest**-**lowest  **print(**'Lowest: {0}\tHighest: {1}\tRange: {2}'**.format(**lowest**,** highest**,** r**))**    # data  points **=** **[**7**,**8**,**9**,**2**,**10**,**9**,**9**,**9**,**9**,**1**,**5**,**6**,**7**]**  find\_range**(**points**)** |
|  |