			l 0			Obside Handling	
Commands:	int\str\		Operations:	add (s,	٥)	String Handling: string[1:3]	tr
type(x) isinstance(x,class)		•	x + y	add(s, radd (s,		string[::-1]	gnirts
print x, y, z	,		a + b	concat (s,		string[:-1]	strin
len(x)	len (se	1f)	x - y	sub(s,		string[-3:]	ing
str(x)	convert to		-y	neg(s)	, ,	string[:-3]	str
int(x)	convert to	-	x * y	mul (s,		string[-1]	d a
float(x)	convert to	_	x **v			string.find('s')	Ő
tuple(x)	convert to	tuple	х / у	div(s,		string.upper()	STRING
dict(x)	convert to di	ctionary	х % у	mod(s,		string.lower()	string
raw_input('text')		х < у	lt(s,)	string.replace('str',		
abs(x)abs(self)		х <= у	le(s,		string.startswith(x)	T\F	
global x	make x glo	bal	х > у	gt(s,c		'-'.join([a,b,c])	'a-b-c'
max(x,y,key=f)			x >= y	ge(s,c		string.split() list string.isalpha() is	t split by '_' s letter?
min(x,y.key=f) cmp(x,y)	x <y:-1, x="=y</td"><td>·0 v>v·1</td><td>x == y x != y</td><td>eq(s,c</td><td></td><td>2 1 17</td><td>it x's from end</td></y:-1,>	·0 v>v·1	x == y x != y	eq(s,c		2 1 17	it x's from end
ord(x)	ASCII # of	_	_	contains (s,	-	Random:	ac A B IIOM CHA
chr(x)	char of AS		x is y	Tuple	<i>-</i> ,	from random import ra	indom
op(enumerate(x))	(index,resu		x and y	Handling:		random() - 0-1	
zip([1,2,3],[4,5,6]			x or y	w = (x, y, z)		from random import ra	indint
sorted(items to be		,, (-,-,	not x	a,b,c = w		randint(x,y) - x-y	
comparator function		mparison	+=			from random import ch	
function, reverse =	= True\False)	_	-=	a=x, b=y, c	=z	choice('abcd') - rand	lom from string
List Handling:		Dictiona	ary Handling:	•	M	Math (Numpy):	
list = [a,b,c,w]			{k1:v1, k2:v2	•		mport numpy as np	
list[0]	a	dict[k1]	•	v0		np.log(8)/np.log(2)	3
list[0] = aa	[aa,b,c,w]	dict[k3]		v3		np.sqrt(x)	4-4//-
list[0:3:2]	[aa,c]		t(k, default)			np.loadtxt('file.txt',	<pre>delimiter=',')</pre>
list.insert(3, v) list.append(x) [$s_k = (k) \setminus k$ ems() = (k1, k)			np.int_(x) np.uint8(x)	
list.extend([y,z])		dict.key		V1), (K2, V2)]		np.arange(1,2.5,0.5)	[1,1.5,2]
list[0:6] =[t]	[t,y,z]	dict.val	•				[1,1.5,2,2.5,3]
list.remove(y)	[t,z]		p(k, default)			np.random.random intege	
copy = list[:]		dict.upc	date(dict2)		r	np.random.rand(2)	[0-1,0-1]
1 1 1	emove index x			s, returns su		Range Handling:	
list.reverse()			[1] index 1			range (2,5) [2,3,4]	
		7 / 1 1		1 / \			
list.sort()		list +	[x] same as	appena(x)		range $(0,5,2)$ $[0,2,4]$	
list.count(x) co	ount x in list	list +	[x] same as	appena(x)		range (0, 5, 2) [0, 2, 4] range (5, 0, -2) [5, 3, 1]	
list.count(x) co		list +			r	range (5, 0, -2) [5, 3, 1]	1)
list.count(x) co Array Handling: np.array([[1,2,3],			ax	is=0 a	r along	=	
<pre>list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2</pre>	[4,5,6]]) min\max values :skip]	within ar	ax ray ax	is=0 a is=1 a shape[0] H	along along Rows	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including)	
<pre>list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1)</pre>	[4,5,6]]) min\max values :skip] sum of column	within ar	ax ray ax x. x.	is=0 a is=1 shape[0] shape[1]	along along Rows Colum	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) uns (including)	Ė
<pre>list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x)</pre>	[4,5,6]]) min\max values :skip] sum of column of sum of entire of	within ar	ax ray ax x. x.	is=0 a is=1 a shape[0] I shape[1] (resize((height	along along Rows Colum	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) uns (including) cluding, width includir	Ė
<pre>list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1)</pre>	[4,5,6]]) min\max values :skip] sum of column of sum of entire of sort array	within ar or row array	ax ray ax x. x. x.	is=0 a is=1 a shape[0] I shape[1] cresize((height== y	along along Rows Colum t inc	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) uns (including) cluding, width including T,F,F]	Ė
<pre>list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1)</pre>	[4,5,6]]) min\max values :skip] sum of column of sum of entire of sort array) instance of	within ar or row array sorted ar	ax ray ax x. x. x. xray x[is=0 a is=1 a shape[0] I shape[1] (resize((height== y r,:]	along along Rows Colum t inc [T,F,	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) ans (including) cluding, width including T,F,F]	Ė
<pre>list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y)</pre>	[4,5,6]]) min\max values :skip] sum of column of sum of entire of sort array) instance of array of min vo	within ar or row array sorted ar alues of x	ax ray ax x. x. x. xray x[is=0	along along Rows Colum t inc [T,F, row r	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) ans (including) cluding, width including T,F,F]	Ė
<pre>list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y)</pre>	[4,5,6]]) min\max values :skip] sum of column of sum of entire of sort array) instance of array of min volumn of array of max volumn of	within ar or row array sorted ar alues of x alues of x	ax ray ax x. x. x. xray x[and y x[and y x. and y x.	is=0 a is=1 shape[0] I shape[1] cresize((height == y r,:] i;,c] any() a a	along along Rows Colum t inc [T,F, row r colum	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) ans (including) cluding, width including T,F,F]	Ė
<pre>list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y)</pre>	[4,5,6]]) min\max values :skip] sum of column of sum of entire of sort array) instance of array of min vo	within ar or row array sorted ar alues of x alues of x array	ax ray ax x. x. x. xray x[and y x[and y x[and y x. x]	is=0	along along Rows Colum t inc [T,F, row r colum are a	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) us (including) cluding, width including tr,F,F] an c ny true?	Ė
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x)	[4,5,6]]) min\max values :skip] sum of column of sum of entire sort array) instance of array of min values array of max values mean of entire array of mean of median of enti	within ar or row array sorted ar alues of x alues of x array of each co re array	ax ray ax x. x. x. xray x[and y x[and y x. (x) xl xx. (x) xx. xx. xx. xx. xx. xx. xx. xx. xx. x	is=0 is=1 shape[0] is=1 shape[1] resize((height == y r,:] :,c] any() ==0).any() all() anonzero()	along along along Rows Column t incolumn row roolumn are a does are a get t	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) uns (including) cluding, width including T,F,F] un c ny true? x contain 0? ll true? rue indices	ng))
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0)	[4,5,6]]) min\max values :skip] sum of column of sum of entire sort array) instance of array of min vo array of max vo mean of entire array of mean of median of enti- array of median	within ar or row array sorted ar alues of x alues of x array of each co re array	ax ray ax x. x. x. xray x[and y x[and y x. tand y x. tand y x. xx x	is=0 is=1 shape[0] shape[1] resize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y	along along along Rows Column t incolumn row roolumn are a does are a get tarray	range(5,0,-2) [5,3,1] r vertical (each column horizontal (each row) (including) with including reducing, width including reducing, with including reducing	ng)) n x and y
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y)	[4,5,6]]) min\max values :skip] sum of column sum of entire sort array) instance of array of min va array of max va mean of entire array of mean median of enti array of median y),axis=0\1)	within ar or row array sorted ar alues of x alues of x array of each co re array n of each	ax ray ax x. x. x. xray x[and y x[and y x. xl\row x. x. col\row x.	is=0 is=1 shape[0] shape[1] resize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y op a	along along Rows Colum t inc [T,F, row r colum are a does are a get t array	range(5,0,-2) [5,3,1] r vertical (each column horizontal (each row) (including) uns (including) width including that the column transfer in column true? x contain 0? ll true? rue indices of op results between of op results with a	ng)) n x and y over x
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v)	[4,5,6]]) min\max values :skip] sum of column of sum of entire sort array) instance of array of min values array of max values median of entire array of median y),axis=0\1) diagonal of vel	within ar or row array sorted ar alues of x alues of x array of each co re array n of each	ax ray ax x. x. xray x[and y x[and y x. tand y x. tand y x. tand y x. col\row x. xx	is=0 is=1 shape[0] Fresize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y op a mask] = c	along along Rows Colum t inc [T,F, row r colum are a does are a get t array array	range(5,0,-2) [5,3,1] r vertical (each column horizontal (each row) (including) with including relating, width including relating relatin	ng)) n x and y over x ange to c
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T	[4,5,6]]) min\max values :skip] sum of column of sum of entire of array of min volume of array of max volume median of entire array of median y),axis=0\1) diagonal of vec Transpose	within ar or row array sorted ar alues of x alues of x array of each co re array n of each	ax ray ax x. x. x. xray x[and y x[and y x. xl xl xl xcol\row x. xl xl xl	is=0 is=1 shape[0] Ishape[1] resize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y op a mask] = c (x>a)&(x <b)]=0< td=""><td>along along Rows Column trocolumn are a does aget trarray array</td><td>range(5,0,-2) [5,3,1] r vertical (each column horizontal (each row) (including) and (includin</td><td>n x and y over x ange to c , change to c</td></b)]=0<>	along along Rows Column trocolumn are a does aget trarray array	range(5,0,-2) [5,3,1] r vertical (each column horizontal (each row) (including) and (includin	n x and y over x ange to c , change to c
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b]	[4,5,6]]) min\max values :skip] sum of column of sum of entire of array of min volume of array of max volume median of entire array of mean of median of entire array of median y),axis=0\1) diagonal of vec Transpose value at coord.	within ar or row array sorted ar alues of x alues of x array of each co re array n of each ctor inate	ax ray ax x. x. x. xray x[and y x[and y x. xl xl xl xcol\row x. xl xl xl xl xl	is=0 is=1 shape[0] is=1 shape[1] resize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y op a mask] = c (x>a)&(x <b)]=0 c<="" td="" x!="a]="><td>alongalong Rows Columt ince [T,F, row recolumn are a does are a get t array array</td><td>range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) with including reluding, width including reluding, width including reluding reluding reluding reluding reluding reluding reluding reludices relucions relucio</td><td>ng)) n x and y over x ange to c , change to c ange to c</td></b)]=0>	alongalong Rows Columt ince [T,F, row recolumn are a does are a get t array array	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) with including reluding, width including reluding, width including reluding reluding reluding reluding reluding reluding reluding reludices relucions relucio	ng)) n x and y over x ange to c , change to c ange to c
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b]	[4,5,6]]) min\max values :skip] sum of column of sum of entire of array of min volume of array of max volume median of entire array of median y),axis=0\1) diagonal of vec Transpose	within ar or row array sorted ar alues of x array of each co re array n of each ctor inate of y	ax ray x. x. x. x. cray x[and y x[and y x. x] xl row x. col\row x x[x	is=0 is=1 shape[0] Ishape[1] resize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y op a mask] = c (x>a)&(x <b)]=c c="np.zeros_lil</td" x!="a]="><td>along along along Rows Column tinc [T,F, row r column are a does are a get tarray array ke(y)</td><td>range(5,0,-2) [5,3,1] r vertical (each column horizontal (each row) (including) and (including) and (including) are found including. With including true? In c ny true? x contain 0? ll true? rue indices rof op results between of op results with a there mask is True, chawhere condition is met where value not a, chawhere value not a, chawhere value not a, chawhere condition is met</td><td>ng)) n x and y over x ange to c , change to c ange to c y of y</td></b)]=c>	along along along Rows Column tinc [T,F, row r column are a does are a get tarray array ke(y)	range(5,0,-2) [5,3,1] r vertical (each column horizontal (each row) (including) and (including) and (including) are found including. With including true? In c ny true? x contain 0? ll true? rue indices rof op results between of op results with a there mask is True, chawhere condition is met where value not a, chawhere value not a, chawhere value not a, chawhere condition is met	ng)) n x and y over x ange to c , change to c ange to c y of y
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y)	[4,5,6]]) min\max values :skip] sum of column of sum of entire of sort array) instance of array of min values of max values array of mean of entire array of mean of entire array of median of entire array of median of entire array of median of vectors of the state	within ar or row array sorted ar alues of x array of each co re array n of each ctor inate of y	ax ray x. x. x. x. cray x[and y x[and y x. col\row x.	is=0 is=1 shape[0] Ishape[1] resize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y op a mask] = c (x>a)&(x <b)]=c c="np.zeros_lil</td" x!="a]="><td>along along along Rows Column tinc [T,F, row race a does are a get tarray array ke(y), e(y,v)</td><td>range(5,0,-2) [5,3,1] vertical (each column horizontal (each row)) (including) ins (including) cluding, width including T,F,F] in c ny true? x contain 0? ll true? rue indices of op results between of op results with a where condition is met where value not a, cha make x a zero-copy</td><td>ng)) n x and y over x ange to c , change to c ange to c y of y nd fill with val</td></b)]=c>	along along along Rows Column tinc [T,F, row race a does are a get tarray array ke(y), e(y,v)	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row)) (including) ins (including) cluding, width including T,F,F] in c ny true? x contain 0? ll true? rue indices of op results between of op results with a where condition is met where value not a, cha make x a zero-copy	ng)) n x and y over x ange to c , change to c ange to c y of y nd fill with val
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.	[4,5,6]]) min\max values :skip] sum of column sum of entire sort array) instance of array of min values array of mean median of entire array of median y),axis=0\1) diagonal of ver Transpose value at coord make x a copy dot product of pyplot as plt	within ar or row array sorted ar alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url	ax ray ax x. x. x. xray x[and y x[and y x. xray x. xx x	is=0 is=1 shape[0] Final Property of the interval of the inter	along along Rows Colum t inc [T,F, row r colum are a does are a get t array array c v ke(y) e(y,v Mont	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) uns (including) uluding, width including the contain 0? Il true? rue indices r of op results between r of op results with a where mask is True, character where value not a, character value in the condition is met where value not a, character value in the condition is met where value not a, character value in the condition is met condition is met range (5,0,-2) [5,3,1] vertical (each column the condition)	n x and y over x ange to c , change to c ange to c y of y nd fill with val
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.g x = np.linspace(-ng	[4,5,6]]) min\max values :skip] sum of column of entire entir	within ar or row array sorted ar alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op	ax ray ax x. x. x. xray x[and y x[and y x. xray x. xx x	is=0 is=1 shape[0] Final Property of the interval of the inter	along along Rows Colum t inc [T,F, row r colum are a does are a get t array array c v ke(y) e(y,v Mont from	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) uns (including) cluding, width including T,F,F] on c ny true? x contain 0? ll true? rue indices of op results between of op results with a chere mask is True, chewhere condition is met where value not a, chewhere value not a,	ng)) n x and y over x ange to c , change to c ange to c y of y nd fill with val
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.p x = np.linspace(-npy,z = np.cos(x),np	[4,5,6]]) min\max values :skip] sum of column of entire entir	within ar or row array sorted ar alues of x alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read	ax ray ax x. x. x. xray x[and y x[and y x. x] col\row x. x[x]	is=0 is=1 shape[0] Final Property of the interval of the inter	along along Rows Column t inc [T,F, row r column are a does are a get t array array ke(y) e(y,v Mont from n = circ	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) uns (including)	ng)) n x and y over x ange to c , change to c ange to c y of y nd fill with val
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.put x = np.linspace(-npy,z = np.cos(x),nppplt.plot(x,y)	[4,5,6]]) min\max values :skip] sum of column of entire entir	within ar or row array sorted ar alues of x alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read for line	ax ray ax x.	is=0 is=1 shape[0] Final Property of the interval of the inter	alongalong Rows Columt incolumn are a does are a get t tarray array c w (y) e(y,v) Mont	range(5,0,-2) [5,3,1] revertical (each column horizontal (each row) (including) and (includin	ng)) n x and y over x ange to c , change to c ange to c y of y nd fill with val n , 0.5), 0.5)
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.y x = np.linspace(-ny y,z = np.cos(x),np plt.plot(x,z)	[4,5,6]]) min\max values :skip] sum of column of entire entir	within ar or row array sorted ar alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read for line file.read	ax ray ax x.	is=0 is=1 shape[0] Final Property of the interval of the inter	along along Rows Colum tinc [T,F, row r colum are a does are a get t array array c v Mont from n = circ cour poir	range(5,0,-2) [5,3,1] revertical (each column horizontal (each row) (including) and (includin	ng)) n x and y over x ange to c , change to c ange to c y of y nd fill with val n , 0.5), 0.5)
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.y x = np.linspace(-ny y,z = np.cos(x),np plt.plot(x,y) plt.plot(x,z) plt.show()	[4,5,6]]) min\max values :skip] sum of column of entire entir	within ar or row array sorted ar alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read for line file.read file.read	ax ray ax x.	is=0 is=1 shape[0] Final Property of the interval of the inter	along along Rows Colum t ince [T,F, row r colum are a does are a get t array array c v Mont from n = circ cour poir i ir	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) uns (includ	nx and y over x ange to c , change to c ange to c y of y d fill with val , 0.5), 0.5) random()) for
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.y x = np.linspace(-ny y,z = np.cos(x),np plt.plot(x,z)	[4,5,6]]) min\max values :skip] sum of column of sum of entire as sort array) instance of array of min values of max values array of mean of entire array of median of entire array of median of entire array of median of vertical array of median of vertical value at coord make x a copy dot product of pyplot as plt p.pi,np.pi) .sin(x)	within ar or row array sorted ar alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read for line file.read	ax ray ax x. x. x. x. ray x[and y x[and y x. x] x] x[x[x[x[x[x[x] x] in file lline() llines() ee()	is=0 is=1 shape[0] Final Property of the interval of the inter	along along Rows Colum t ince [T,F, row r colum are a does are a get t array array c v Mont from n = circ cour i irr	range(5,0,-2) [5,3,1] revertical (each column horizontal (each row) (including) and (includin	nx and y over x ange to c , change to c ange to c y of y d fill with val , 0.5), 0.5) random()) for
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.y x = np.linspace(-ny y,z = np.cos(x),np plt.plot(x,y) plt.plot(x,y) plt.show() Image Handling: import matplotlib.py from scipy import mi	[4,5,6]]) min\max values :skip] sum of column of entire sort array) instance of array of min values array of max values mean of entire array of mean of median of entire array of median y),axis=0\1) diagonal of vertical value at coord make x a copy of the value at coord make x a copy of the value at coord make x a copy of the value at coord pyplot as plt p.pi,np.pi) .sin(x)	within ar or row array sorted ar alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read for line file.read file.read file.clos file.writ '\n' - ne	ax ray ax x.	is=0 is=1 shape[0] Final Property of the interval of the inter	along along along Rows Colum t inc [T,F, row r colum are a does are a get t array array from n = circ cour poir i ir cour circ	range(5,0,-2) [5,3,1] revertical (each column horizontal (each row) (including) ins (including) ins (including) ins (including) cluding, width including T,F,F] in c iny true? x contain 0? ll true? rue indices of op results between of op results with a where mask is True, cha where condition is met where value not a, cha make x a zero-copy ral) make x y-shaped ar ce Carlo Pi Calculation a random import random 10000 cle = Circle(Point(0.5, nt = 0 tts = [Point(random(), n range(n)] nt = sum([1 for p in point)	nx and y over x ange to c , change to c ange to c y of y d fill with val , 0.5), 0.5) random()) for
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.py r = np.cos(x), np plt.plot(x,z) plt.show() Image Handling: import matplotlib.py from scipy import mi image = misc.imread	[4,5,6]]) min\max values :skip] sum of column of sum of entire of sort array) instance of array of min values of array of max values of entire array of mean of entire array of median of entire array of median of entire array of median of vertical product of make x a copy dot product of pyplot as plt p.pi,np.pi) .sin(x) yplot as plt isc ('pic.jpg')	within ar or row array sorted ar alues of x alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read for line file.read file.clos file.writ '\n' - ne Show Image	ax ray ax x. x. x. x. x. rray x[and y x[and y x. col\row x. x. x. x. x. l\row x. col\row x. x[x[x[x[x[x] in file cline() clines() e() e(str) w w line :	is=0 is=1 shape[0] Final Property of the interval of the inter	along along along Rows Colum t inc [T,F, row r colum are a does are a get t array array from n = circ cour poir i ir cour circ	range(5,0,-2) [5,3,1] respectively (each column horizontal (each row) (including) and (includ	nx and y over x ange to c , change to c y of y nd fill with val n, 0.5), 0.5) random()) for oints if p in
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.y x = np.linspace(-ny y,z = np.cos(x),np plt.plot(x,z) plt.show() Image Handling: import matplotlib.py from scipy import mi image = misc.imread(misc.imsave('new.bmg	[4,5,6]]) min\max values :skip] sum of column of sum of entire of sort array) instance of array of min values of array of max values of entire array of mean of entire array of median of entire array of median of entire array of median of vertical product of make x a copy dot product of pyplot as plt p.pi,np.pi) .sin(x) yplot as plt isc ('pic.jpg')	within ar or row array sorted ar alues of x alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read for line file.read file.clos file.writ '\n' - ne Show Image plt.figure	ax ray ax x.	is=0 is=1 shape[0] shape[1] resize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y op a mask] = c (x>a)&(x <b)]=c ',="" 'w\r\a')<="" =="" c="np.zeros_lil" np.full_like="" td="" x!="a]=" x).read()=""><td>along along along Rows Colum t inc [T,F, row r colum are a does are a get t array array from n = circ cour poir i ir cour circ</td><td>range(5,0,-2) [5,3,1] revertical (each column horizontal (each row) (including) and (includin</td><td>nx and y over x ange to c , change to c y of y nd fill with val n, 0.5), 0.5) random()) for oints if p in</td></b)]=c>	along along along Rows Colum t inc [T,F, row r colum are a does are a get t array array from n = circ cour poir i ir cour circ	range(5,0,-2) [5,3,1] revertical (each column horizontal (each row) (including) and (includin	nx and y over x ange to c , change to c y of y nd fill with val n, 0.5), 0.5) random()) for oints if p in
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.py r = np.cos(x), np plt.plot(x,z) plt.show() Image Handling: import matplotlib.py from scipy import mi image = misc.imread	[4,5,6]]) min\max values :skip] sum of column of sum of entire of sort array) instance of array of min values of array of max values of entire array of mean of entire array of median of entire array of median of entire array of median of vertical product of make x a copy dot product of pyplot as plt p.pi,np.pi) .sin(x) yplot as plt isc ('pic.jpg')	within ar or row array sorted ar alues of x alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read for line file.read file.clos file.writ '\n' - ne Show Image plt.figure	ax ray ax x. x. x. x. x. rray x[and y x[and y x. col\row x. x. x. x. x. l\row x. col\row x. x[x[x[x[x[x] in file cline() clines() e() e(str) w w line :	is=0 is=1 shape[0] shape[1] resize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y op a mask] = c (x>a)&(x <b)]=c ',="" 'w\r\a')<="" =="" c="np.zeros_lil" np.full_like="" td="" x!="a]=" x).read()=""><td>along along along Rows Colum t inc [T,F, row r colum are a does are a get t array array from n = circ cour poir i ir cour circ</td><td>range(5,0,-2) [5,3,1] respectively (each column horizontal (each row) (including) and (includ</td><td>nx and y over x ange to c , change to c y of y nd fill with val n, 0.5), 0.5) random()) for oints if p in</td></b)]=c>	along along along Rows Colum t inc [T,F, row r colum are a does are a get t array array from n = circ cour poir i ir cour circ	range(5,0,-2) [5,3,1] respectively (each column horizontal (each row) (including) and (includ	nx and y over x ange to c , change to c y of y nd fill with val n, 0.5), 0.5) random()) for oints if p in
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.py x = np.linspace(-npy,z = np.cos(x), npplt.plot(x,y) plt.plot(x,y) plt.plot(x,y) plt.show() Tmage Handling: import matplotlib.py from scipy import mi image = misc.imread misc.imsave('new.bmg image.rotate(deg)	[4,5,6]]) min\max values :skip] sum of column sum of entire sort array) instance of array of min values array of mean median of entire array of median y),axis=0\1) diagonal of ver Transpose value at coord make x a copy dot product of pyplot as plt p.pi,np.pi) .sin(x) Vert	within ar or row array sorted ar alues of x alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read for line file.read file.clos file.writ '\n' - ne Show Image plt.figure plt.figure plt.fishow() ical Gradien	ax ray ax x. x. x. x. ray x[and y x[and y x. col\row x x[x	is=0 is=1 shape[0] shape[1] resize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y op a mask] = c (x>a)&(x <b)]=c ',="" 'w\r\a')="" =="" c="np.zeros_lil" np.full_like="" t.cm.gray)<="" td="" x!="a]=" x).read()=""><td>along along Rows Column tinc [T,F, row roolumn are a aloes are a get t array array when the cour poir i ir cour poir i ir cour prir</td><td>range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) us (including) us</td><td>n x and y over x ange to c , change to c ange to c y of y nd fill with val , 0.5), 0.5) random()) for oints if p in</td></b)]=c>	along along Rows Column tinc [T,F, row roolumn are a aloes are a get t array array when the cour poir i ir cour poir i ir cour prir	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) us	n x and y over x ange to c , change to c ange to c y of y nd fill with val , 0.5), 0.5) random()) for oints if p in
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.py r = np.cos(x),np plt.plot(x,y) plt.plot(x,z) plt.show() Image Handling: import matplotlib.py from scipy import mi image = misc.imread misc.imsave('new.bmg image.rotate(deg) Equation Solver: from scipy import lin	[4,5,6]]) min\max values :skip] sum of column of sum of entire of sort array) instance of array of min values of array of max values of entire array of mean of entire array of median of entire array of median of vertical of the value at coord make x a copy of dot product of pyplot as plt p.pi,np.pi) .sin(x) yplot as plt isc ('pic.jpg') p',image) Vertanly	within ar or row array sorted ar alues of x alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read for line file.read file.los file.writ '\n' - ne Show Image plt.figure plt.imshow plt.show() ical Gradiei = np.zeros	ax ray ax x.	is=0 is=1 shape[0] shape[1] resize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y op a mask] = c (x>a)&(x <b)]=c ',="" 'w\r\a')="" =="" c="np.zeros_lil" np.full_like="" t.cm.gray)="" t.cm.gray)<="" td="" x!="a]=" x).read()=""><td>alongalong Rows Column [T,F, row r column are a does are a get t tarray array Mont from n = circ cour poir i ir cour circ prir</td><td>range(5,0,-2) [5,3,1] revertical (each column horizontal (each row) (including) and (includin</td><td>nx and y over x ange to c , change to c , change to c y of y nd fill with val n, 0.5), 0.5) random()) for oints if p in cl,width</td></b)]=c>	alongalong Rows Column [T,F, row r column are a does are a get t tarray array Mont from n = circ cour poir i ir cour circ prir	range(5,0,-2) [5,3,1] revertical (each column horizontal (each row) (including) and (includin	nx and y over x ange to c , change to c , change to c y of y nd fill with val n, 0.5), 0.5) random()) for oints if p in cl,width
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.y x = np.linspace(-ny y,z = np.cos(x), np plt.plot(x,y) plt.plot(x,y) plt.plot(x,z) plt.show() Tmage Handling: import matplotlib.py from scipy import mi image = misc.imread misc.imsave('new.bmg image.rotate(deg) Equation Solver: from scipy import lin def solve_by_inverse([4,5,6]]) min\max values :skip] sum of column sum of entire sort array) instance of array of min values array of mean median of entire array of median y),axis=0\l) diagonal of ver Transpose value at coord make x a copy dot product of pyplot as plt p.pi,np.pi) .sin(x) vert pricalg cero (A, b): which will be selected to be selected with the selected to be selected to be selected with the selected to be selecte	within ar or row array sorted ar alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read for line file.read	ax ray ax x.	is=0 is=1 shape[0] shape[1] resize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y op a mask] = c (x>a)&(x <b)]=c ',="" 'w\r\a')="" =="" c="np.zeros_lil" np.full_like="" t.cm.gray)="" t.cm.gray)<="" td="" x!="a]=" x).read()=""><td>alongalong Rows Column [T,F, row r column are a does are a get t tarray array Mont from n = circ cour poir i ir cour circ prir</td><td>range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) ms (including) ms (including) ms (including) cluding, width including T,F,F] m c my true? x contain 0? 11 true? rue indices rof op results between rof op results with a shere mask is True, change where condition is met where condition is met where value not a, change are carlo Pi Calculation andom import random 10000 cle = Circle(Point(0.5, nt = 0 nts = [Point(random(), n range(n)] nt = sum([1 for p in point count * 4.0 / n Empty B&W Image: x = np.zeros((height incline), dtype=np.uint8) Get Neighbours: def neighbours(matrix, x return matrix[max(x-nx,</td><td>nx and y over x ange to c , change to c , change to c y of y nd fill with val n, 0.5), 0.5) random()) for oints if p in cl,width</td></b)]=c>	alongalong Rows Column [T,F, row r column are a does are a get t tarray array Mont from n = circ cour poir i ir cour circ prir	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) ms (including) ms (including) ms (including) cluding, width including T,F,F] m c my true? x contain 0? 11 true? rue indices rof op results between rof op results with a shere mask is True, change where condition is met where condition is met where value not a, change are carlo Pi Calculation andom import random 10000 cle = Circle(Point(0.5, nt = 0 nts = [Point(random(), n range(n)] nt = sum([1 for p in point count * 4.0 / n Empty B&W Image: x = np.zeros((height incline), dtype=np.uint8) Get Neighbours: def neighbours(matrix, x return matrix[max(x-nx,	nx and y over x ange to c , change to c , change to c y of y nd fill with val n, 0.5), 0.5) random()) for oints if p in cl,width
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.py r = np.cos(x),np plt.plot(x,y) plt.plot(x,z) plt.show() Image Handling: import matplotlib.py from scipy import mi image = misc.imread misc.imsave('new.bmg image.rotate(deg) Equation Solver: from scipy import lin	[4,5,6]]) min\max values :skip] sum of column sum of entire sort array) instance of array of min values array of mean median of entire array of median y),axis=0\l) diagonal of vec Transpose value at coord make x a copy dot product of pyplot as plt p.pi,np.pi) .sin(x) yplot as plt isc ('pic.jpg') p',image) Vert cero (A, b): shif (A).dot(b)	within ar or row array sorted ar alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read for line file.read	ax ray ax x. x. x. x. x. x. ray x[and y x[and y x. x] x.	is=0 is=1 shape[0] shape[1] resize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y op a mask] = c (x>a)&(x <b)]=c ',="" 'w\r\a')="" =="" c="np.zeros_lil" np.full_like="" t.cm.gray)="" t.cm.gray)<="" td="" x!="a]=" x).read()=""><td>alongalong Rows Column [T,F, row r column are a does are a get t tarray array Mont from n = circ cour poir i ir cour circ prir</td><td>range(5,0,-2) [5,3,1] revertical (each column horizontal (each row) (including) and (includin</td><td>nx and y over x ange to c , change to c ange to c y of y nd fill with val n , 0.5), 0.5) random()) for oints if p in cl,width x, y, nx=1, ny=1): 0) : min(x+nx+1,</td></b)]=c>	alongalong Rows Column [T,F, row r column are a does are a get t tarray array Mont from n = circ cour poir i ir cour circ prir	range(5,0,-2) [5,3,1] revertical (each column horizontal (each row) (including) and (includin	nx and y over x ange to c , change to c ange to c y of y nd fill with val n , 0.5), 0.5) random()) for oints if p in cl,width x, y, nx=1, ny=1): 0) : min(x+nx+1,
list.count(x) co Array Handling: np.array([[1,2,3], x.min\max() x[r1:r2:skip,c1:c2 x.sum(axis=0\1) np.sum(x) x.sort(axis=0\1) np.sort(x,axis=0\1) np.minimum(x,y) np.maximum(x,y) np.mean(x) x.mean(axis=0) np.median(x) x.median(axis=0) np.concatenate((x,y) np.diag(v) x.T x[a,b] x = np.copy(y) np.vdot(x,y) Graphs: import matplotlib.y x = np.linspace(-ny y,z = np.cos(x), np plt.plot(x,y) plt.plot(x,y) plt.plot(x,z) plt.show() Image Handling: import matplotlib.py from scipy import mi image = misc.imread misc.imsave('new.bmg image.rotate(deg) Equation Solver: from scipy import lin def solve_by_inverse(return linalg.inv([4,5,6]]) min\max values :skip] sum of column sum of entire sort array) instance of array of min values array of mean median of entire array of median y),axis=0\1) diagonal of ver Transpose value at coord make x a copy dot product of pyplot as plt p.pi,np.pi) .sin(x) yplot as plt isc ('pic.jpg') p',image) vert zero (A, b): (A).dot(b) diff imDy	within ar or row array sorted ar alues of x array of each co re array n of each ctor inate of y x and y File Hand url = url file = op file.read for line file.read file.read file.dos file.writ '\n' - ne Show Image plt.figure plt.imshow plt.show() ical Gradies = np.zeros t = np.concs = np.int_(s	ax ray ax x. x. x. x. x. x. ray x[and y x[and y x. x] x.	is=0 is=1 shape[0] shape[1] resize((height == y r,:] :,c] any() ==0).any() all() nonzero() op y op a mask] = c (x>a)&(x <b)]=c ',="" 'w\r\a')="" =="" c="np.zeros_lil" np.full_like="" t.cm.gray)="" t.cm.gray)<="" td="" x!="a]=" x).read()=""><td>alongalong Rows Column [T,F, row r column are a does are a get t tarray array Mont from n = circ cour poir i ir cour circ prir</td><td>range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) us (including) us</td><td>nx and y over x ange to c , change to c ange to c y of y nd fill with val n , 0.5), 0.5) random()) for oints if p in cl,width x, y, nx=1, ny=1): 0) : min(x+nx+1,</td></b)]=c>	alongalong Rows Column [T,F, row r column are a does are a get t tarray array Mont from n = circ cour poir i ir cour circ prir	range(5,0,-2) [5,3,1] vertical (each column horizontal (each row) (including) us	nx and y over x ange to c , change to c ange to c y of y nd fill with val n , 0.5), 0.5) random()) for oints if p in cl,width x, y, nx=1, ny=1): 0) : min(x+nx+1,

```
Optimal Cost w/ Memo:
                                                                        Squeeze Image:
                                                                       def squeeze_image(im, factor):
   __new_n = im.shape[0]
def optimal cost memo (N, prices, start=1):
   if start not in memo:
      min price = prices[start, N]
                                                                          new m = im.shape[1] / factor
            for i in range (N - 1, start, -1):
                                                                          new mat = np.zeros((new n,new m))
                                                                           for j in range(new mat.shape[1]):
            opt i = optimal cost memo (N, prices, i, memo)
                                                                              curr=range(j*factor,min((j+1)*factor,im.shape[1]))
            min_price = min(min_price, prices[start, i] + opt_i)
                                                                              new mat[:,j] = im[:,curr].mean(axis=1)
     memo[start] = min price
   return memo[start]
                                                                           return new mat
Condition:
                 Loops:
                                                Functions:
                                                                             Memoization:
                                                                                                           Greatest Common Divider:
                 for i in iterable:
if condition.
                                                def function(x,y):
                                                                             def \underline{\text{memo}} (x,y,mem=None):
                                                                                                           def GCD(x,v):
                                                                             ___if mem==None:
   statement
                    statement
                                                   body
                                                                                                           ___if y==0:
                                                                                   mem={}
elif.
                 \overline{\text{while}} expression:
                                                   return value
                                                                                                                 return x
                 ___statement
                                                                             ___if n not in mem:
                                                \overline{\text{def}} \ \underline{\text{rec}} \ (x,y):
                                                                                                              return GCD(y,x%y)
  statement
                                                \underline{\phantom{a}} if x == 0: #stop
                                                                                 rec(x-1,mem) + y
else:
                    count += 1
  _statement
                 ___break - exits loop
                                                      _return 0
                                                                                   mem[n]=value
                                                   return rec (x-1) + y
                    continue - skips cycle
                                                                                return mem[n]
Error Handling:
                              Object Handling:
                                                             Reminders:
raise errortype('text')
                              class classname:
                                                             - Define parameters before using them.
                              ___def __init__(self,x,y):
                                                             - Use functions from previous questions.
                                                             - Use class properties from previous questions.
- Be sure to use 'self' in method definition and calling.
ValueError
                                   self.a = x
                                    self.b = y
TypeError
ZeroDivisionError
                                 def method(self,z):
                                                             - Treat methods as functions (specific input and output).
                                  body
                                                             - When validating array entries: (0<=i<M).
                                   _return value
                                                             - Length\Height\Width are including, range is not.
                                def __repr__(self):
                                                             - Use range(len()) in loops if possible.
except exception as errl:
                                   return str(self.a)
                                                             - In loops, make new data and re-assign rather than edit.
  print err1.args[0]
                                def_str_(self)
                                                             - Use == for comparison, not =.
finally: #runs anyway
                                    return str(self.a)
                                                             - Note difference between arrays and matrix-like lists.
                                                             - Do not forget calling of non-returning functions when required.
                              object = classname(x,y)
                              object.a = w
                              object.method(z)
Search & Sort:
                                                             Distance:
Binary Search - in sorted list - if value<middle,
                                                             Hamming Distance - number of positions at which corresponding
search lower half of list.
                                                             symbols are different.
Bubble Sort - compare sequentially and swap if needed
                                                             QWERTY Distance - horizontal and vertical distances between keys on
Merge Sort - split in two until length of one, sort
                                                             keyboard representing each corresponding symbol.
                                                             Levenshtein Distance - number of edits (insert, delete or
each by parts, merge and sort.
<u>Counting Sort</u> - histogram and rebuild in order.
                                                             substitute) required to transform one word into another.
Selection Sort - find index of minimal\maximal value
in sublist and swap with start\end if needed.
                                                             Polynomial Handling:
                                                                                               Cumulative Sum:
Insertion Sort - assume sorted so far, insert in
                                                             import numpy.polynomial.
                                                                                               def cum sum(lst):
correct place and continue until end.
                                                             polynomial as poly
                                                                                               ___a = []
                                                                                               for i in range(len(lst)):
                                                             p1= poly.Polynomial([6,-5,1])
Bucket Sort - put each x into k buckets by [x/k], sort
each bucket and merge sequentially.
                                                                                                     a.append(sum(lst[:i])+lst[i])
Shell Sort - variant of Insertion; sort first objects
                                                             pl.coef, pl.roots(), pl(x)
                                                                                                  return a
in each half, divide by four, sort again and so on
until sorted.
Permutation:
                                           Bucket Sort:
                                                                                   Selection Sort:
def permute(elems):
                                           def bucket sort(lst):
                                                                                   def selection sort(lst):
                                              buckets=[]
  _if len(elems) == 0:
                                                                                     lst = lst[:]
     return [[]]
                                              for i in range(10):
                                                                                      for i in range(len(lst)):
                                                                                       _m = min_index(lst, i)
  perms = []
                                                 buckets.append([])
   for curr in elems:
                                                                                         lst[m], \overline{lst[i]} = lst[i], lst[m]
                                              for x in 1st:
     _elems_minus_curr = elems[:]
                                                 buckets[x / 10].append(x)
                                                                                      return 1st
      elems_minus_curr.remove(curr)
                                              sorted list = [ ]
                                                                                   Bubble Sort:
                                              for b in buckets:
      sub perms =
                                                                                   def bubble sort(lst):
permute (elems minus curr)
                                                 if len(b) > 0:
                                                                                      for i in range (len(lst)-1, 0, -1):
  ____for perm in sub perms:
                                                    b = selection sort(b)
                                                                                       ___for j in range(i):
                                                 sorted list.extend(b)
        perm.append(curr)
                                                                                          __if lst[j] > lst[j+1]:
     perms.extend(sub perms)
                                             return sorted list
                                                                                               lst[j], lst[j+1] = lst[j+1], lst[j]
   return perms
                                                                                                 Sort Dictionary Keys by Values:
Histogram:
                                 Binary Search:
                                                                Fibonacci w\ Memo:
                                 def bin_iter(key,vals):
                                                                def fib(n, mem=None):
def histogram(string):
                                                                                                 def tuc(tup):
                                                                ___if n<2:
                                                                                                   return tup[1]
  d = \{ \}
                                    n = \overline{len(vals)}
___for char in string:
                                                                                                 def keys_by_values(d):
                                     10 = 0
                                                                      return n
                                                                ___if mem == None:
     _count = d.get(char,0)
                                                                                                 ___items = list(d.items())
                                    up = n-1
                                                                                                    _sort_list=sorted(items,key=tuc)
     d[char] = count + 1
                                     while up >= low:
                                                                      mem = {} {}
                                       _mid = (up + lo) / 2
                                                                ___if n not in mem:
  return d
                                                                                                    sort keys = []
                                        if kev == vals[mid]:
                                                                                                 ___for item in sort_list:
                                                                      mem[n] = \
Anagram Check:
                                          return True
                                                                \overline{\text{fib}(n-1,\text{mem})} + \text{fib}(n-2,\text{mem})
                                                                                                      _sort_keys.append(item[0])
def anagrams(s1,s2):
                                        elif key< vals[mid]:</pre>
                                                                    return mem[n]
                                                                                                    return sorted_keys
if len(s1) != len(s2):
                                          _{\rm up} = mid - 1
                                                                Integrate Between 0 & 1:
     _return False
                                        else:
                                                                from scipy import integrate
   d1 = histogram(s1)
                                         __lo = mid + 1
                                                                integrate.quad(f,0,1)[0]
   d2 = histogram(s2)
                                    return False
   return d1 == d2
Hamming Distance:
                                       Levenstein Distance:
                                                                                   Reverse Digits:
def hamming dist(w1,w2):
                                       def lev(s1,s2):
                                                                                   def reverse digits(num):
                                         if len(s1) == 0 or len(s2) == 0:
  score = abs(len(w1) - len(w2))
                                                                                     if num==\overline{0}:
     for c1, c2 in zip(w1, w2):
                                             return max(len(s1),len(s2))
                                                                                        return 0
       __if c1 != c2:
                                          d1 = lev(s1[:-1], s2) + 1
                                                                                      d = len(str(num))-1
            score += 1
                                          d2 = lev(s1, s2[:-1]) + 1
                                                                                      last num = num%10
                                          last = 0 if s1[-1]==s2[-1] else 1
                                                                                      last num = last num*(10**d)
  return score
                                          d3 = lev(s1[:-1], s2[:-1]) + last
                                                                                      return last num+reverse digits(num/10)
                                          return min(d1,d2,d3)
```