>>> "She's 18"

"She's 18"

>>> 'She's 18'

SyntaxError: invalid syntax

>>> 'She\'s 19'

"She's 19"

>>> ' seh \sshh'

' seh \\sshh'

>>> print("shweta")

shweta

>>> print('shweta\ntanear')

shweta

tanear

>>> print(r'shweta\ntanear')

shweta\ntanear

>>> firstname="Shweta"

>>> lastname="Tanwar"

>>> firstname+lastname

'ShwetaTanwar'

>>> firstname\*5

'ShwetaShwetaShwetaShwetaShweta'

>>> firstname \* 5

'ShwetaShwetaShwetaShwetaShweta'

>>> firstname/5

Traceback (most recent call last):

File "<pyshell#17>", line 1, in <module>

firstname/5

>> firstname[1]

'h'

>>> firstname[-1]

'a'

>>> firstname[0:3]

'Shw'

>>> firstname[:4]

'Shwe'

>>>

age=27  
**if** age<21:  
 print(**"age more"**)  
**elif** age>=21:  
 print(**"go ahead and fuck"**)

**def** add\_numbers(\*args):  
 total=0  
 **for** i **in** args:  
 total+=i  
 print(**"total"**,total)  
  
 add\_numbers(1,2,3,4,5,6)  
  
 print(**"hello"**)

...  
**def** add\_numbers(\*args):  
 total=0  
 **for** i **in** args:  
 total+=i  
 print(**"total"**,total)  
  
add\_numbers(1,2,3,4,5)  
...  
print(**"hello"**)

**import** file\_func  
file\_func.get\_time()

**import** datetime  
  
**def** print\_name(x):  
 print(x)  
  
**def** get\_time():  
 x=datetime.datetime.now()  
 print(x)

**sets**

festivals={**"Holi"**,**"Diwali"**,**"Rakshabandhan"**,**"Diwali"**,**"Karwachauth"**,**"Christmas"**,**"Diwali"**}  
  
print(festivals)  
  
**if 'Diwali' in** festivals:  
 print(**"present"**)  
**else**:  
 print(**"not there"**)

**union in sets**

1)indian\_festivals={**"Holi"**,**"Diwali"**,**"Rakshabandhan"**,**"Diwali"**,**"Karwachauth"**,**"Christmas"**,**"Diwali"**}  
usa\_festivals={**"independence"**,**"memorial"**,**"christmas"**,**"thanksgiving"**}  
all\_festivals=indian\_festivals.union(usa\_festivals)  
  
print(all\_festivals)

2) *#import file\_func  
#file\_func.get\_time()*indian\_festivals={**"Holi"**,**"Diwali"**,**"Rakshabandhan"**,**"Diwali"**,**"Karwachauth"**,**"Christmas"**,**"Diwali"**}  
usa\_festivals={**"independence"**,**"memorial"**,**"christmas"**,**"thanksgiving"**}  
all\_festivals=indian\_festivals|usa\_festivals  
  
print(all\_festivals)

Empty curly braces {} will make an empty dictionary in Python. To make a set without any elements we use the set() function without any argument.

indian\_festivals[2]

--indexing doesn’t work for sets

**Pass arg as list**

**def** health\_calculator(age,apples\_ate,cigs\_smoked):  
 ans=(100-age)+(apples\_ate\*3.5)-(cigs\_smoked\*2)  
 print(**"your health"**,ans)

health\_data=[30,7,0]  
  
health\_calculator(\*health\_data)

health\_calculator(\*health\_data[1:3])  
health\_calculator(\*health\_data)

**Palindrome code**

t = 102  
i=0  
rev = []  
rev\_num=0

**while** t > 0:  
 i=t%10  
 rev.append(i)  
 t = int(t / 10)  
i=len(rev)  
**for** j **in** rev:  
 rev\_num+=j\*(10\*\*(i-1))  
 i-=1  
print(rev\_num)

**pass args splits**

**def** health\_calculator(age,apples\_ate,cigs\_smoked):  
 ans=(100-age)+(apples\_ate\*3.5)-(cigs\_smoked\*2)  
 print(**"your health"**,ans)  
  
  
  
health\_data=[30,7,0,27,5,1]  
  
  
  
health\_calculator(\*health\_data[3:6])

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Download image from url**

**import** random  
**import** urllib.request  
  
**def** download\_web\_image(url):  
 name=random.randrange(1,10)  
 fullname=str(name)+**".jpg"** urllib.request.urlretrieve(url,fullname)  
  
download\_web\_image(**"https://pixabay.com/en/image-statue-brass-child-art-1465348/"**)

**Read and write a text file on ur system**

fw=open(**'shweta.txt'**,**'w'**)  
fw.write(**'I am learning Phython\n'**)  
fw.write(**'today is 9th july'**)  
fw.close()

text=**''**fw=open(**'shweta.txt'**,**'r'**)  
text=fw.read()  
print(text)  
fw.close()

...  
 csv=response.read()  
 csv\_str=str(csv)  
 lines=csv\_str.split(**"\\n"**)  
 dest\_url=**r'goog.csv'** fx=open(dest\_url,**'w'**)  
 **for** line **in** lines:  
 fx.write(line+**"\n"**)  
...

fx.close()

**from** urllib **import** request  
goo\_url=**"https://people.sc.fsu.edu/~jburkardt/data/csv/addresses.csv"  
  
def** download\_files\_google(url):  
 response=request.urlopen(url)  
 csv = response.read()  
 csv\_str = str(csv)  
 lines = csv\_str.split(**"\\n"**)  
 *# dest\_url = r'goog.csv'* fx=open(**"down.csv"**,**'w'**)  
 **for** line **in** lines:  
 fx.write(line+**"\n"**)  
  
  
download\_files\_google(goo\_url)

var=int(input(**"Enter magic number"**))34  
print(var)

**try**:  
 print(0/0)  
**except**:  
 print(**"error"**)

**Program to accept the strings which contains all vowels**

x=str(input(**"Enter the string"**))  
a=[**"a"**,**"e"**,**"i"**,**"o"**,**"u"**]  
counter=0  
**for** i **in** a:  
 **if** i **in** x.lower():  
 counter+=1  
**if** counter==5:  
 print(**"accepted"**)  
**else**:  
 print(**"not accepted"**)  
print(counter

n = int(input())  
**if** n **in** range(1,100):  
 **if** (n%2 !=0 ):  
 print(**"Weird"**)  
 **elif** (n%2 == 0):  
 **if** n **in** range(2,5):  
 print(**"Not Weird"**)  
 **elif** n **in** range(6,20):  
 print(**"Weird"**)  
 **elif** n >20:  
 print(**"Not Weird"**)  
**else**:  
 print(**"Number should be between 1 and 100.Try Again..."**)

Read an integer . For all non-negative integers , print

a=int(input())  
**if** a **in** range(1,20):  
 i=0  
 **while** (i<a):  
 print(i\*\*2)  
 i+=1  
**else**:  
 print(**"Number should be between 1 and 20.Try again"**)

# List Comprehensions

X,Y,Z,N=map(int, input().split(**','**))  
x = [[i,j,k] **for** i **in** range(0,X+1) **for** j **in** range(0,Y+1) **for** k **in** range(0,Z+1) **if** (i+j+k) !=N]  
print (x)

n = 3  
arr = map(int, input().split( ))  
arr=list(arr)  
z=max(arr)  
tmp=[]  
**for** i **in** arr:  
 **if** i != z:  
 tmp.append(i)  
print(max(tmp))

grade=[[x,y] **for** x **in** name **for** y **in** score]  
print(grade)

Nested List

name\_list=[]  
score\_list=[]  
grade\_list=[]  
n=3  
**for** \_ **in** range(int(input())):  
 name = input()  
 score = float(input())  
 name\_list.append(name)  
 score\_list.append(score)  
**for** i **in** range(len(name\_list)):  
 grade\_list.append([name\_list[i],score\_list[i]])  
grade\_list.sort(key=**lambda** x: x[1])  
temp\_list=[grade\_list[1][0]]  
**for** i **in** range(2,len(name\_list)):  
 **if** grade\_list[i][1]==grade\_list[i-1][1]:  
 temp\_list.append(grade\_list[i][0])  
 **else**:  
 **break**temp\_list.sort()  
print(temp\_list)

name\_list=[]  
score\_list=[]  
grade\_list=[]  
n=3  
**for** \_ **in** range(int(input())):  
 name = input()  
 score = float(input())  
 name\_list.append(name)  
 score\_list.append(score)  
**for** i **in** range(len(name\_list)):  
 grade\_list.append([name\_list[i],score\_list[i]])  
grade\_list.sort(key=**lambda** x: x[1])  
temp\_list=[grade\_list[1][0]]  
**for** i **in** range(2,len(name\_list)):  
 **if** grade\_list[i][1]==grade\_list[i-1][1]:  
 temp\_list.append(grade\_list[i][0])  
  
 **elif** grade\_list[i][1]==grade\_list[i+1][1]:  
 temp\_list.append(grade\_list[i][2])  
 **else**:  
 **break**temp\_list.sort()  
print(temp\_list)

The str.split() method returns a list of strings that are separated by whitespace if no other parameter is given.

The str.replace() method can take an original string and return an updated string with some replacement.

The str.join() method is also useful to combine a list of strings into a new single string.

The functions str.upper() and str.lower() will return a string with all the letters of an original string converted to upper- or lower-case letters. Because strings are immutable data types, the returned string will be a new string. Any characters in the string that are not letters will not be changed.

**Convert first name and last name to capital**

s=**"shweta tanwar mhatre"**ind=[0]  
x=0  
while x != -1:  
 x = s.find(**' '**,x, len(s))  
 print(x)  
 ind.append(x+1)  
print(**""**.join(c.upper() if i in ind else c for i,c in enumerate(s)))

Set challenge

m,n=map(int, input().split())  
arr=list(map(int, input().split()))  
A=set(map(int, input().split()))  
B=set(map(int, input().split()))  
hp=0  
**for** i **in** arr:  
 **if** i **in** A:  
 hp=hp+1  
 **elif** i **in** B:  
 hp = hp-1  
print(hp)

print(**"{}\n{}"**.format(a//b, a/b))

print(int(a/b),(a/b),sep=**'\n'**)

**print list elements without comma**

n = int(input())  
num\_list=[]  
**for** c **in** range(1,n+1):  
 num\_list.append(c)  
print(\*num\_list,sep=**''**)

Tuples are immutable, which means you cannot update or change the values of tuple elements. You are able to take portions of the existing tuples to create new tuples

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the **del** statement

Hash function on tuple

**n = int(input())**

**integer\_list = map(int, input().split())**

**print(hash(tuple(integer\_list)))**

**Wrap text**

**import** textwrap  
**def** wrap(string, max\_width):  
 **return** textwrap.fill(string, width=max\_width)  
  
print(wrap(**'ABCDEFGH'**,3))

**swap lower to upper and vice versa**

*#def swap\_case(s):*s=**"American Comp"  
def** swap\_case(s):  
 **return** (**""**.join( i.upper() **if** i.islower() **else** i.lower() **for** i **in** s))  
  
print(swap\_case(s))

**Split and join**

**def** split\_and\_join(line):  
 **return** (**'-'**.join(line.split( )))  
  
print(split\_and\_join(**"htis is a bffj"**))

**Print full name**

**def** print\_full\_name(a, b):  
 print(**"Hello {} {}! You just delved into python."**.format(a,b))

**def** mutate\_string(string, position, character):  
 **return** string[:position] + character + string[position+1:]  
  
print(mutate\_string(**"abracadabra"**,5,**'k'**))

lists are mutable (they can be changed), and tuples are immutable (they cannot be changed).

Strings are immutable. This way we can modify them.

str = raw\_input()

print any(c.isalnum() for c in str)

print any(c.isalpha() for c in str)

print any(c.isdigit() for c in str)

print any(c.islower() for c in str)

print any(c.isupper() for c in str)

**def** domainGet(x):  
 print(x[x.find(**'@'**) + 1: ])  
  
domainGet(**'user@domain.com'**)

**def** findDog(x):  
 print(x.find(**'dog'**)!= -1)

seq = [**'soup'**,**'dog'**,**'salad'**,**'cat'**,**'great'**]  
print(list(filter(**lambda** x:x[0] == **'s'**,seq)))

### Final Problem

**You are driving a little too fast, and a police officer stops you. Write a function to return one of 3 possible results: "No ticket", "Small ticket", or "Big Ticket". If your speed is 60 or less, the result is "No Ticket". If speed is between 61 and 80 inclusive, the result is "Small Ticket". If speed is 81 or more, the result is "Big Ticket". Unless it is your birthday (encoded as a boolean value in the parameters of the function) -- on your birthday, your speed can be 5 higher in all cases.**

**def** caught\_speeding(speed, is\_birthday):  
 **if** is\_birthday:  
 print( **"No Ticket" if** speed <65 **else "Small ticket" if** speed **in** range(66,85) **else "Big Ticket"** )  
 **else**:  
 print(**"No Ticket" if** speed < 61 **else "Small ticket" if** speed **in** range(61, 80) **else "Big Ticket"**)  
  
caught\_speeding(81,**False**)

planet = "Earth"

diameter = 12742

print("The diameter of {} is {}".format(planet,diameter))

### arange

Return evenly spaced values within a given interval.

In [22]:



np.arange(0,10)

Out[22]:

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [23]:



np.arange(0,11,2)

Out[23]:

array([ 0, 2, 4, 6, 8, 10])

### zeros and ones

Generate arrays of zeros or ones

In [24]:



np.zeros(3)

Out[24]:

array([ 0., 0., 0.])

In [26]:



np.zeros((5,5))

Out[26]:

array([[ 0., 0., 0., 0., 0.],

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

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

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

[ 0., 0., 0., 0., 0.]])

In [27]:



np.ones(3)

Out[27]:

array([ 1., 1., 1.])

In [28]:



np.ones((3,3))

Out[28]:

array([[ 1., 1., 1.],

[ 1., 1., 1.],

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

### linspace[¶](http://localhost:8888/notebooks/Python1/Python-Data-Science-and-Machine-Learning-Bootcamp/Python-for-Data-Analysis/NumPy/NumPy%20Arrays.ipynb" \l "linspace)

Return evenly spaced numbers over a specified interval.

In [29]:



np.linspace(0,10,3)

Out[29]:

array([ 0., 5., 10.])

## eye

Creates an identity matrix

In [37]:



np.eye(4)

Out[37]:

array([[ 1., 0., 0., 0.],

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

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

[ 0., 0., 0., 1.]])

## Random

Numpy also has lots of ways to create random number arrays:

### rand

Create an array of the given shape and populate it with random samples from a uniform distribution over [0, 1).

### randn

Return a sample (or samples) from the "standard normal" distribution. Unlike rand which is uniform:

### randint

Return random integers from low (inclusive) to high (exclusive).

arr=np.ones(10)

arr

barr=arr\*5

barr

#### Create an array of the integers from 10 to 50[¶](http://localhost:8888/notebooks/Python1/Python-Data-Science-and-Machine-Learning-Bootcamp/Python-for-Data-Analysis/NumPy/Numpy%20Exercise%20.ipynb#Create-an-array-of-the-integers-from-10-to-50)

varr=np.arange(10,51)

varr

#### Create an array of all the even integers from 10 to 50

varr=np.arange(10,51,2)

varr

#### Create a 3x3 matrix with values ranging from 0 to 8

varr=np.arange(0,9).reshape(3,3)

varr

#### Create a 3x3 identity matrix

arr=np.eye(3,3)

arr

#### Use NumPy to generate a random number between 0 and 1

arr=np.random.rand(1)

arr

#### Use NumPy to generate an array of 25 random numbers sampled from a standard normal distribution

arr=np.random.randn(25)

arr

#### Create the following matrix:[¶](http://localhost:8888/notebooks/Python1/Python-Data-Science-and-Machine-Learning-Bootcamp/Python-for-Data-Analysis/NumPy/Numpy%20Exercise%20.ipynb" \l "Create-the-following-matrix:)

In [35]:



arr**=**np.arange(0.01,1.01,0.01).reshape(10,10)

arr

Out[35]:

array([[ 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ],

[ 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ],

[ 0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 ],

[ 0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 ],

[ 0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 ],

[ 0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 ],

[ 0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7 ],

[ 0.71, 0.72, 0.73, 0.74, 0.75, 0.7

arr=np.arange(0.01,1.01,0.01).reshape(10,10)

arr

#### Create an array of 20 linearly spaced points between 0 and 1:

arr=np.linspace(0,1,20)

arr

mat = np.arange(1,26).reshape(5,5)

mat

array([[ 1, 2, 3, 4, 5],

[ 6, 7, 8, 9, 10],

[11, 12, 13, 14, 15],

[16, 17, 18, 19, 20],

[21, 22, 23, 24, 25]])

array([[12, 13, 14, 15],

[17, 18, 19, 20],

[22, 23, 24, 25]])

mat[2:,1:]

array([[ 2],

[ 7],

[12]])

mat[0:3,1].reshape(3,1)

* If you use **import** numpy, all sub-modules and functions in the numpy module can only be accesses in the numpy.\* namespace. For example numpy.array([1,2,3]).
* If you use **import** numpy **as** np, an alias for the namespace will be created. For example np.array([1,2,3]).
* If you use **from** numpy **import** \*, all functions will be loaded into the local namespace. For example array([1,2,3]) can then be used.

df.drop('R',axis=1,inplace=True)

df.drop('D',inplace=True)

df.loc[['A','B'],['W','Y']]

df.iloc[1,2]

df[df['W']>0]['X'] –if sf[‘W]>0 is true for C then whole C row is displayed

df[df['Z']>0][['X','Y']]

sal.BasePay.mean()

fg=pd.Series(sal.JobTitle,sal.EmployeeName)

fg

fg=sal[sal['EmployeeName']== 'JOSEPH DRISCOLL']['JobTitle']

fg

x=sal.TotalPayBenefits.idxmax()

sal.loc[x]['EmployeeName']

x=sal.TotalPayBenefits.idxmin()

sal.loc[x]['EmployeeName']

What was the average (mean) BasePay of all employees per year? (2011-2014) ? \*\*

y=sal.groupby('Year')

y.mean()['BasePay']

.groupby('JobTitle')

**How many unique job titles are there?**

df=sal['JobTitle'].value\_counts()

sd=df.sort\_values(ascending=False)

sd.head()

How many Job Titles were represented by only one person in 2013? (e.g. Job Titles with only one occurence in 2013?)

**How many people have the word Chief in their job title?**

df=sal['JobTitle']

for i in sal['JobTitle']:

if (i.find('Chief')!= -1):

print(i)

df=pd.read\_csv('Ecommerce Purchases.csv')

df

**How many people have English 'en' as their Language of choice on the website?**

ecom.Language.value\_counts()['en']

**How many people have the job title of "Lawyer" ?**

ecom.Job.value\_counts()['Lawyer']

**What are the 5 most common Job Titles?**

ecom['Job'].value\_counts().head(n=5)

**Someone made a purchase that came from Lot: "90 WT" , what was the Purchase Price for this transaction?**

ecom[ecom['Lot']=='90 WT']['Purchase Price']

**What is the email of the person with the following Credit Card Number: 4926535242672853**

ecom[ecom['Credit Card']==4926535242672853]['Email']

& ecom['Purchase Price']>float(95)

ecom[ecom['CC Exp Date']==2025]

**Hard: How many people have a credit card that expires in 2025?**

df=ecom['CC Exp Date']

count=0

for i in df:

if i.find('/25')!= -1:

count+=1

print(count)

z=pd.DataFrame(lambda y:y.split('@'),x)

for i in range(0,j):

lst\_dom.append(x[i][1])

print(x)

**What are the top 5 most popular email providers/hosts (e.g. gmail.com, yahoo.com, etc...)**

x=ecom['Email'].apply(lambda y:y.split('@'))

lst\_dom=[]

j=len(x)-1

for i in range(0,j):

lst\_dom.append(x[i][1])

df=pd.DataFrame(lst\_dom)

df[0].value\_counts().head(n=5)

Matplotlib

fig, axes = plt.subplots(nrows=2, ncols=2)

count=1

for i in axes[0]:

i.plot(x, x\*\*count, 'g')

count+=1

for i in axes[1]:

i.plot(x, x\*\*count, 'g')

count+=1

XS for multilevel index

df1.columns=['Col1','Col2','Col3','Col4','Col5']

df1.index=['R1','R2','R3','R4']

df=np.random.randn(4,5)

data = io.StringIO('''Fruit,Color,Count,Price

Apple,Red,3,$1.29

Apple,Green,9,$0.99

Pear,Red,25,$2.59

Pear,Green,26,$2.79

Lime,Green,99,$0.39

''')

data

df\_unindexed = pd.read\_csv(data)

df\_unindexed

df\_unindexed.xs(['Count','Price'],axis=1)

df=df\_unindexed.set\_index(['Fruit','Color'])

df.xs(['Count','Price'],axis=1)

import csv

f = open("top100.csv","r")

music = list(csv.reader(f))

stream\_numbers=list(int(music[i][3]) for i in range(1,len(music)))

track\_names=list(music[i][0] for i in range(1,len(music)))

def counting(list1):

dict1=dict()

for i in list1:

if i in dict1.keys():

dict1[i]+=1

else:

dict1[i]=1

return dict1

counts=counting(artists)

counts

from collections import Counter

artist\_counts = Counter(artists)

artist\_counts\_list = [[artist,count] for artist, count in artist\_counts.items()]

print(artist\_counts\_list)

artist\_counts\_list.sort(key= lambda x:x[1], reverse=True)

top\_artist=artist\_counts\_list[0]

top\_artist