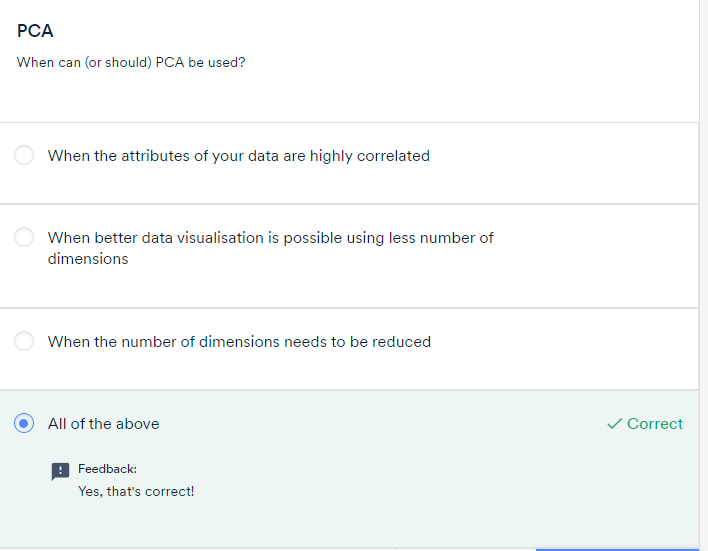
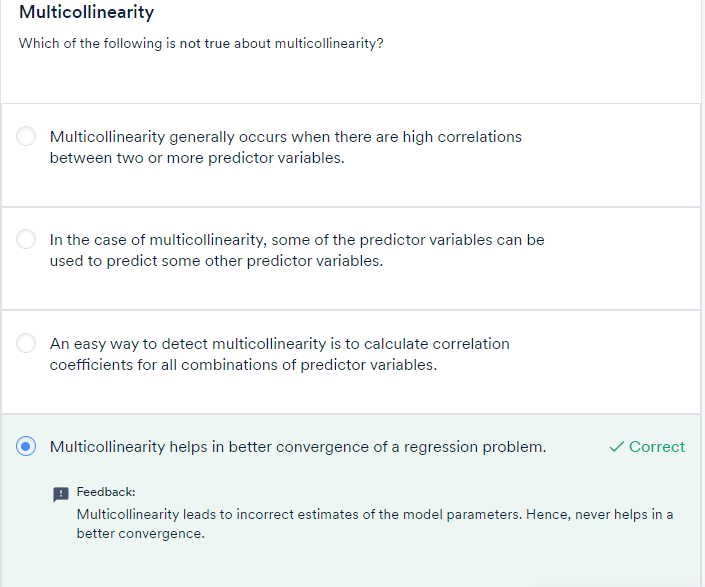
 a couple of situations where having a lot of features posed problems for us are as follows:

* The predictive model setup: Having a lot of correlated features leads to the multicollinearity problem. Iteratively removing features is time-consuming and also leads to some information loss.
* Data visualisation: It is not possible to visualise more than two variables at the same time using any 2-D plot. Therefore, finding relationships between the observations in a data set having several variables through visualisation is quite difficult.

In the image above, you can see that a data set having N dimensions has been approximated to a smaller data set containing 'k' dimensions. In this module, you will learn how this manipulation is done. And this simple manipulation helps in several ways such as follows:

* For data visualisation and EDA
* For creating uncorrelated features that can be input to a prediction model:  With a smaller number of uncorrelated features, the modelling process is faster and more stable as well.
* Finding latent themes in the data: If you have a data set containing the ratings given to different movies by Netflix users, PCA would be able to find latent themes like genre and, consequently, the ratings that users give to a particular genre.
* Noise reduction





As discussed in the previous segment, PCA is fundamentally a **dimensionality reduction technique**; it helps in manipulating a data set to one with fewer variables

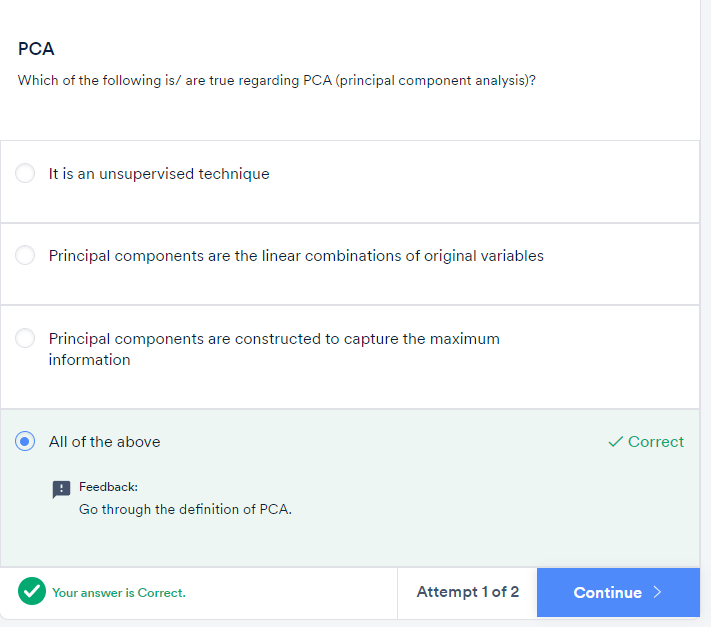
In simple terms, dimensionality reduction is the exercise of dropping the unnecessary variables, i.e., the ones that add no useful information. Now, this is something that you must have done in the previous modules. In EDA, you dropped columns that had a lot of nulls or duplicate values, and so on. In linear and logistic regression, you dropped columns based on their p-values and VIF scores in the feature elimination step.

Similarly, what PCA does is that it converts the data **by creating new features from old ones**, where it becomes easier to decide which features to consider and which not to.

PCA is a statistical procedure to convert observations of possibly correlated variables to ‘principal components’ such that:

* They are **uncorrelated** with each other.
* They are **linear combinations** of the original variables.
* They help in capturing maximum **information** in the data set.

Now, the aforementioned definition introduces some new terms, such as ‘**linear combinations**’ and ‘**capturing maximum information**’, for which you will need some knowledge of linear algebra concepts as well as other building blocks of PCA. In the next session, we will start our journey in the same direction with the introduction of a very basic idea: the **vectorial representation of data**.



**Summary**

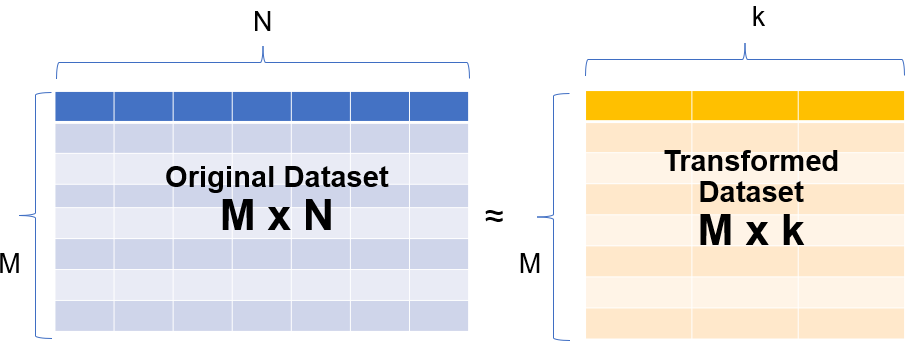
Here's a brief summary of what you learnt in this session!

Dimensionality reduction is a way of transforming a dataset having a high number of features into a smaller dataset. A couple of situations where having a lot of features posed problems for us are as follows:

* The predictive model setup: Having a lot of correlated features lead to the multicollinearity problem. Iteratively removing features is time-consuming and also leads to some information loss.
* Data visualisation: It is not possible to visualise more than two variables at the same time using any 2-D plot. Therefore, finding relationships between the observations in a data set having several variables through visualisation is quite difficult.

In simple terms, dimensionality reduction is the exercise of dropping the unnecessary variables, i.e., the ones that add no useful information. Now, this is something that you must have done in the previous modules. In EDA, you dropped columns that had a lot of nulls or duplicate values, and so on. In linear and logistic regression, you dropped columns based on their p-values and VIF scores in the feature elimination step.

 PCA is one such dimensionality reduction technique, i.e., it approximates the original data set to a smaller one containing fewer dimensions. What PCA does is that it converts the data **by creating new features from old ones**, where it becomes easier to decide which features to consider and which not to.  To understand it visually, take a look at the following image.

****

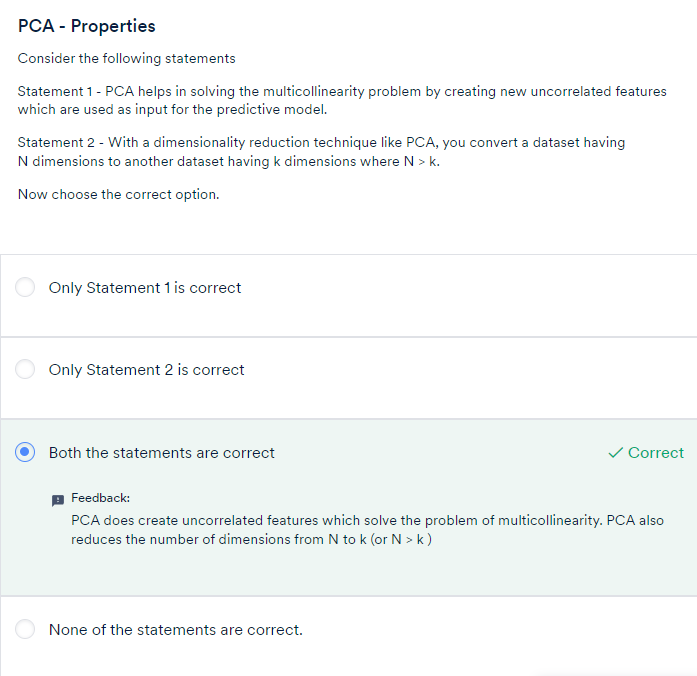
In the image above, you can see that a data set having N dimensions has been approximated to a smaller data set containing 'k' dimensions. In this module, you will learn how this manipulation is done. And this simple manipulation helps in several ways such as follows:

* For data visualisation and EDA
* For creating uncorrelated features that can be input to a prediction model:  With a smaller number of uncorrelated features, the modelling process is faster and more stable as well.
* Finding latent themes in the data: If you have a data set containing the ratings given to different movies by Netflix users, PCA would be able to find latent themes like genre and, consequently, the ratings that users give to a particular genre.
* Noise reduction

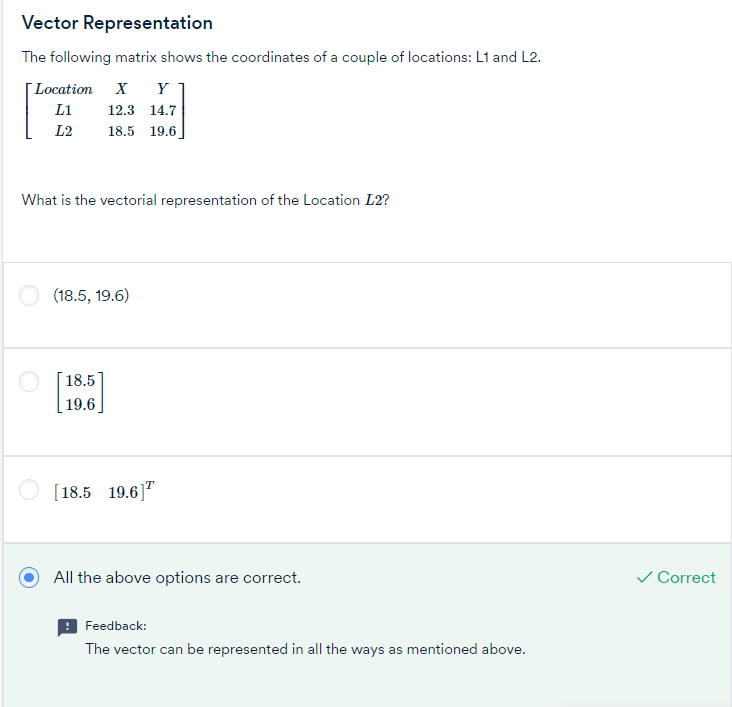
As explained in the video above, PCA is a statistical procedure to convert observations of possibly correlated variables to ‘principal components’ such that:

* They are **uncorrelated** with each other.
* They are **linear combinations** of the original variables.
* They help in capturing maximum **information** in the data set.

GRADED QUESTIONS:

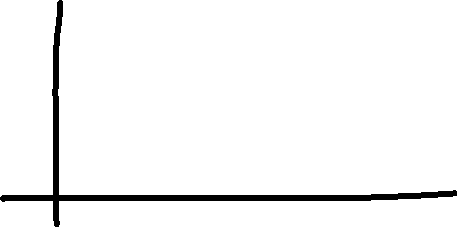
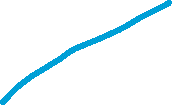


# Fundamentals of PCA - I!



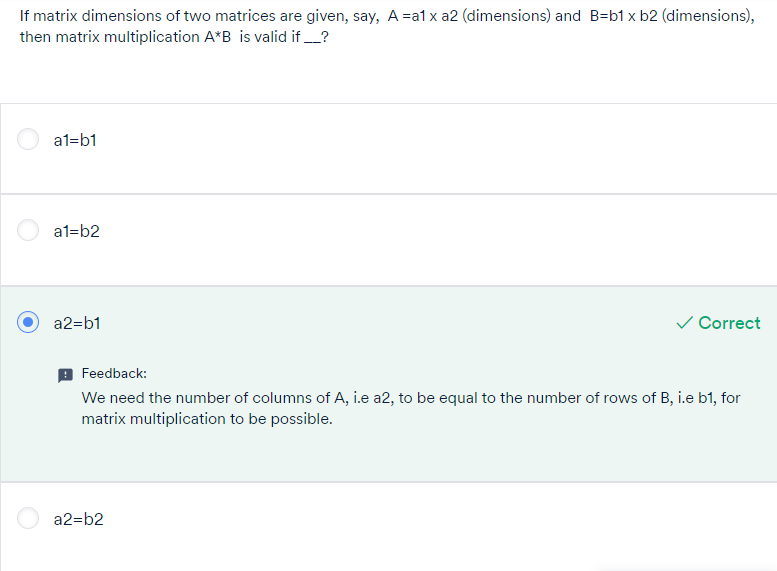


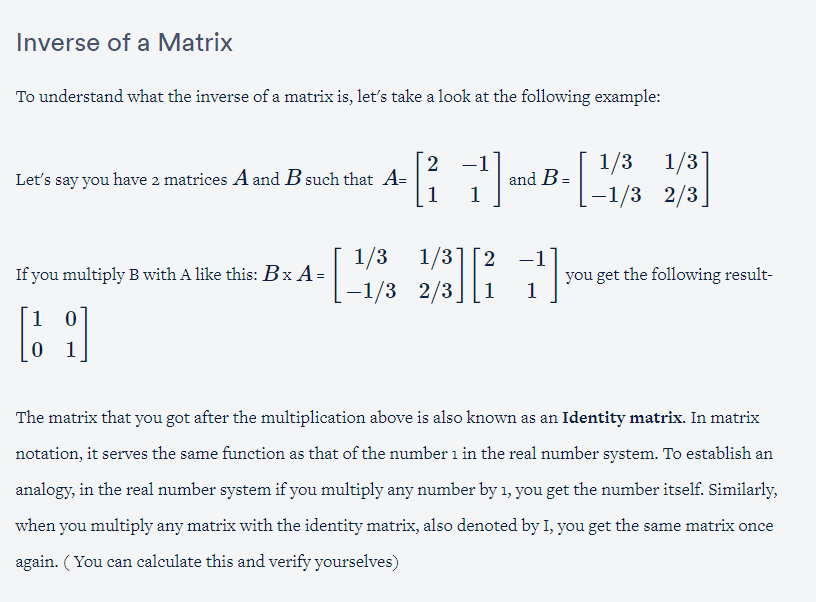


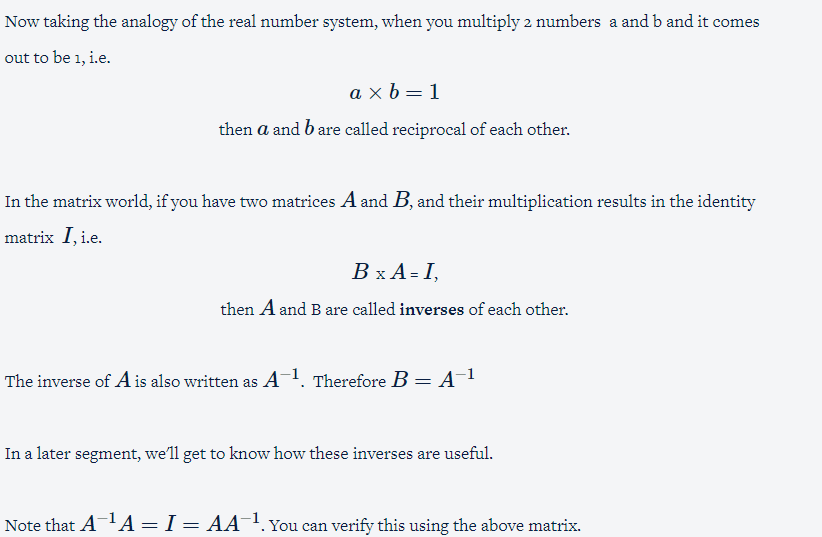


Let's summarise the learnings of the above lecture. Major outcomes from the above video are:

1. **Vectors have a direction and magnitude**  
   Each vector has a direction and magnitude associated with it. The direction is given by an arrow starting from the origin and pointing towards the vector's position. The magnitude is given by taking a sum of squares of all the coordinates of that vector and then taking its square root.  
     
   For example, the vector (2,3) has the direction given by the arrow joining (0,0) and (2,3) pointing towards (2,3). Its magnitude is given by  √22+32=√13.  
     
   Similarly, for a vector in 3 dimensions, say (2,-3,4) its direction is given by the arrow joining (0,0,0) and (2,-3,4) pointing towards (2,-3,4). And as in the 2D case, we get the magnitude of this vector as  √(2)2+(−3)2+(4)2=√29 .
2. **Vector Addition**  
   When you add two or more vectors, we essentially add their corresponding values element-wise. The first elements of both the vectors get added, the second element of both get added, and so on.  
   For example, if you've two vectors say   
   V1=(2,3) and V2=(1,2) then   
   V1+V2=(2+1,3+2)=(3,5).
3. In the**i, j**notations that we introduced earlier, the above addition can be written as V1+V2=(2i+3j)+(i+2j)=(2+1)i+(3+2)j=3i+5j  
   Similarly, this idea can be extended to multiple dimensions as well.
4. **Scalar Multiplication**  
   If you multiply any real number or scalar by a vector, then there is a change in the magnitude of the vector and the direction remains same or turns completely opposite depending on whether the value is positive or negative respectively.





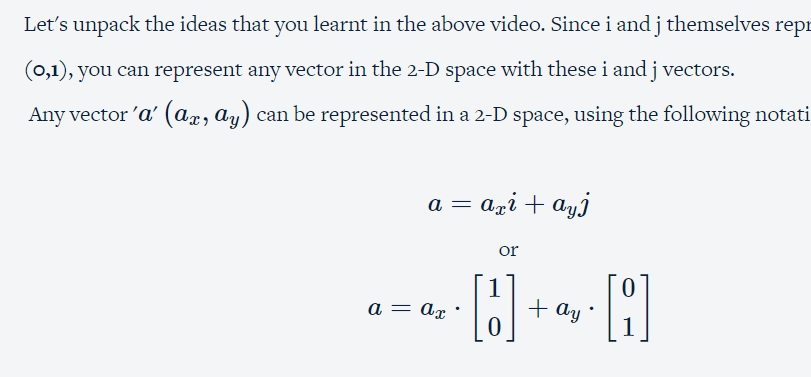


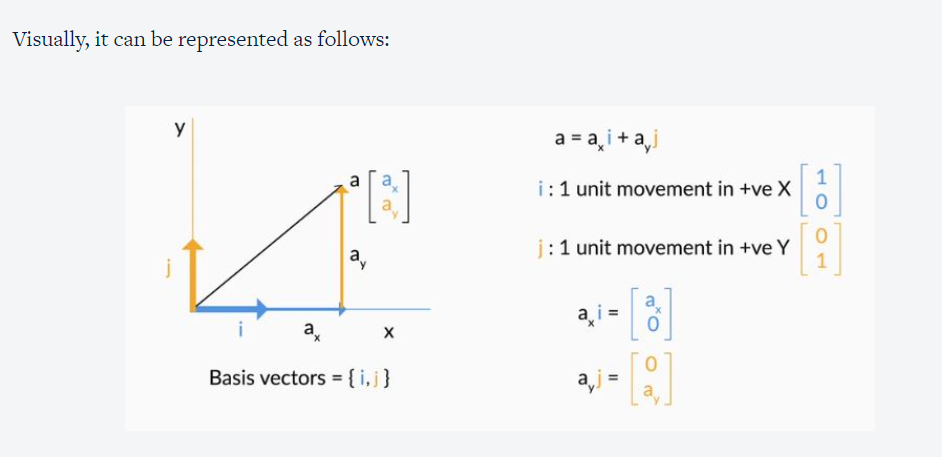
Basis:

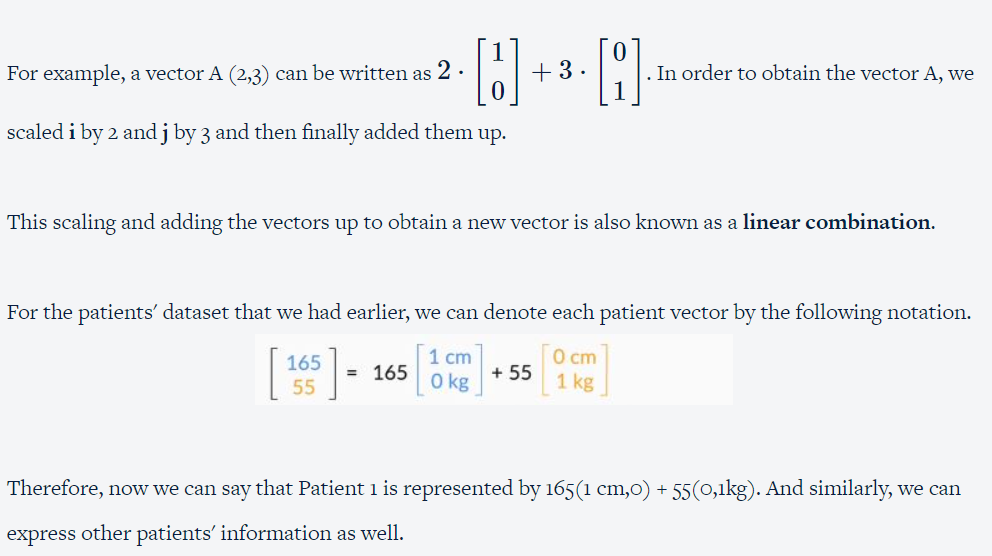
‘basis’ is a unit in which we express the vectors of a matrix.

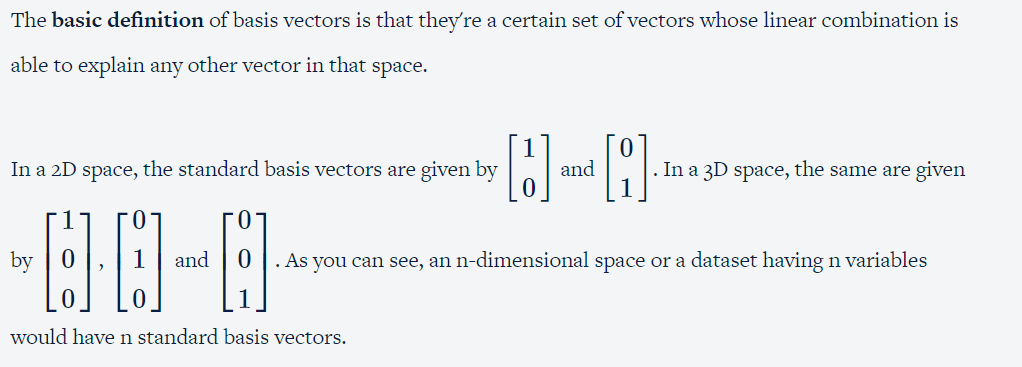
For example, we describe the weight of an object in terms of the kilogram, gram, and so on; to describe length, we use a metre, centimetre, etc. So for example, when you say that an object has a length of 23 cm, what you are essentially saying is that the object’s length is 23×1 cm. Here, 1 cm is the unit in which you are expressing the length of the object.

Similarly, vectors in any dimensional space or matrix can be represented as a linear combination of basis vectors.

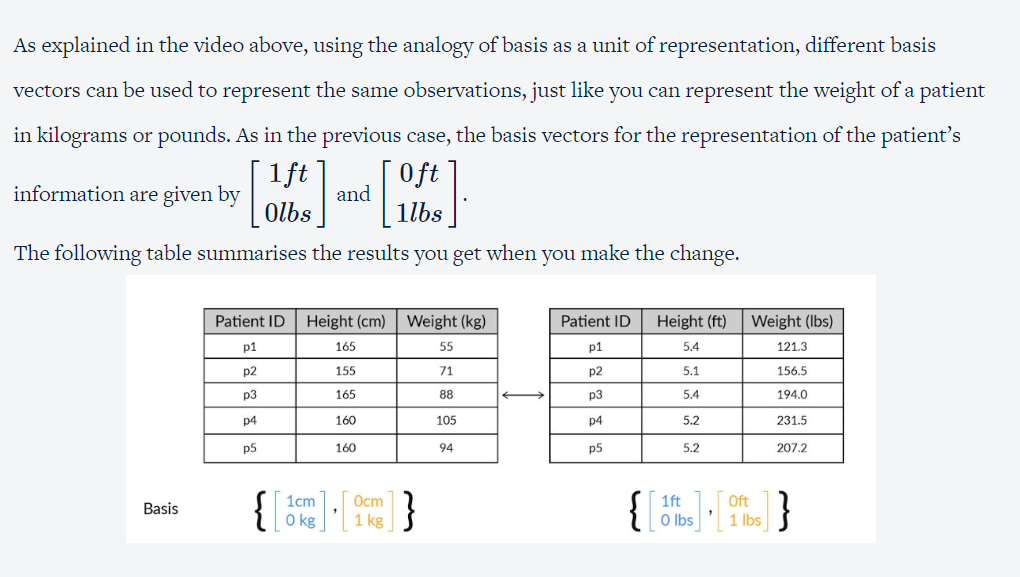


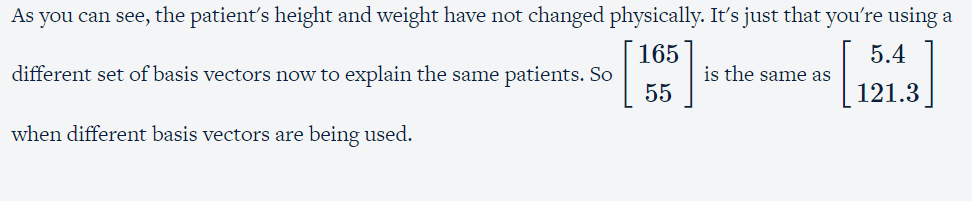




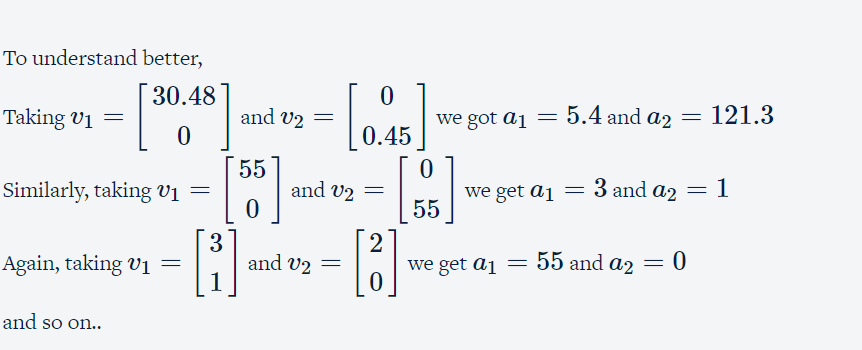


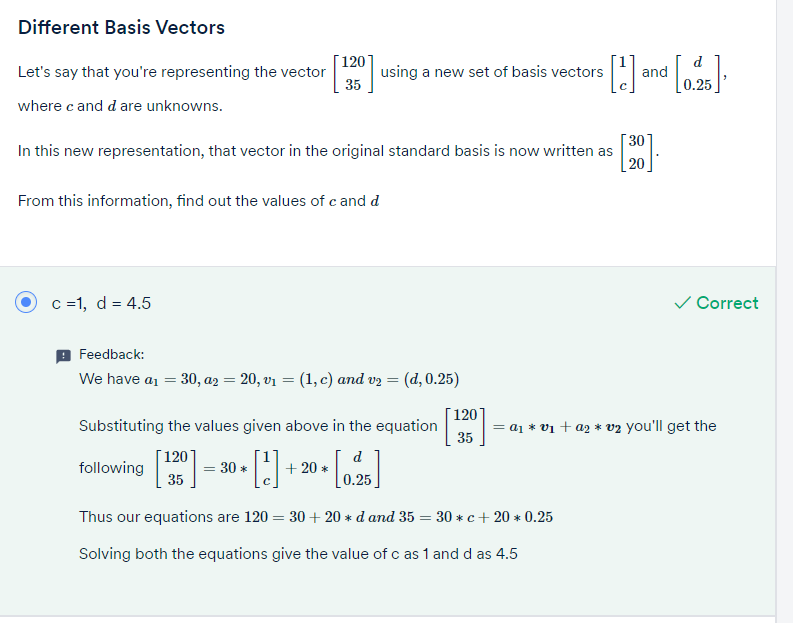


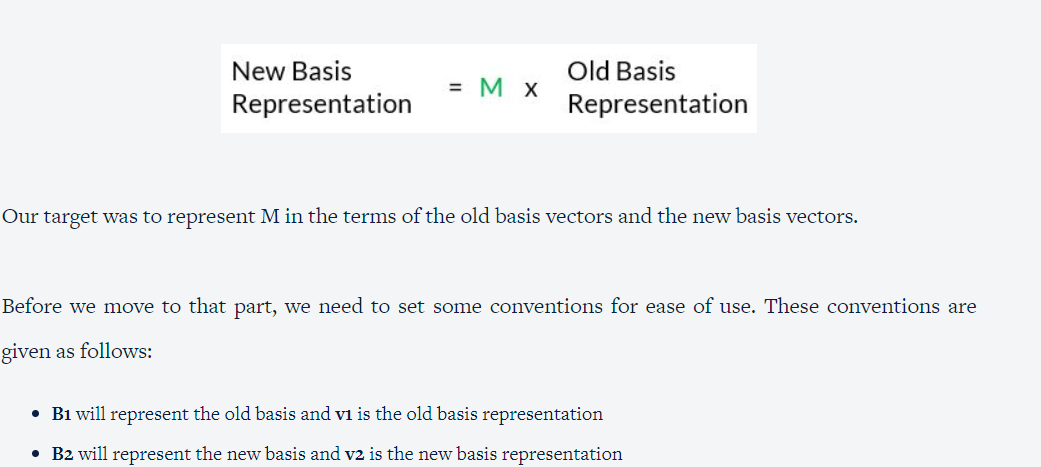


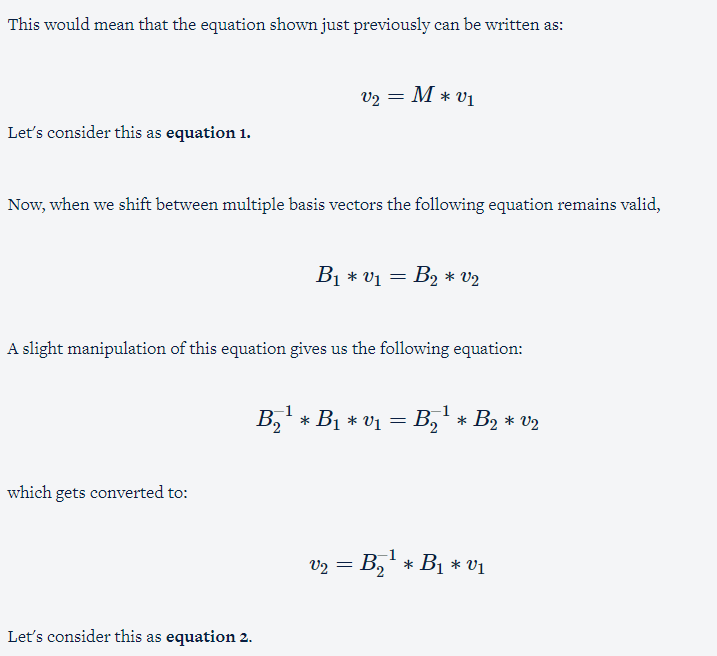


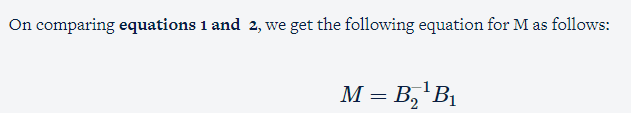
## **Relationship between the two sets of  basis vectors**

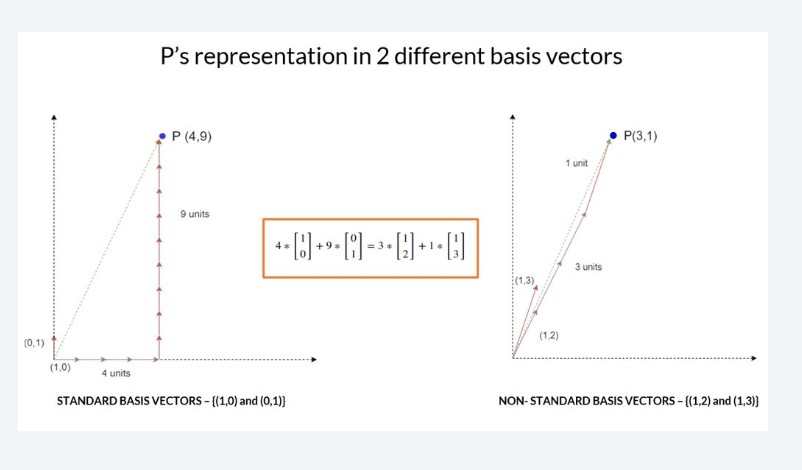


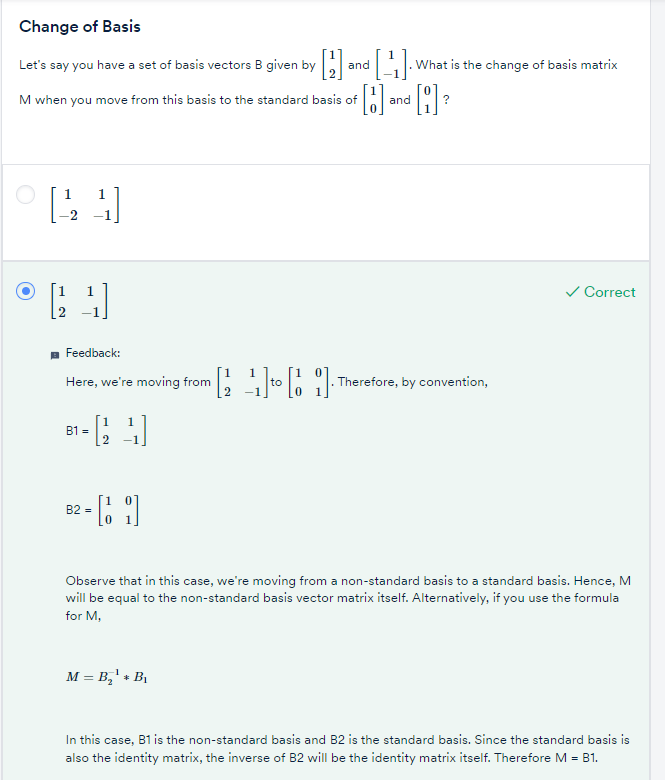


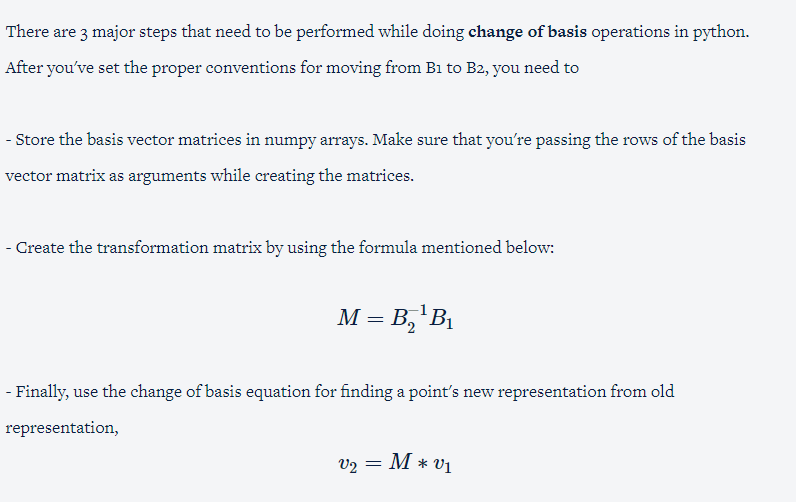




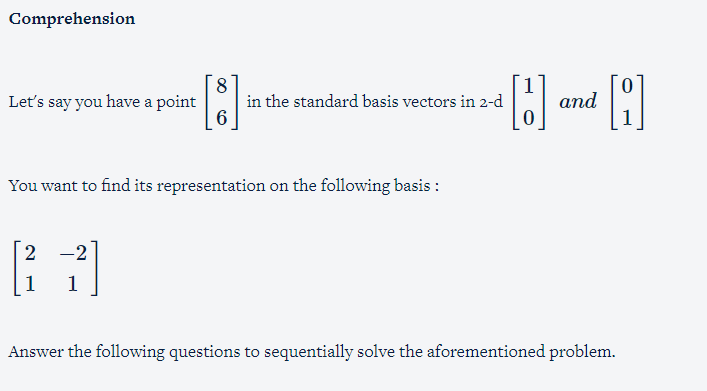


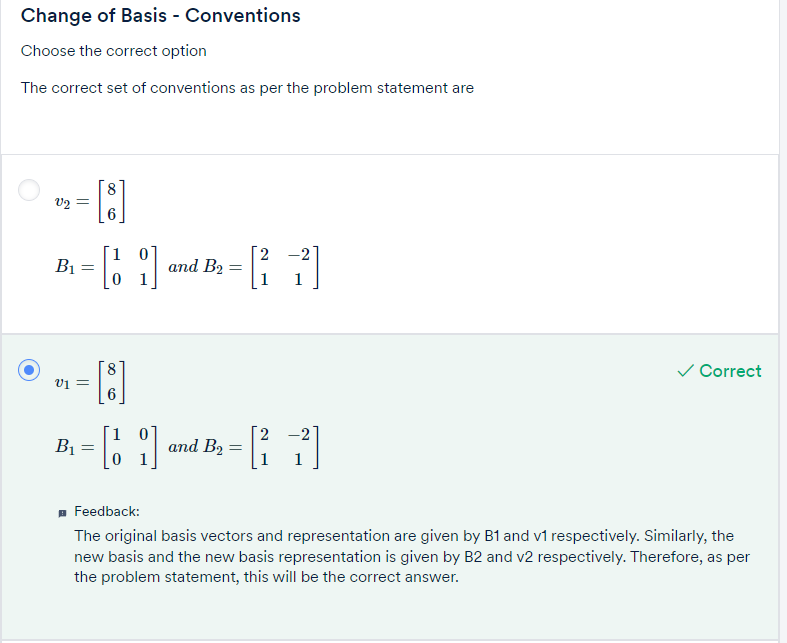


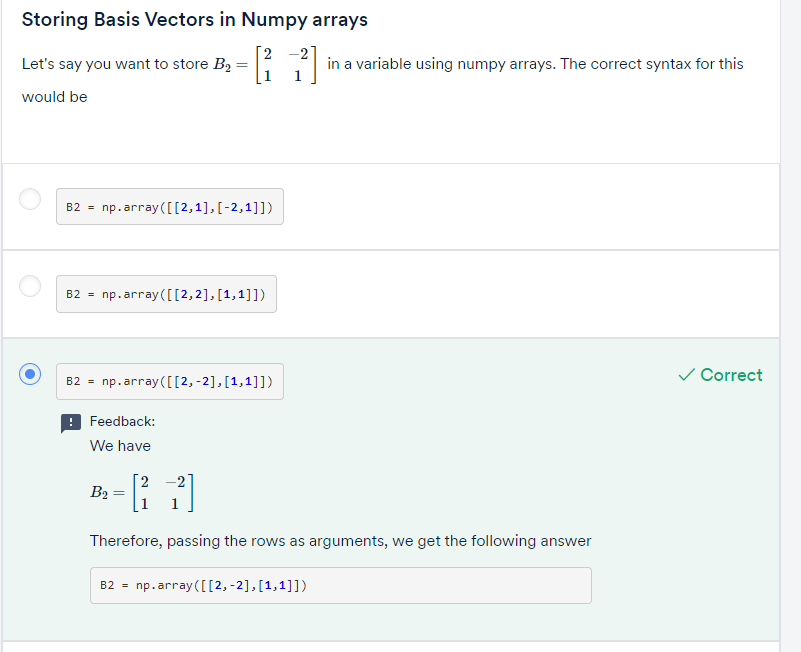


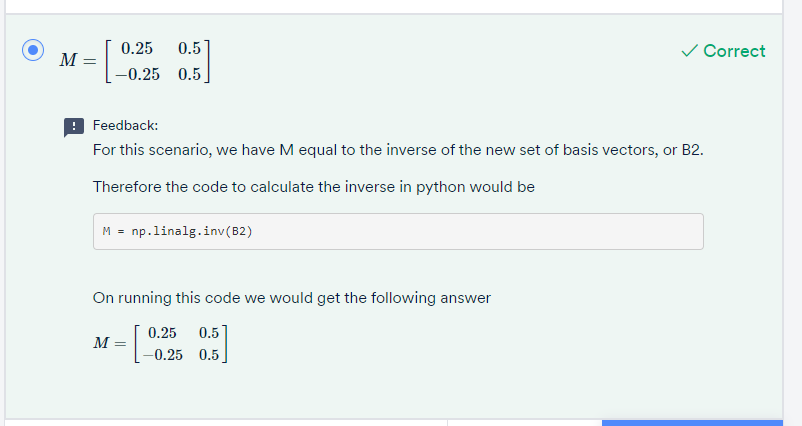


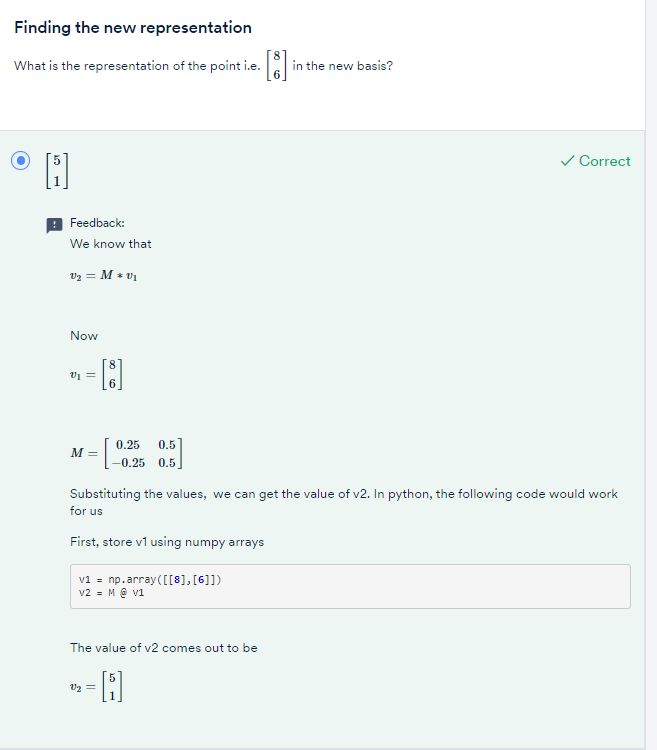
Question:

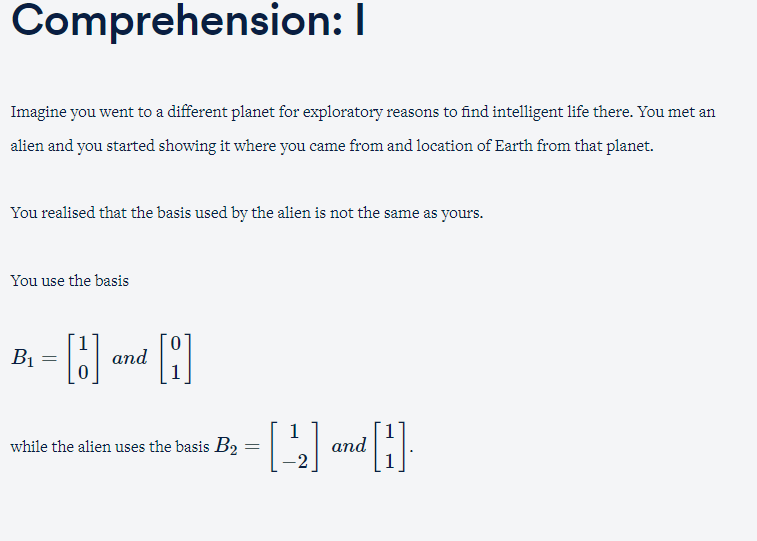


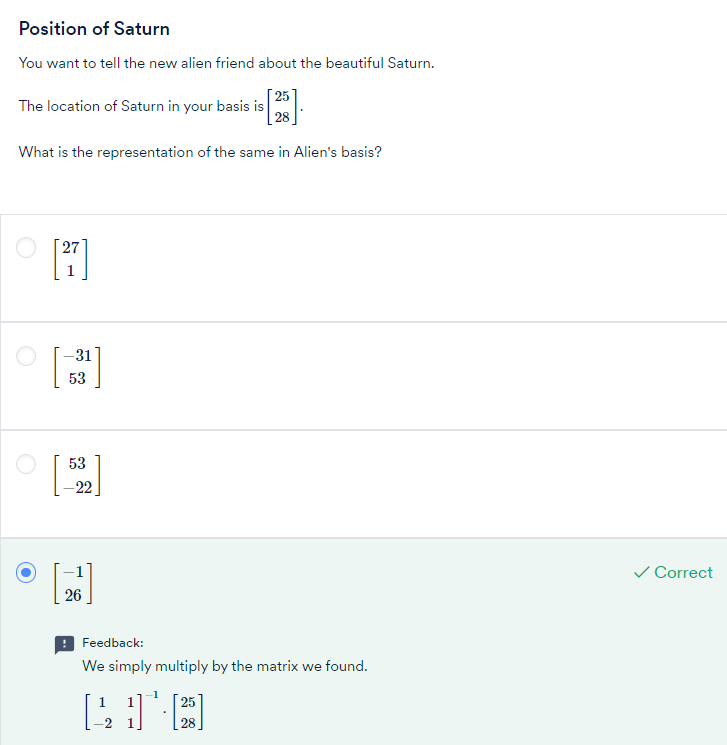


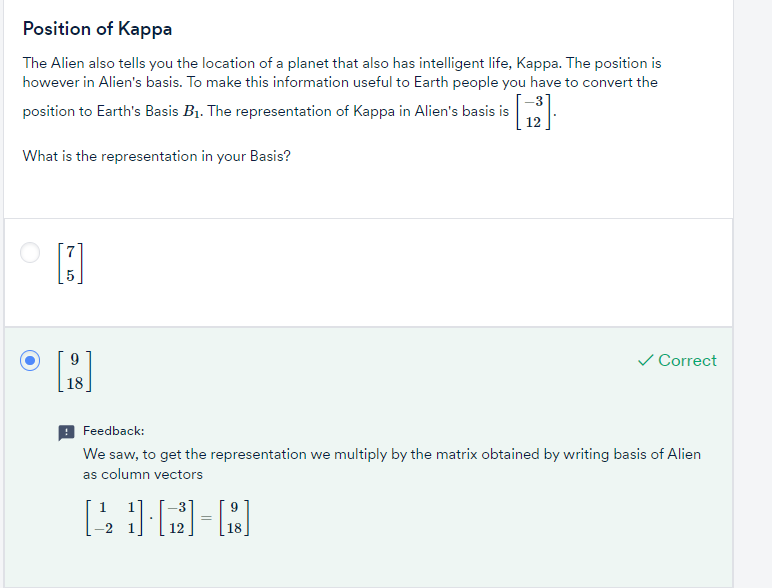






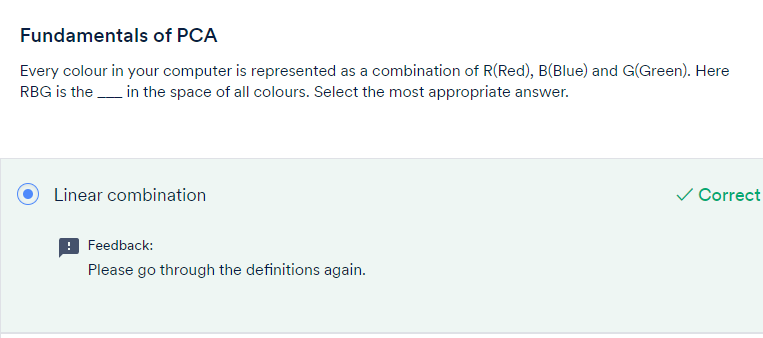


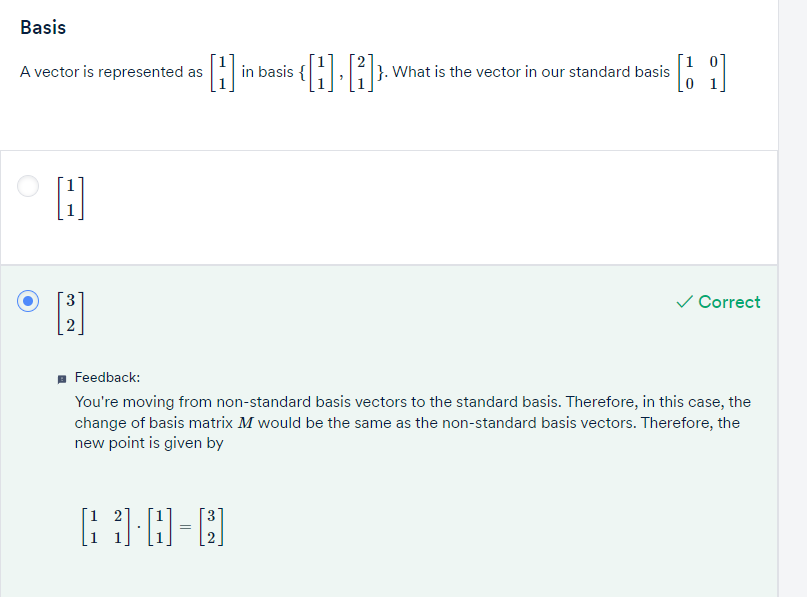




To summarise,

* PCA finds new basis vectors for us. These new basis vectors are also known as Principal Components.
* We represent the data using these new Principal Components by performing the change of basis calculations.
* After doing the change of basis, we can perform dimensionality reduction. In fact, PCA finds new basis vectors in such a way that it becomes easier for us to discard a few of the features.





Q&A

**Summary: I**

That was the end of quite a hectic session! Here's a summary of what you've learnt so far.

First, you came to know about PCA and how it is essentially a dimensionality reduction technique.  You saw the necessity for doing PCA in a couple of situations like

* A predictive model setup where there are a lot of features to eliminate
* A dataset where you needed to perform EDA and Data Visualisation

You understood how PCA not only helps in resolving the above two issues but has applications in several other areas like noise reduction, finding latent Themes and so on. Then you got a brief understanding of its definition:

**It is a statistical procedure that finds principal components or directions that are:**

* **Linear combination of the original variables**
* **Are uncorrelated**
* **Capture Maximum information in the dataset.**

After that, you went ahead and learnt some essential linear algebra concepts like vectors and their properties along with their associated operations. Then you studied another tool called matrix multiplication and matrix inverse, both of which proved invaluable in understanding the first fundamental building block of PCA: **Basis**

Basis is essentially the fundamental units in which you express your data. As you saw in the lecture videos, it is similar to how we use units for physical objects to measure things like height, weight, temperature, etc.

In vectors and vector spaces, we use basis vectors to represent the points in space. You understood how every observation in the space can be represented by **scaling** and **adding**the scaled basis vectors. This process is also called**a linear combination**.

Then you learnt one of the key ideas that helped you connect basis vectors and the idea of dimensionality reduction: using different basis vectors to represent the same points.

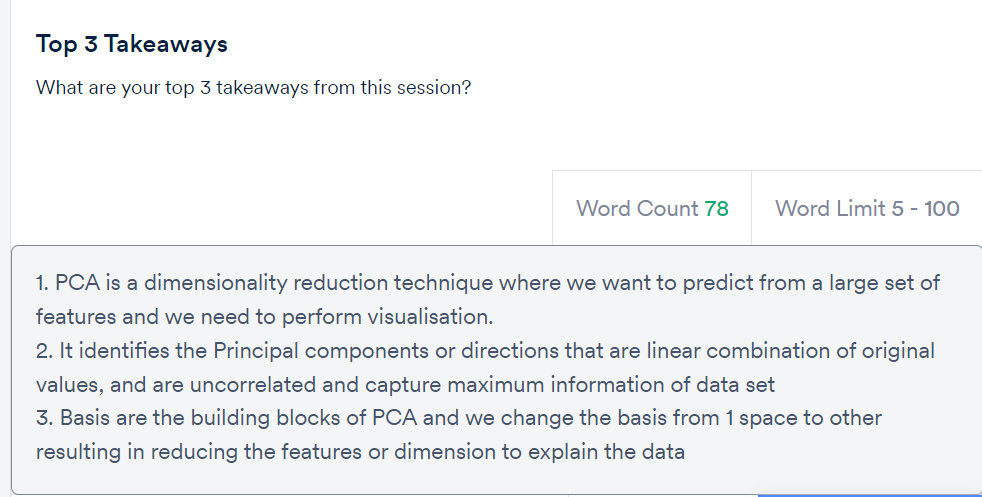
From there, you learnt how to change from one basis space to another using matrices. Here's a list of rules to help you revise the same.

1.) If you're moving from a basis space B  to the standard basis, then the change of basis matrix M is the same as the basis vectors of B written as its column vectors. Therefore, if there is a vector v  represented in B and you want to find its representation in the standard basis, then you'd have to perform Mv.

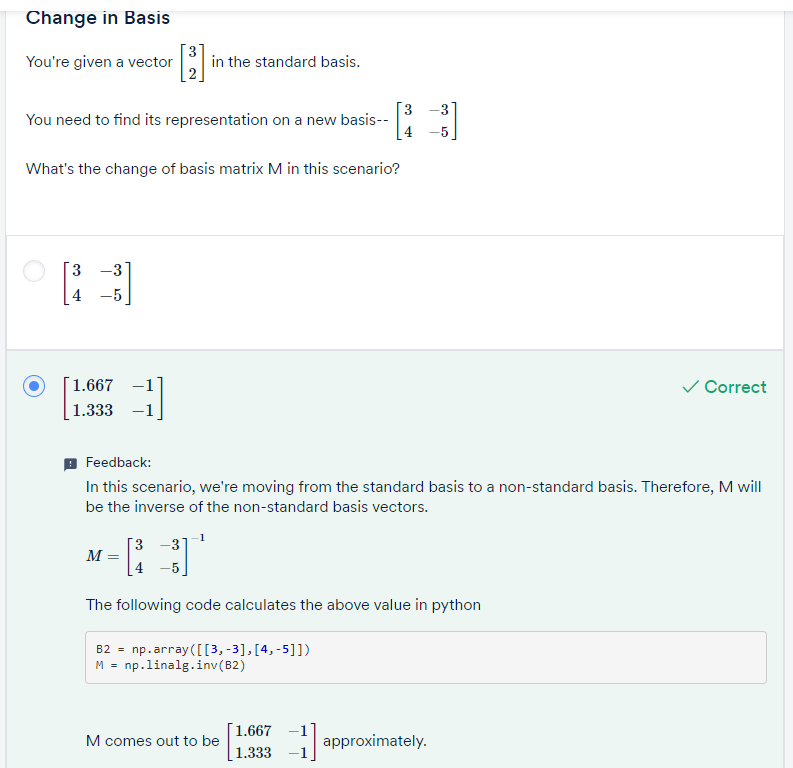
2.) If you want to go the other way around, where you have v represented in the standard basis and want to find its representation in B  you multiply it by its inverse  - M−1v

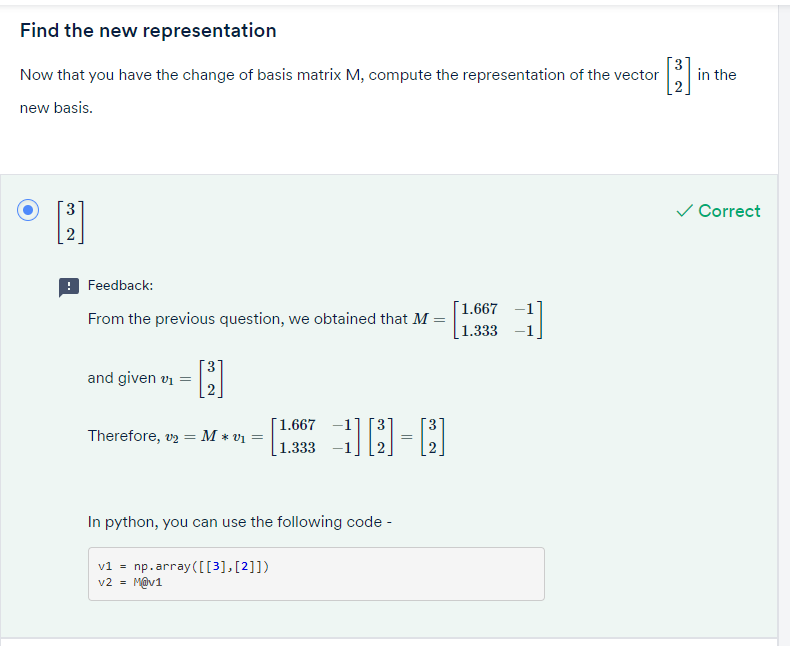
3.) Finally, if you want to find the change of basis matrix M where you move from two non-standard basis vectors - say from B1 to B2 then you can get that by calculating this value -

B−12B1. Note that in all the above cases, the basis vectors should be represented in the same units.



GRADED QUESTIONS:

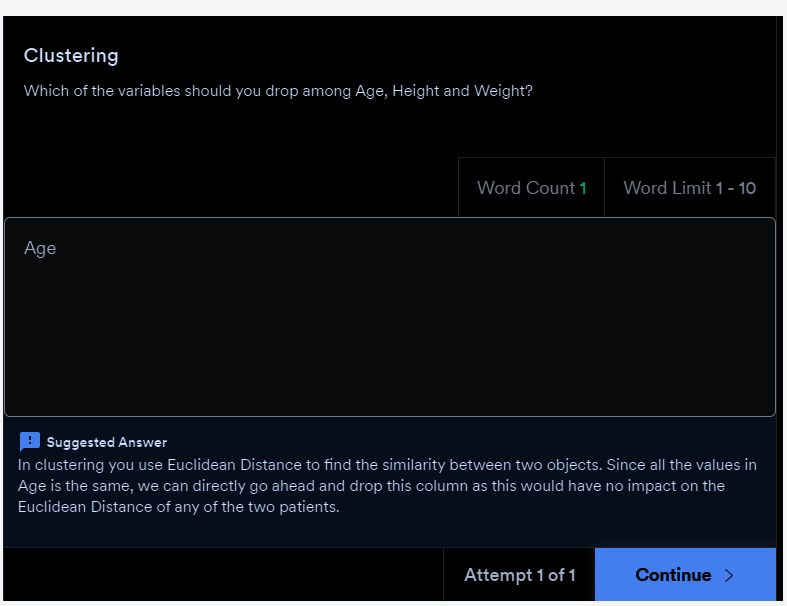




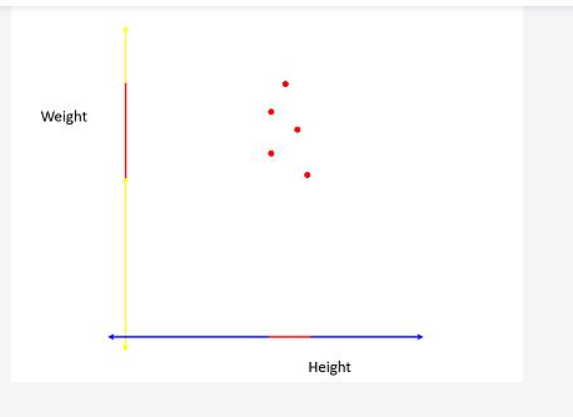
# Variance as Information:

As you saw in the example, the first image didn't have much information in it. Speaking of it in the ways the pixels are arranged, it is the same colour throughout. However, there are a lot of things that you could distinguish easily in the second image and therefore that image has a lot to offer in terms of information. The pixels have a lot of variety and therefore that image has more variance and equivalently, more information.

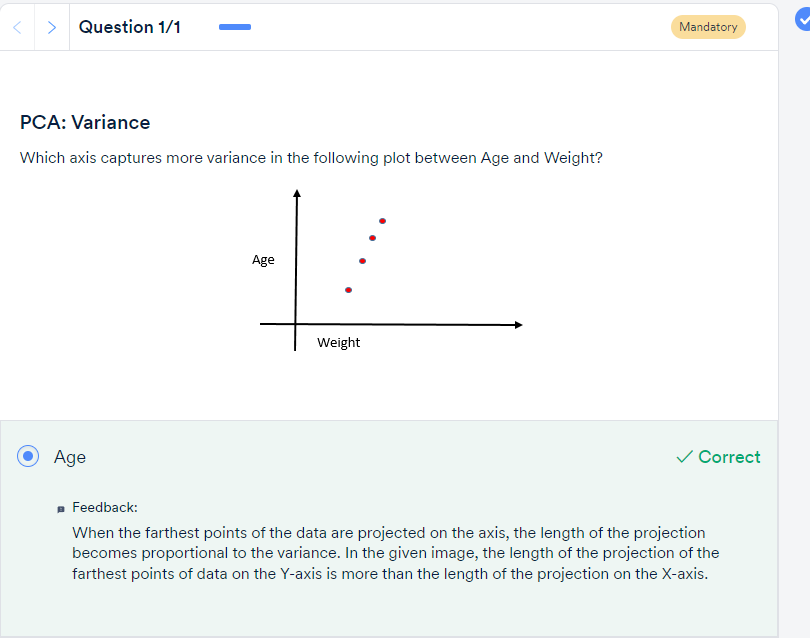
Variance = Information



So the key takeaway from the above lecture is to measure the importance of a column by checking its variance values. If a column has more variance, then this column will contain more information.

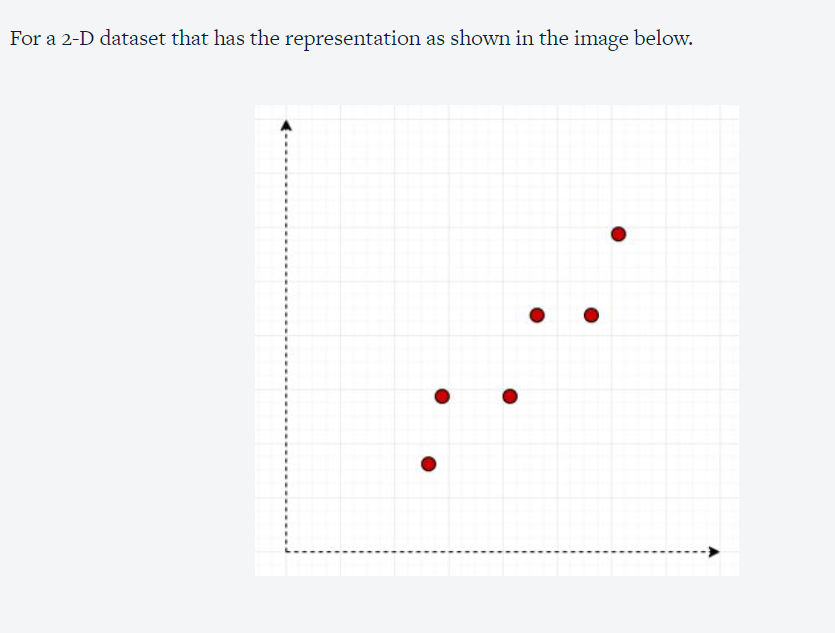


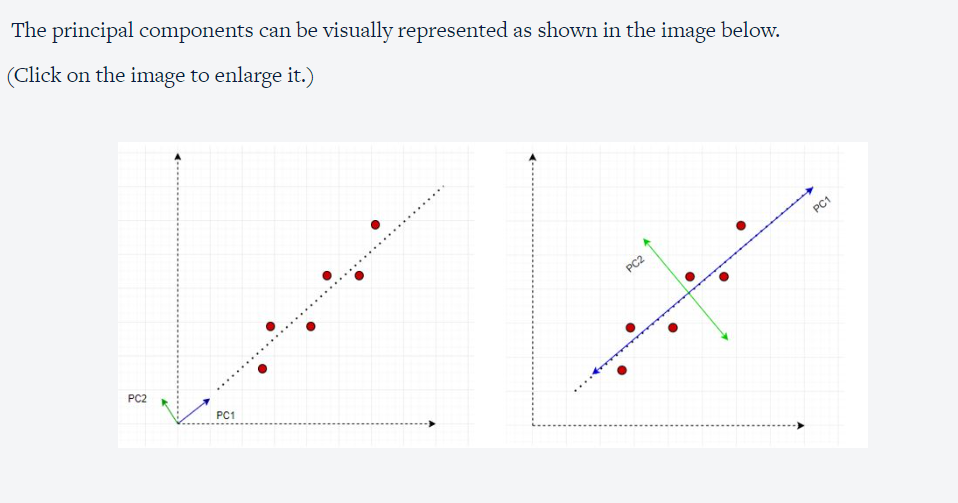
The red line on the Height and Weight axes shows the**spread** of the projections of the vectors on those axes. As you can see here, the spread of the line is quite good on the Weight axis as compared to the Height axis. Hence you can say that Weight has more variance than Height. This idea of the spread of the data being equivalent to the variance is quite an elegant way to distinguish the important directions from the non-important ones.



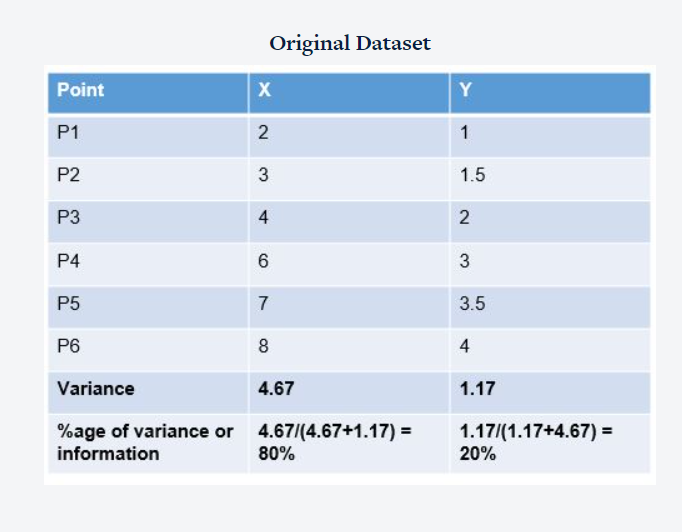
Basically, the steps of PCA for finding the principal components can be summarised as follows.

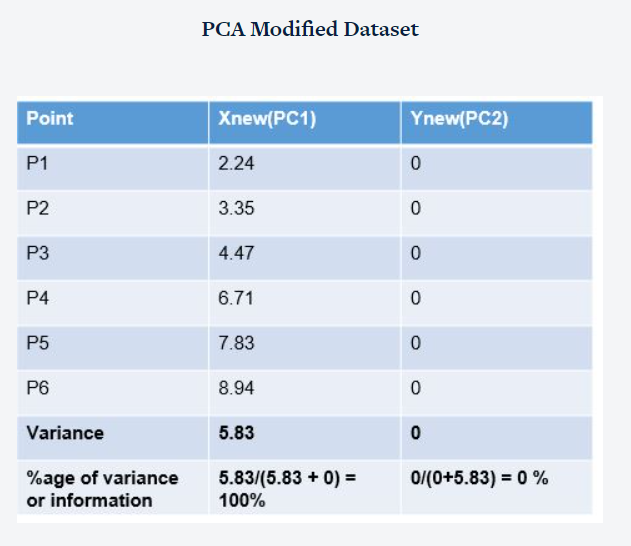
* First, it finds the basis vector which is along the best- fit line that maximises the variance. This is our first **principal component or PC1.**
* The second principal component is perpendicular to the first principal component and contains the next highest amount of variance in the dataset.
* This process continues iteratively, i.e. each new principal component is perpendicular to all the previous principal components and should explain the next highest amount of variance.
* If the dataset contains ***n*** independent features, then PCA will create ***n*** Principal components.





Also, once the Principal Components are found out, PCA assigns a %age variance to each PC. Essentially it's the fraction of the total variance of the dataset explained by a particular PC. This helps in understanding which Principal Component is more important than the other and by how much





Since 100% of the total variance or information of the entire dataset is present in only one of the columns (PC1) we can safely drop PC2 and still be assured of losing no information.

The steps  of PCA as summarised in the above video are as follows:

* Find n new features - Choose a different set of n basis vectors (non-standard). These basis vectors are essentially the directions of maximum variance and are called Principal Components
* Express the original dataset using these new features
* Transform the dataset from the original basis to this PCA basis.
* Perform dimensionality reduction - Choose only a certain k (where k < n) number of the PCs to represent the data.  Remove those PCs which have fewer variance (explain less information) than others.

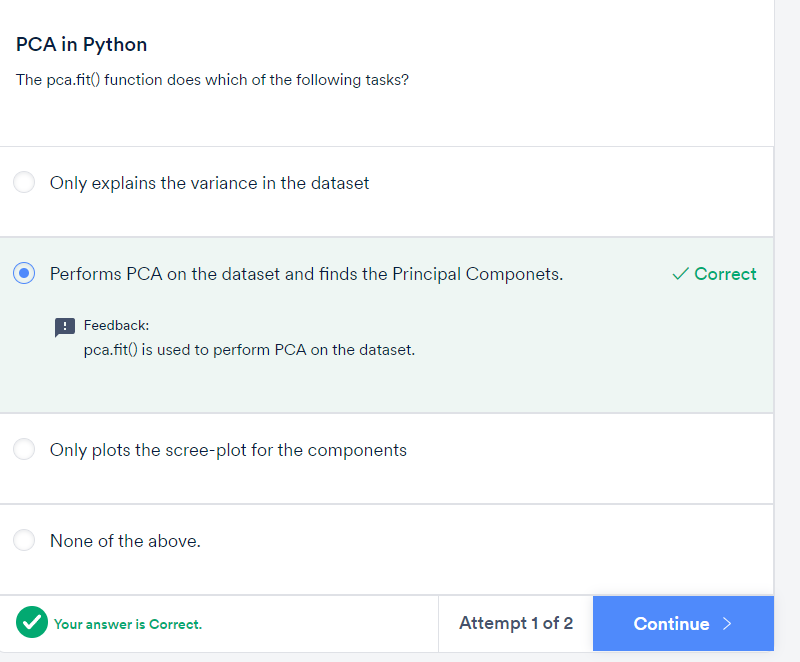
**Summary: II**

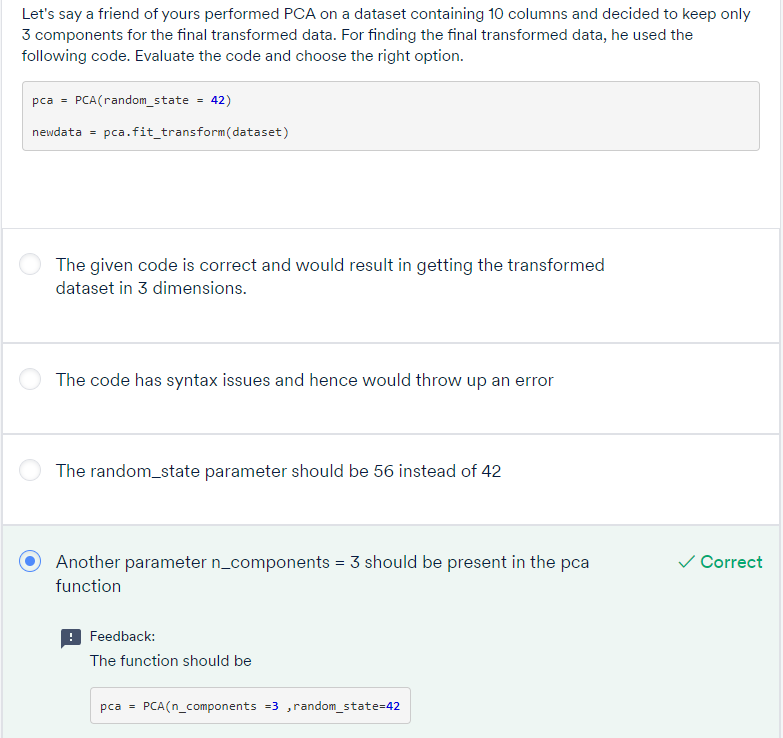
Let's reiterate the learnings of the past two sessions:

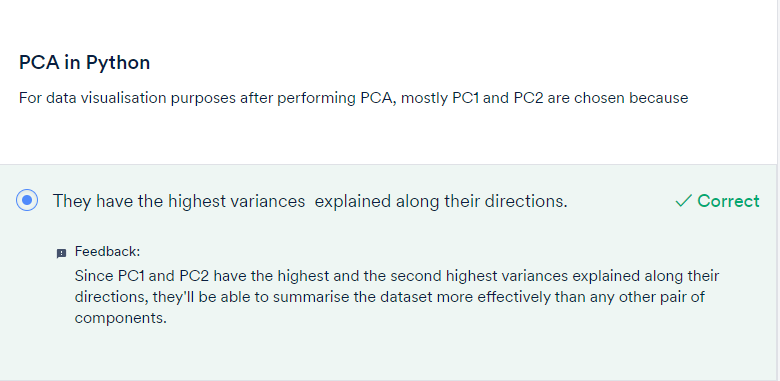
* You understood the concept of basis vectors and how they're helpful in the representation of data points.
* You then learnt about how you can use different basis vectors to represent the same information.
* Using the previous knowledge, you came to know that when represented under some '**ideal basis vectors'**, it becomes easier for us to do dimensionality reduction. However, you didn't exactly know how to find those ideal basis vectors.
* Then you learnt about the concept of variance and how more variance meant more information.
* Then you derived that the more important columns in a dataset are the ones which capture more variance than the others.
* Subsequently, you deduced that the most important directions, rather than just columns, are those that **capture maximum variance**. The ideal basis vectors that we talked about in the previous case are in fact those that do the same.
* These basis vectors or directions that capture the maximum variance are essentially the **Principal Components**for the dataset.

Here's a brief summary of what you've learnt so far:

* Applying PCA on a dataset in python
* Evaluate the amount of variance explained by each component
* Use the scree-plot to choose how much variance you need to explain with your transformed dataset
* Transform the dataset to the new chosen Principal Components and then perform dimensionality reduction
* Use the new dataset for visualisation of the observations



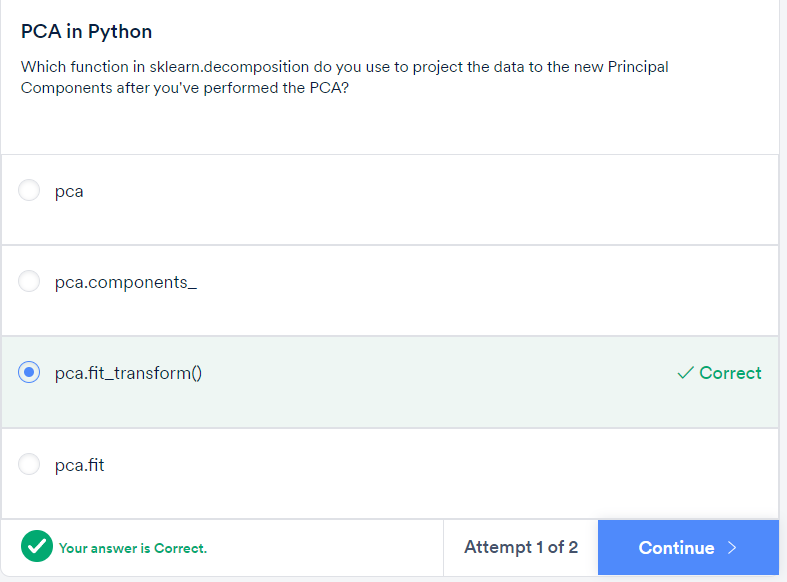


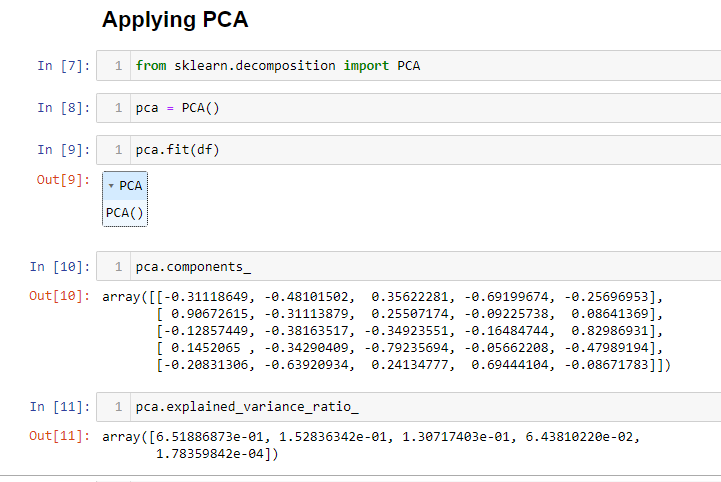


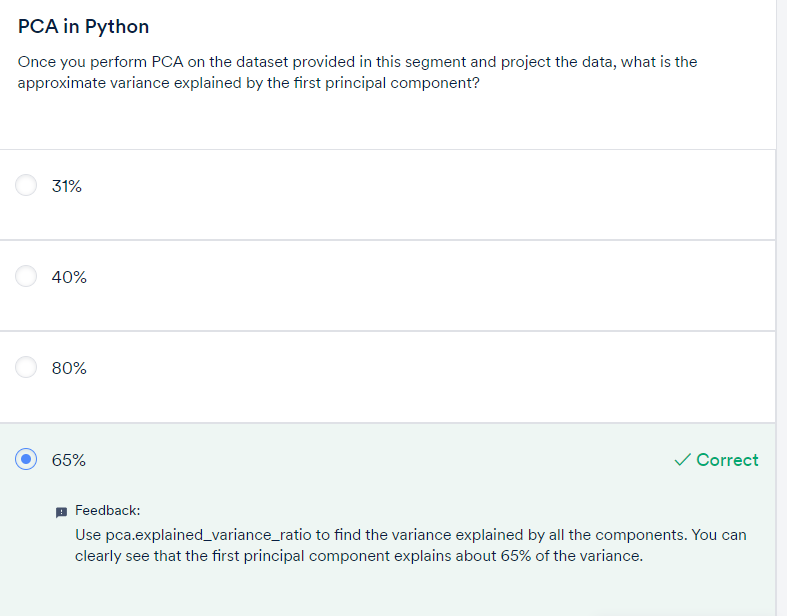
# Improving Model Performance – I:

You saw the process of building a churn prediction model using logistic regression. Some important problems with this process that Rahim pointed out are:

* **Multicollinearity** among a large number of variables, which is not totally avoided even after reducing variables using RFE (or a similar technique)
* Need to use a **lengthy iterative procedure**, i.e. identifying collinear variables, using variable selection techniques, dropping insignificant ones etc.
* A **potential loss of information**due to dropping variables
* **Model instability** due to multicollinearity







Those were some important points to remember while using PCA. To summarise:

* Most software packages use SVD to compute the principal components and assume that the data is **scaled and centred,**so it is important to do standardisation/normalisation.
* PCA is a**linear transformation method** and works well in tandem with linear models such as linear regression, logistic regression, etc., though it can be used for computational efficiency with non-linear models as well.
* It should **not be used forcefully to reduce dimensionality**(when the features are not correlated).

SVD – Singular value decomposition

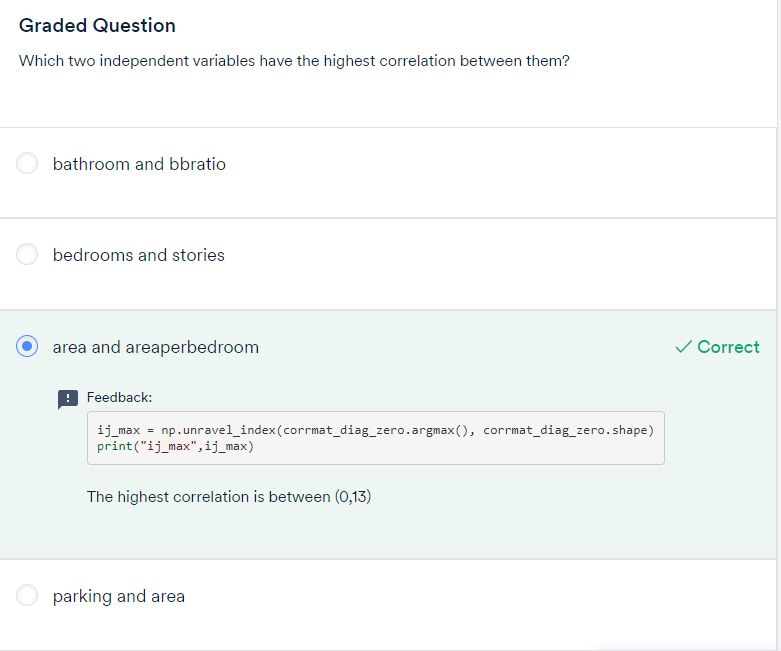
You learnt some important shortcomings of PCA:

* PCA is limited to linearity, though we can use **non-linear techniques such as t-SNE**as well (you can read more about t-SNE in the optional reading material below).
* PCA needs the components to be perpendicular, though in some cases, that may not be the best solution. The alternative technique is to use **Independent Components Analysis.**
* PCA assumes that columns with low variance are not useful, which might not be true in prediction setups (especially classification problem with a high class imbalance).

t-SNE (t-Distributed Stochastic Neighbor Embedding) or ICA

Here's a list of useful functions that use after importing the PCA function from sklearn libraries.

* **pca.fit()** - Perform PCA on the dataset.
* **pca.components\_ -**Explains the principal components in the data
* **pca.explained\_variance\_ratio\_** - Explains the variance explained by each component
* **pca.fit(n\_components = k)**- Perform PCA and choose only k components
* **pca.fit\_transform  -**Transform the data from original basis to PC basis.
* **pca(var) -**Here 'var' is a number between 0-1. Perform PCA on the dataset and choose the number of components automatically such that the variance explained is (100\*var)%.



a = np.array([[1,2,3],[4,3,11]])

print("Original array:")

print(a)

print(a.argmax())

print(a.shape)

i,j = np.unravel\_index(a.argmax(), a.shape)

print("Index of a maximum element in a numpy array along one axis:")

print(i,j)

ij\_max = np.unravel\_index(corrmat\_diag\_zero.argmax(), corrmat\_diag\_zero.shape)

print("ij\_max is",ij\_max)

print("Maximum correlation :",corrmat\_diag\_zero[ij\_max])

