this video was brought to you by IND
dentle IO learning python Made Simple

how's it going everyone in today's video
we're going to be covering 10 python
Concepts that you should know about so
starting with the first concept after
you have downloaded python from
python.org or wherever you downloaded it
from you can create a file anywhere on
your computer using the py extension for
example if we want to create our main
script we'll just type in main as the
name and add the py extension this will
make it a pi file or a python file which
will be executed by an interpreter in

Python if you want to create a variable
you start by typing the variable name
followed by an equal sign and the value
that you want to assign to that name for
example here we can type in Bob and Bob
might have an age so we can create
another variable called age and that's
going to be 20 now if we ever want to
use these we can just refer to those
variable names

instead of hardcoding in the values or
typing them in by hand each time we want
to use them for example using the print
statement we can print the name comma
age if I can spell that I still can't
spell that age and when we run it in the
console you should see that we were able
to use both of those variables which
again is quite convenient because we can
use these anywhere in the program I mean
another example would be
hello my name is and then insert name
and we would get that as an output in

Python we have many data types and the
basic data types are integers floats
strings Boolean which can either be set
to true or false then we have lists
tupes sets and dictionaries an integer
is any whole number while a float is any
decimal number a string is anything
that's inside quotes and that can be
either single quotes or double quotes
and it just represents text while a
Boolean is used to represent two states
either true or false then we have lists
which can contain an arbitrary amount of
elements so in case you have a grocery

list or a list of names we have a way of handling that kind of data after we have tuples which are a lot like lists except they are immutable which means once you set the dat
you can't add anything to it or remove anything from it while with lists you can perform many operations such as adding elements and removing elements after that we have sets which are quite similar to lists except here you can not have any duplicates everything is guaranteed to be unique and finally we have dictionaries which is the basic way of representing key and value pairs so here we have a name of Bob and an age of 20 20 in Python we have the option to

use type annotations type annotations are completely optional and to show you what type annotations look like let's create a simple example such as name is equal to Bob this is obviously a string but we can also tell python explicitly that this was meant to be a string by annotating it with the string type and we can do the same thing for the age we'll say age of type integer is equal

to 11 but what you will notice is that
if we insert something wrong our code
editor is going to give us a warning
that we did not pass in the right type
and when your code gets more and more
complex These Warnings become a
lifesaver because as programmers we are
bound to make mistakes eventually
whether it's because we are drunk tired
or just not that attentive we're bound
to make a mistake eventually and having
These Warnings help a lot because if we
did not specify this to be of type
integer we would not get any warning
there the code editor would not know
what we were trying to do so we would be
able to do that even if it's not what we
wanted although it's important to point
out that even if we Define this to be of
type integer and we assigned it a string
the program is still going to run as
normal type annotations do nothing in
terms of how the program executes they
are just a tool for the developer and my
favorite analogy for using type
annotations is the traffic light if you
think about it when you're driving a
traffic light doesn't really do anything

it just has three callers that tell you
when it's safe to drive through an
intersection it doesn't do anything at
all you can still drive through that
intersection even if the lights are red
sometimes that might go well and other
times you might end up in an accident so
type annotations are kind of like the
same thing they do not prevent you from
making terrible mistakes but they warn
you that you're about to make a mistake

moving on constants
in Python we don't really have any way
of creating constants but you can use
type annotations along with the caps
lock naming convention to show that it
is a constant for example if we were to
import from typing the final type we can
now tell python using type annotations
that we are creating a constant for
example we might have a version number
of type final of type
string and that's going to equal
1.0
12 and that would be the correct way to
define a constant even without the type
annotation you will know it's a constant

by the use of uppercase characters but unfortunately we can still change the constant so we can say that the version is now 1.1 and your code editor will give you a warning only if you're using the final type because this type annotation explicitly tells the developer that this value or this variable should not be overwritten so here we're going to get that warning but otherwise you can even have a constant such as Pi which will be of type final of float and that can be 3.1415 and that's another example of how you could create a constant in Python

next we're going to be looking at how we can create reusable code in Python so to get started I'm going to import from datetime the date and time and what we want to do with this is show the user the current date and time so we're going to print this is the current time and we're going to print the datetime dot now when we run this we're going to get the following output this is the current time with the current date and time so that works perfectly fine but if we ever

want to reuse this we're going to have
to copy and paste this everywhere which
is a terrible idea because one day you
might decide what if I wanted to type in
this is the current time and date or
date and time well that's perfectly fine
we updated it where we had to but the
problem is it did not update in the
other places which sucks because now we
need to manually waste our time going
through our project which could be
thousands of lines long to fix all the
occurrences so this is where functions
come in and in Python to create a
function we use the dev keyword so here
we can type in show date and this is
going to return none and then all we
need to do is indent this code inside
the function since python uses uses
significant
indentation for its code blocks but now
thanks to that we can just type in show
date and duplicate that and no matter
how many times we use this function it's
always going to come from the same
source which means now we can type in
this is the current date and remove the
last bit and it will update everywhere

we use this function otherwise if you want to make a function more customizable you can provide parameters and to do so we can create another function called greit that takes a parameter called name and that will be of type string and once again this will return none and inside here we can print the F string of hello name so that the next time we want to use this function we can say greet Bob duplicate that and greet Luigi and that will greet both of them using the parameter that we have specified which is really nice because once again if we ever need to apply any changes we can do so here we can say ciao instead of hello and the next time we run this we will get two outputs with that update and functions can even return results for example we might have a function called add that takes a of type float and B of type float because we want to add these two numbers together and the intention with this is to return a float well to do so you just need to return that data type so a plus b and now if we were to print let's say say add one and

two this function would return a result
which means calling this function will
give us back the result of three and
it's actually optional to define a
return type it just once again makes our
code much more safe because if for
whatever reason we return the string
of hello our code editor is going to
complain that hello was the wrong type
and if your function doesn't return
anything for example if you're just
printing hello you can explicitly say
that you are returning none which just
tells the developer that this function
was only meant to be run and that they
should not expect a return value from it

moving on we have the concept of classes
and to break down what a class is a
class is just a blueprint for code but
let's get started by creating the main
components of a class to see how it
works and in Python to define a class we
use the class keyword followed by a name
starting with an uppercase character and
this is how you would create a class now

obviously it would be much more useful

if this class could do something because all we did is insert an ellipses as a placeholder so what we're going to do instead of that is Define an initializer and an initializer is used to set up an instance of the class which means we're going to create an object from this class using some specific information and the specific information we want to use use for each one of these cars in this class is what color the car should be which will be of type string and how much horsepower that car has which will be of type integer and initializers return none by default and they will always return none but personally I love to specify that now inside the initializer we need to assign these values to the instance of the car and to do so we need to use the self keyword so self. Cola is going to equal Cola and self . horsepower is going to equal horsepower now with that being done we can create a car called Volvo which will be of type car and the car is going to be a red car with 200 horsepower so this part here is what we refer to as an instance of the class or an object of

the class because we are instantiating
it with this specific information so
that we can have a customized object and
what that means is that we can type in
Volvo and refer to the cola or be more
fitting to print that and also the
horsepower we can refer to those
attributes which are related to the
instance so that when we run this we
will get that information back as you
could see the class was just the
blueprint for how that car should look
and with that blueprint we can create
several cars and it just simplifies that
process so instead of having a Volvo or
in addition we could also have another
car called BMW which would be blue and
it would have the the horsepower of
240 now we could print BMW with the
cola and horsepower and we would get the
output for the BMW as well so classes
simplify the process of creating objects
or code that has to be duplicated a lot
because otherwise it would be quite
difficult to create many different CS
that all share the same attributes or
the same properties but I'm going to

remove all of this because what we're going to do next is learn about methods and instead of cola I'm going to change this to brand and the brand for Volvo is obviously going to be Volvo anyway with this we can actually go down inside our class and Define a function and a method is just a function that's inside a class anyway here we're going to create a method called drive and that's going to return none because what we're going to do is print that the self. brand is driving and what's important to note here is that we have self written everywhere and self refers to the instance of the class so here we have a Volvo and we are referring to its attributes so in other words self is the Volvo and if we had a BMW self would be the BMW it refers to the instance so if you actually want to refer to these values you're going to have to use the self keyword but we'll create one more method called get info and that's going to return none because we're going to print self. brand with self. horsepower horsepower and that's going to be our entire class now we have

a class with some attributes and some
methods so with that we can type in
Volvo do drive and
Volvo do get info these are methods that
we can use on that that instance and
when we run that we're going to see that
the Volvo is driving and that it's a
Volvo with 200 horsepower and this can
be used with any object of this class so
if we were to have a BMW once again
which will be of type car and that was
to be BMW with the 240 horsepower we can
now drive that
BMW as you can see the BMW is driving
and also you're not limited to just
using the information inside the class
but you can also Define your own
parameters in methods so after the self
keyword anything you type in is going to
be a normal parameter and I'm just going
to call this variable of type integer
just as an example to show you that we
can now use that and to refer to it you
do not use the self keyword because this
is included inside the method signature
so now the next time we actually use get
info we're going to have to also give it
a value and I'm going to remove the BMW

and run this so as you can see now we have 10 and Volvo with 200 horsepower we were able to use this information inside the method just by including it as a parameter but let's remove all of that because the final concept I want to talk about in this

video is the concept of Dunder methods and we're still going to have our Volvo car for this but uh right now I want to show you that if we were to print Volvo as it is what we're going to get back is a car object which is located at this memory address and that information isn't that useful to anyone who's not a programmer what if we actually want to get back the information that's inside the car well we can do so using a Dunder method and a Dunder method is just a double uncore method so here we're going to define the string Dunder method and this is going to return a string so what we're going to do is return the F string of self. brand and self. horsepower HP now the next time we try to print this Volvo what you're going to notice is that we're not going to get

that complex object back but we're
actually going to get the string that we
specified because now in any situation
we refer to it as a string it's going to
use the string dander method and another
example of a d method would be if we
wanted to add cars together we could use
the add Dunder method so this is going
to take self and other which is going to
be the object that we want to add to the
current
object and for this example we're just
going to return a string why not so what
we're going to do is return the F string
of self. brand
and other. brand and if you actually
want your code Editor to give you
suggestions for this you would have to
annotate this as self which can be
imported from the typing module so from
typing import self other of type
self and that will lead you to this code
completion which contains brand and
horsepower anyway to make this work
we're going to have to create another
car called BMW of type car which will
equal a
car called BMW with 240 horsepower next

we can print
Volvo plus BMW and this thunder method
is going to take care of the
functionality of plus which means that
when we run this we're going to get
Volvo and BMW as an output without this
Dunder method Python's not going to know
how to handle this because it does not
define the add Dunder method and it
actually wants you about that but if we
were to run it anyway once again it
would not know how to do that so you can
choose to Define that in your class and
then it will work properly and it's
worth mentioning that there are a lot of
Dunder methods worth researching because
obviously maybe in the future you will
want to multiply cars so for all of
these operations there's a Dunder method
for them such as multiply and once you
define that you'll be able to use this

comfortably but yeah those were 10
python Concepts that are quite important
and this video was just an introduction
to them I still recommend you study
these Concepts more thoroughly do some
more research online watch some other

YouTube videos regarding them but otherwise it should have been a great starting point for how the concepts in Python actually work so yeah let me know in the comment section down below whether you have any other questions or whether some Concepts should have had some more explanation but otherwise with all that being said as always thanks for watching and I'll see you in the next video