**Data types & Structures**

Phyton was initially designed to be simple and extendable. But most importantly it came into an age of large internet traffic and obviously, processing of traffic data. And this is where python got big, because it can handle all the text and numbers easily, which other languages have trouble. Python can handle combined data easily, and that’s one of its stronger assets.

But, let’s start with the basic data types

Let’s take a small step back to look at data types. Like any other language, python has integers, floats, complex, strings and Boolean data type.

Assign a float value to x in the next cell and execute. Now assign a integer to y and see what the operation result is.

In python, some or many types can be combined, which can be very useful, but sometimes, if we do not keep track, can lead to errors.

Complex numbers are identified by a j after a number.

And a thing to remember is that variables do not have to be initialized as any type, or even to keep a type through a program. They can be reassigned any time.

The reason I’m telling you this is because sometimes we reuse variables and forget about it.

So, here and there, it is good to test our variables, that they have the values and type we expect.

In the next cell define and print the type of variables indicated.

And one of the best variable types: **datetime**

This one is a cool one, although other languages have datetime functions, python’s is very functional and easy to use.

We will see the most basic things here, and use it later in the oceanography examples. But to see a full description and use see the link in the resources page.

First note that we need to **import the datetime** package. And that we are actually just importing two classes from that package: date and datetime.

Execute the code in the next cell to assign/define a date variable. Now, print both variables. What’s the difference? This is because they are different instances or variables of the same package. They do have different properties.

But print the type.

Now we use a new thing a .something and (). This indicates a function of the objects or variable. This is syntax is due to the fact that python is an object oriented language. Each of this variable is an object with particular functions. In this case, today and now. Or strftime.

For more details about it see this page. Shows enough examples to get familiar with datetime package.

**Data collections**

Now, those are the basic data types. Now we are going to talk about data collections.

Python has 4 basic ones, which can have any type of data, but with different properties.

**List –** it is probably the most basic type. It is ordered, changeable, and allows duplicates (as each element has an ordination. Technically as you saw before, it can contain different types of data. Mixed. But here we’ll make a list of strings.

Note the use of brackets to define a list.

To access the data in a list, you go by index, since it is an ordered collection. And just to make this clear, remember that the indices in python start in zero.

Try printing the first item of the list.

Lists are also changeable. Meaning, that we can change the values. For this we just assign a new value to an element. Try assigning the value orca to the list.

Now, to add an element… at the end, we use append.

Now try printing the list with the for loop in the next cell.

Note that animal now is each element of the list, and animal is string

Another thing to note is the + sign in print. That is a way to concatenate strings.

Now, let’s take a brief parenthesis to talk about **copying** list or any other object in python. Because this has to be done explicitly. And for this I mean, tell python we want a separate copy of the object. Otherwise, python just create an alternative name for our object.

This is illustrated in the next example.

Type or copy paste the next code. In the first one you are creating an actual new copy of mylist. This is a separate object. In the second one, you are making a fake copy. An alternative name to my list.

To test this. Now modify yourlist1 by adding an element or reassigning one.

Now print both my list and yourlist1 – and voila, only yourlist1 is modify. Mylist is the same. As we only modify the actual copy.

Now modify yourlist2, and then print it along with mylist. What happened now? See how yourlist2 is not a real new object, it is just an alternative name for mylist.

So, be careful when assigning copies. This is a common error in python since many other languages do not have this problem.

**Tuples**

Tupples are another ordered collection, but this time it doesn’t allow reassignations or changes. The next code assigns a tuple. Note the use of () instead of brackets. Execute it.and now print the tuple and its type.

And now try to print one element. And note that to access an element you’ll need to use square brackets [].

I know, confusing…. But differences are only in the definition of your object.

Now try changing a value.

This gives you an error.

And I’m going to make another parenthesis to talk about **errors**. Because you’ll encounter them left and right. They tend to look like this… and this is the short version. Usually at the end, it tells you the reason for the error. This is that item assignment is not supported by the tuple type. And it also points out the line in which the error occurs. (and by now you know how to add lines to your cell).

But sometimes, the error is more complex, and it can give you very very long error messages. Do not be afraid (I was the first few times – it is not nice to execute 2 lines of code and get two pages of errors). But it is because python is very explicit about the errors. Just look at the end or for the line with the error. Do not get intimidated. If you don’t immediately understand what the error is or how to fix it, that’s when you go and ask google. You can copy past the error message… starting with python and usually somebody already found that problem and fix it.

Ok, back to tuples. We will use or rather define tuples very rarely. But some functions give you tuples back and you need to know their properties. Mainly, that they cannot be changed. So if you intend to, you might need to copied to a list or another type of object.

**Sets**. Unordered, unindexed, and doesn’t allow duplicates.

Yet another collection, that frankly I haven’t use as stand-alone collections yet. But! They are widely used to pass data, arguments, or other values to functions.

Take a look at the code below. and not the {} brackets. You’ll note that those brackets are used inside functions to pass … yes, sets of arguments.

Now, just for fun… try printing an element of the set. What happened? Can you guess why?

**Dictionaries**

This collection is not new to python, other languages have them… but python does a pretty good job with them. They are also unordered collections, as sets, which means they have no numerical index, instead they do have a pair of keys and values.

Try it out with the code below. execute to assign and print it out.

Now, to add or change an element, instead of an index inside the square brackets, you use the ‘key’. Same to access and element. Give it a try.

**Objects**

There are more collections, specific to packages, but those are the basic blocks, and we will talk about other types as in the next notebook. But first a few words about python object oriented characteristic.

Python is an object-oriented programming language. This means that pretty much every thing you interact with is an object. Which in turn is an instance of a define class. There is a class for each type of variable, and set. And when you make a variable a type of object, you’re creating an instance of that class. So, variables are objects of a certain class.

And as all classes in all object-oriented programing languages, they have specific attributes and methods.

Properties or attributes are values that the class intrinsically have. And are access with a dot and the name of the attribute after the object. We do not interact with those very often, except for the defined objects later while using netcdf.

A method on the other hand are functions that belong to that class, and they are accessed also with a dot and a name, but also with parenthesis and if needed, arguments.

We’re not going to create classes or functions but we will use them extensively.

Code in the next cell will create an instance of the class today, which belongs to the date class of the datetime library or package.

So, there are attributes for the instance today, and there are methods. Here we are just exemplifying them.

You already use methods in this and previous notebooks, and we will use them in the rest of the workshop. But just as a little bit of practice. Let’s try the next three methods.

The first one we already use, and the second use mydict defined above.

Q: what name goes before the .method()?

For the third, you need to define a new string variable that contains two or more commas, then use a method split to separate the text using the comma as separator. What kind of variables you get?