11) List and Dictionary

Agenda

- List
- Dictionary

Lists

So far we have worked with **primitive data types** that can only store **one piece** of data: strings, integers, floats, and booleans.

Let's see some examples:

```
some_num = 4  # integer (whole number)
some_decimal = 3.14  # float (decimal number)
some_str = "Hello, SmartNinja!"  # string
human = True  # boolean (either True or False)
```

But there's a **data type**, that can hold **multiple pieces of data**. It's called a **list.** Let's see how it looks like:

```
some_list = [1, 34, 12, 17, 87]
print(some_list)

another_list = ["tesla", "toyota", "nissan"]
print(another_list)

mixed_list = [22, "elon", True, "SmartNinja", 3.14]
print(mixed_list)
```

Sorting

Lists have some **useful features**, like sorting:

```
some_list = [1, 34, 12, 17, 87]
some_list.sort()
print(some_list)
```

For-in loop

You can even go through a list with a for loop and print each item **separately**:

```
another_list = ["tesla", "toyota", "nissan"]

for item in another_list:
    print(item)
```

Pretty cool, huh? :)

But where does this come in handy?

For example in our "Guess the secret number" game. Right now we are storing **only one score** in the score.txt file - the best one.

But what if we'd want to store **all of the results** and then sort them from the best to the worst? And print out, for example, the top 3?

We could definitelly use a list. Let's do it!

Start from scratch

Let's **delete** all the file code that we added before, so our game looks as it did in the **beginning** of the lesson:

```
import random

secret = random.randint(1, 30)
attempts = 0

while True:
    guess = int(input("Guess the secret number (between 1 and 30): "))
    attempts += 1

    if guess == secret:
        print("You've guessed it - congratulations! It's number " + str(secret))
        print("Attempts needed: " + str(attempts))
        break
    elif guess > secret:
        print("Your guess is not correct... try something smaller")
    elif guess < secret:
        print("Your guess is not correct... try something bigger")
```

You can also create a new Python file and add this code in it (and keep the old code in the previous file).

Create the score_list.txt file

Open the `score_list.txt` file and add an empty list in it:

```
[]
```

Read the text file

Let's open the text file inside our Python script and read the contents of the file. Add this piece of code just above the while loop:

```
with open("score_list.txt", "r") as score_file:
    score_list = json.loads(score_file.read())
    print("Top scores: " + str(score_list))

while True:
    # the rest of the code...
```

Note the `json` library that we used. This library will help us seamlessly convert between a list and a string (we can only store strings in a text file).

So make sure you import the `json` library in the top of the Python script:

```
import json
```

If you run the program now, it will show you an empty score list. So let's make our program add some data into it.

Insert data into a list

We add data to a list using the `.append()` method. Add this line of code after the `if guess == secret:` line:

```
if guess == secret:
    score_list.append(attempts)
```

Save the list into the text file

Now let's save the score list back into the `score_list.txt` file:

```
if guess == secret:
    score_list.append(attempts)

    with open("score_list.txt", "w") as score_file:
        score_file.write(json.dumps(score_list))
```

Play a few games. You'll see that the scores list is growing bigger and bigger.

Sort the list

As you can see, the list is not sorted from the lowest to the biggest number. So let's sort it and show only the top 3 scores. Add this code just above the while loop:

```
score_list.sort()
print(score_list[:3])

while True:
    # the rest of the code...
```

As you can see, `[:3]` helps us show only the first three elements from a list.

Pretty cool, right? But this is not all. We can make our data storage even better! How? With dictionaries!

> *See the complete example here (https://github.com/smartninja/wd1-py3-exercises/tree/master/lesson-11/guess-secret-number-list).*

Dictionary

The last thing we'll learn today is called a dictionary.

It's a data structure similar to a list in a way that it can hold many pieces of data. But with one *key* difference - you can actually name these pieces of data.

Let's say you'd like to have a list of features that some object has. For example: a **box.** A box has **length**, **width** and **height.** Let's write this using a Python list:

```
box = [20, 45, 30]
```

Ok, but how do we know which of these numbers represents length, width or height?

Here comes the power of dictionaries. In a dictionary, each piece of data (called **value**) has an associated **key.** So a box dictionary would look like this:

```
box = {"height": 20, "width": 45, "length": 30}
```

This makes box features much more clear.

> *Even clearer version would be to have keys also hint a unit, like `"height_cm"` (centimeters) or `"height_in"` (inches).*

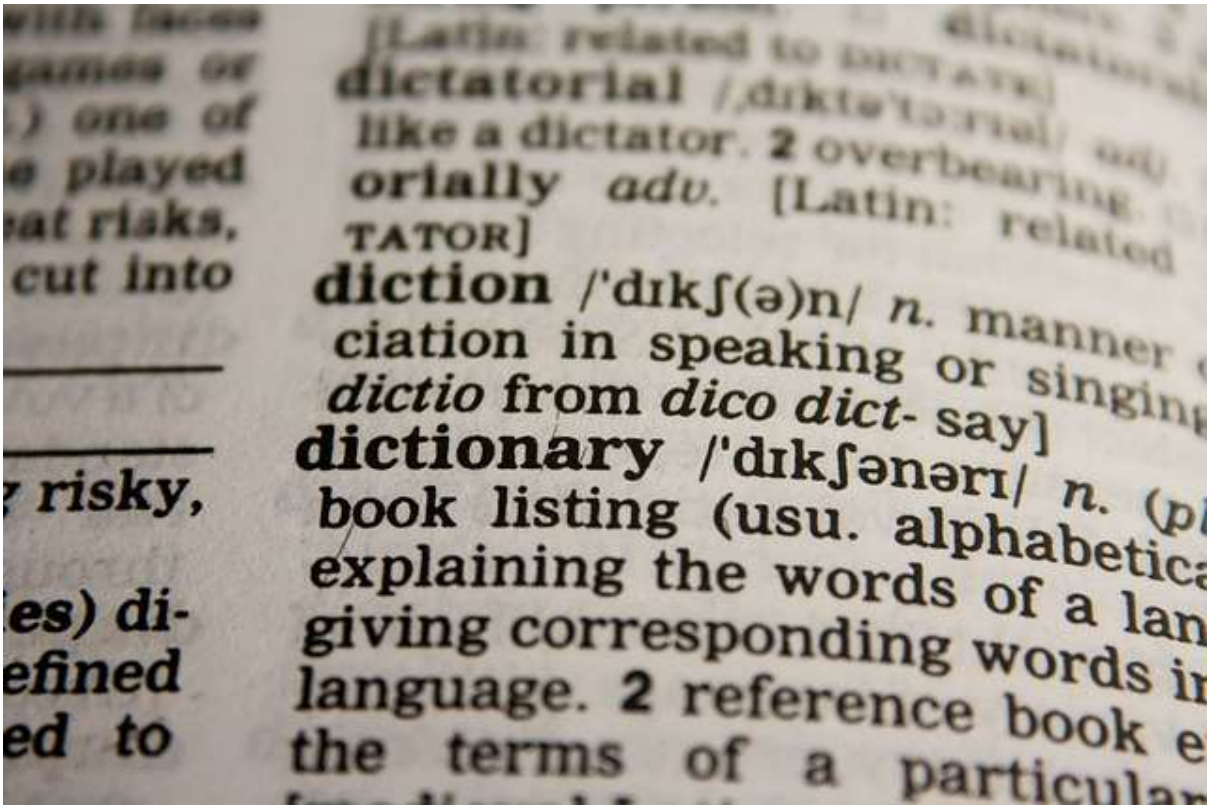
We can access the data in a dictionary like this:

```
print(box["height"])
```

Or like this:

```
print(box.get("height"))
```

Data in a dictionary is stored in a so called "key-value" format. Which is the same as in a paper dictionary, that each of us has already seen:



A **word** is a **key**, and the **description** of the word is its **value**.

Also note that Python lists use **square** brackets `[]` , while dictionaries use **curly** brackets `{}` .

So when to use list and when to use a dictionary?

If you want to group data of the same kind, you'd use a list. Example: a list of European cities:

```
eu_cities = ["Vienna", "Malaga", "Budapest", "Cologne", "Zagreb"]
```

But if you'd like to group data of different kinds, you'd use a dictionary:

```
jordan = {"first_name": "Michael", "last_name": "Jordan", "age": 56, "games_played": 1072, "tot
```

> ***Important:*** *Note that keys in a dictionary must be unique. There can't be two identical keys in the same dictionary.*

How can we use this in our game?

Let's say we want to save the date when each score was made.

We can get the current date and time using this neat Python library:

```
import datetime

current_time = datetime.datetime.now()

print(current_time)
```

In order to store the current time together with a score, we'll use a dictionary:

```
score_data = {"attempts": attempts, "date": datetime.datetime.now()}
```

And because we will have many `score_data` elements (one for each game played), we'll store them all in a list. It will look something like this:

```
[{"attempts": 6, "date": "2019-03-01 12:30:56.198449"}, {"attempts": 5, "date": "2019-03-03 18:
```

So we will combine the power of lists and dictionaries together. Let's build the thing!

Empty the score_list.txt

First let's empty the `score_list.txt` . It should have an empty list in it:

```
[]
```

Append a dictionary

Next let's change the line where we have `score_list.append(attempts)` . Change it into this:

```
score_list.append({"attempts": attempts, "date": str(datetime.datetime.now())})
```

Replace the list sorting

And finally let's replace these list sorting lines of code:

```
score_list.sort()
print(score_list[:3])
```

With this for loop:

```
for score_dict in score_list:
    print(str(score_dict["attempts"]) + " attempts, date: " + score_dict.get("date"))
```

> *We removed sorting the list from the lowest to the highest number of attempts, but you'll do this for a homework.*

Now let's play the game a few times. As you can see in the `score_list.txt` , our database now includes both the number of attempts and the date each game was played.

> *See the complete example here (https://github.com/smartninja/wd1-py3-exercises/tree/master/lesson-11/guess-secret-number-dict).*

Q&A

Any question?

> ***Did you know:*** *in many other programming languages lists are called arrays, although usually arrays can't hold data of a different type, while a Python list can. An array structure also exists in Python, but is rarely used.*

If there's enough time after Q&A, students can start working on their homework.

Homework 11.1: Add some more data in the dictionary

For now our game only stores the number of **attempts** and the **date**. Let's also store the **name** of the player and the secret number in each game.

> *Solution (https://github.com/smartninja/wd1-py3-exercises/tree/master/lesson-11/guess-secret-number-hw1).*

Homework 11.2: Add unsuccessful guesses into the dictionary

During the game, collect user's unsuccessful guesses and then store them in the dictionary under the name `"wrong_guesses"`.

> *Hint: you can store a list in a dictionary. ;)*
>
> *Solution (https://github.com/smartninja/wd1-py3-exercises/tree/master/lesson-11/guess-secret-number-hw2).*

Bonus homework 11.3: Print out only the top 3 results

Now that we removed the `score_list.sort()` line (because it doesn't work with dictionaries in the list), we have to find another way to sort the scores.

There are many ways how to do it. Use your imagination (and Google) and try to figure out at least one of them. :)

> *One of the possible solutions (https://github.com/smartninja/wd1-py3-exercises/tree/master/lesson-11/guess-secret-number-hw3).*

Bonus homework 11.4: Forensics program

> *This exercise is optional to do.*

CSI has asked you to build a forensics program for them. More instructions here: https://github.com/smartninja/wd1-py3-exercises/tree/master/lesson-11/CSI (https://github.com/smartninja/wd1-py3-exercises/tree/master/lesson-11/CSI).

When you finish, push your code on GitHub (https://github.com/) and share it on the SmartNinja forum.

> *There are many possible ways how to solve this exercise. See some of them here (https://github.com/smartninja/wd1-py3-exercises/tree/master/lesson-11/CSI).*

Useful links

- Working with Python lists (https://www.w3schools.com/python/python_lists.asp)
- Working with Python dictionaries (https://www.w3schools.com/python/python_dictionaries.asp)
- Big-O: How Code Slows as Data Grows (https://youtu.be/duvZ-2UK0fc?t=46)