Curso python udemy

Cheatsheet: Data Types

- Integers are used to represent whole numbers:

```
rank = 10
eggs = 12
people = 3
```

- Floats represent decimal numbers:

```
temperature = 10.2
rainfall = 5.98
elevation = 1031.88
```

- Strings represent text:

```
message = "Welcome to our online shop!"
name = "John"
serial = "R001991981SW"
```

- Lists represent arrays of values that may change during the course of the program:

```
members = ["Sim Soony", "Marry Roundknee", "Jack Corridor"]
pixel_values = [252, 251, 251, 253, 250, 248, 247]
```

- Dictionaries represent pairs of keys and values:

```
phone_numbers = {"John Smith": "+37682929928", "Marry Simpons":
"+423998200919"}
volcano_elevations = {"Glacier Peak": 3213.9, "Rainer": 4392.1}
```

- Keys of a dictionary can be extracted with:

```
phone_numbers.keys()
```

- Values of a dictionary can be extracted with:

```
phone_numbers.values()
```

- Tuples represent arrays of values that are not to be changed during the course of the program:

```
vowels = ('a', 'e', 'i', 'o', 'u')
one_digits = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

- You can get a list of attributes of a data type has using:

```
dir(str)
dir(list)
dir(dict)
```

- You can get a list of Python builtin functions using:

```
dir(__builtins__)
```

- You can get the documentation of a Python data type using:

```
help(str)
help(str.replace)
help(dict.values)
```

Tip: Converting Between Datatypes

Sometimes you might need to convert between different data types in Python for one reason or another. That is very easy to do:

- From tuple to list:

```
cool_tuple = (1, 2, 3)
cool_list = list(cool_tuple)
cool_list # [1, 2, 3]
```

- From list to tuple:

```
cool_list = [1, 2, 3]
cool_tuple = tuple(cool_list)
cool_tuple # (1, 2, 3)
```

- From string to list:

```
cool_string = "Hello"
cool_list = list(cool_string)
cool_list # ['H', 'e', 'l', 'l', 'o']
```

- From list to string:

```
cool_list = ['H', 'e', 'l', 'l', 'o']
cool_string = str.join("", cool_list)
cool_string # 'Hello'
```

As can be seen above, converting a list into a string is more complex. Here str() is not sufficient. We need str.join(). Try running the code above again, but this time using str.join(«—», cool_list) in the second line. You will understand how str.join() works.

Cheatsheet: Operations with Data Types

- Lists, strings, and tuples have a positive index system:

```
["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
   0      1      2      3      4      5      6
```

- And they have a negative index system as well:

```
["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
```

```
  -7     -6     -5     -4     -3     -2     -1
```

- In a list, the 2nd, 3rd, and 4th items can be accessed with:

```
days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
days[1:4]
Output: ['Tue', 'Wed', 'Thu']
```

- First three items of a list:

```
days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
days[:3]
Output:['Mon', 'Tue', 'Wed']
```

- Last three items of a list:

```
days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
days[-3:]
Output: ['Fri', 'Sat', 'Sun']
```

- Everything but the last:

```
days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
days[:-1]
Output: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat']
```

- Everything but the last two:

```
days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
days[:-2]
Output: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri']
```

- A dictionary value can be accessed using its corresponding dictionary key:

```
phone_numbers = {"John":"+37682929928","Marry":"+423998200919"}
phone_numbers["Marry"]
Output: '+423998200919'
```

Cheatsheet: Functions and Conditionals

- Define functions:

```
def cube_volume(a):
    return a * a * a
```

- Write if-else conditionals:

```
message = "hello there"

if "hello" in message:
    print("hi")
else:
```

```
    print("I don't understand")
```

- Write if-elif-else conditionals:

```
message = "hello there"

if "hello" in message:
    print("hi")
elif "hi" in message:
    print("hi")
elif "hey" in message:
    print("hi")
else:
    print("I don't understand")
```

- Use the and operator to check if both conditions are True at the same time:

```
x = 1
y = 1

if x == 1 and y==1:
    print("Yes")
else:
    print("No")
```

- Use the or operator to check if at least one condition is True:

```
x = 1
y = 2

if x == 1 or y==2:
    print("Yes")
else:
    print("No")
```

- Check if a value is of a particular type with isinstance:

```
isinstance("abc", str)
isinstance([1, 2, 3], list)
# or directly:

type("abc") == str
type([1, 2, 3]) == lst
```