3.

Basic data structures in python

Hands-on Tests!

- ✓ *Integers*
- ✓ *Characters and Strings*
- ✓ *Booleans*
- ✓ *Lists*
- ✓ *Dictionaries*
- ✓ *Sets*
- ✓ *Tuples*

Contents

Numbers in Python

- We work with two main number types:
    - Integers(int): 1, 3 , 121 , …
    - Floating point numbers(float): 1.2 , 0.1 , 101.0
- Let’s try some simple math in python!

But what do these numbers represent? How can we save them or assign them?

Variable Assignments

- We can assign values to variables in python:
    - My_weight = 75.3
    - My_height = 170 + 4
    - Type()
    - Let’s see some examples in the code!

Some Rules for Variable Names...

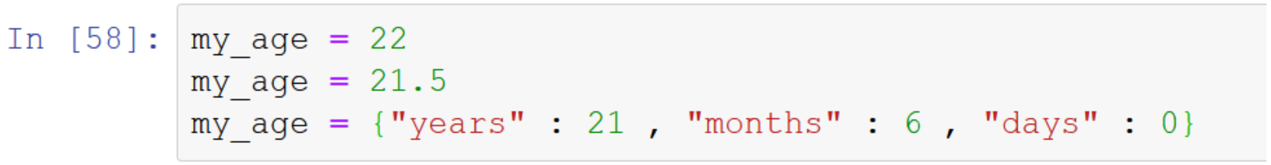
- There mustn’t be any spaces in names: × My Age = 22 ✓ My_Age = 22
- Names must not start with numbers: × 2Age = 22 ✓ Age2 = 22
- Names must not contain any of these symbols: “ , : <> / ? | \ * - + () ~ ! @ # $ % ^ &
- Avoid using reserved names in python : × for = 22 × if = 22 × int = 22
    - You will learn reserved names during the course. Don’t worry about it!

Try to use the same naming format during all parts of your program.

Some Clean Code Tips...

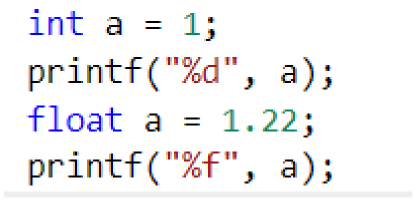
Dynamic Typing in Python

- Python is a **Dynamic-Typed** language,
- It means, it’s possible to reassign same variables to different types of data:



```
In [58]: my_age = 22
         my_age = 21.5
         my_age = {"years" : 21 , "months" : 6 , "days" : 0}
```

- Some languages like C, C++ are **Static-Typed**



```
int a = 1;
printf("%d", a);
float a = 1.22;
printf("%f", a);
```

Try <u>not</u> to use the same variable names for different types of data.

Even in Python!

Some Clean Code Tips...

Strings

- Strings are sequence of characters,
- Both “ and ‘ are acceptable
  - My_name = ‘Kaveh’
  - My_family_name = “Masoumi”
  - My_course_name = “Programming with python”

Indexing Strings

- With the notation variable_name[index], we can access to the characters
  - My_name = “Kaveh”
  - My_name[2]
- Index starts with 0 and continues till n – 1 and also from –(n-1) to 0

Character	‘K’	‘a’	‘v’	‘e’	‘h’
Index Positive	0	1	2	3	4
Index Negative	-5	-4	-3	-2	-1

Slicing Strings

- You can also grab a slice of a string
    - Variable_name[Start : Stop : Step]

Start: $Starting\ index$

Stop: $Ending\ index\ -1$

Step: $Size\ of\ every\ jump$

    - My_name[0:3:2] → “Kv”
- Strings have helpful methods such as .split() , .find() and etc.
- Also remember : **String are immutable**

Now let’s get back to the code...

Contents

Lists

- Ordered sequence of objects :
    - Mylist = [1 , True , “Hey” , 2.1 , [1,2,3]]
    - Use the same indexing and slicing notation as Strings
    - Some useful methods such as .append() , .pop() , .sort() …
    - Let’s get back to the code…

Contents

Dictionaries

- Use the notation { }
- Unordered sequence of key-value objects (no sorting)
- Dictionaries are like lists but they don’t use indexes
- Dictionaries use key-value mapping
    - My_dictionary = {“key1” : “value1” , “key2” : 2 , 5 : 7 , “list” : [1,2,3]}
    - My_dictionary [“key1”] → value1
    - My_dictionary[5] → 7
- Some useful methods like .values() , .keys() , .items() and etc.
- Time to see the concepts in code…

Contents

Sets

- Unordered Unique Objects
  - Myset = set()
  - Useful methods like add() , remove() , pop() , copy() …
  - Could be used to eliminate repetitive elements in list
    - Mylist = [1,1,1,1,1,1,1,2,3,3,3,3]
    - Myset = set(Mylist)
    - Myset → (1 , 2 , 3)
- Now let's try some examples in practice…

Contents

Tuples

- Tuples are like lists → Ordered sequence of objects
- But Tuples are <u>**immutable**</u> → A good question, why immutablity?
- Useful methods such as count() and index()
- The notation to use tuples is ()
    - My_Tuple = (1 , 1 , “Hey” , True)
- Let’s get back to code…

Thanks!

Got any questions or suggestions?

Here's some contact info:

@KMasoumi