PyLadies

Vienna 18.1.2020

Who?

International mentorship group with a focus on helping more women become active participants and leaders in the Python open-source community.

Our mission is to promote, educate and advance a diverse Python community through outreach, education, conferences, events and social gatherings.

Agenda for today

---

1. Python Fundamentals III
2. Working with files
3. Git
4. Exercises, networking, discussion of own projects
5. Join us for a beer (or anything else) later

Recap

- basic data types - list, string, int, float, etc…
- Loops - for and while
- Lists
- Turtle :)
- write your own functions

Dictionaries

- New basic data type
- Inside list multiple values
- `me = {'name': 'Tyna', 'country':'CZ', 'colours':['red','black']}`
- Key
- Value

Dictionaries - operations with them

- Similar to lists, but instead of index, use name of key
- Get the values - `me['name']`
- Change value - `me['colours'] = ['red', 'black', 'yellow']`
- Add value - `me['language'] = 'Python'`
- Delete values - `del me['colours']`

Dictionaries - iterations

- Iteration over dictionary `for key in me:`
  `print(key)`
- What does it return?
- If you want to see values - use me.values()
- If you want pairs of values - use me.items()

```
for key, value in me.items():
    print('{}: {}'.format(key, value))
```

Dictionaries - notes

- Classic usage example - lookup table (phone book)
- During for loop, it’s not possible to add and delete values
- But it is possible to change values with existing keys
- Create dictionary - two ways:
  - as shown before: `{'name': 'Tyna', 'country':'CZ', 'colours':['red','black']}`
  - using dict() function - anything iterable with pair values or another dictionary - `data = [(1, 'one'), (2, 'two'), (3, 'three')]`
  - `number_names = dict(data)`

Exercises

1) (one from practice list) · Write a function, which for an argument number n creates and return a dictionary, where keys will be numbers from 1 to n and values will be their exponents. Example:

```
>>> exponent(4)   {1: 1, 2: 4, 3: 9, 4: 16}
```

2) Write a function, which will return sum of keys and sum of values. You can use dictionary from previous task:

```
>>> sum_key_value(exponent(4))      (10, 30)
```

Exercises

1) Write a function, which will receive string as argument and return dictionary with characters from a string as keys and their occurrence as values:

```
>>> char_count("hello world")        {'h': 1, 'e': 1, 'l': 3, 'o': 2, ' ':
1, 'w': 1, 'r': 1, 'd': 1}
```

2) Write a function which receives dictionary as argument and will print keys and values on separate lines

Encoding, separator, ...

- Encoding is used as representation of characters in some encoding system
- UTF-8 covers most of the symbols used in normal life including emoji symbols and special characters
- It is a nice practice to save files in standard encodings to make it easier for other users
- Separator - can be used to separate lines in text, eg: ‘,’, ‘;’, ‘   ’

How to work with files

- Create in your workshop folder file with a short poem or music lyrics inside and save it as poem.txt

```
file = open('poem.txt', encoding='utf-8')

content = file.read()

file.close()

print(content)
```

Reading and writing text file

- To not forgot to close file, use automatic closing:

```
with open('poem.txt', encoding='utf-8') as file:
    content = file.read()
print(content)
```

```python
with open('second_text.txt', mode='w', encoding='utf-8') as file:
    print('I am here,', file=file)
    print('Here am I', file=file)
```

Modes for files

- mode = ‘w’ - write only, if the file doesn’t exist python creates it. Overwrites the content
- mode = ‘a’ - append
- mode = ‘r’ - read only
- mode = ‘w+’ - write and read
- mode = ‘wb’ - write in binary format

and many others...

Iterations over file

```
with open('poem.txt', encoding='utf-8') as file:
    for row in file:
        print('    ' + row)
```

- In order to get rid of empty spaces and new lines

  `row = row.rstrip()`

JSON

- While working with data structures in python is easy, when you will close python, they will be lost.
- In order to save them and store for later usage, we can save them into files
- JSON is a popular method
- Functions in package json

```
import json
```

Example

---

create more complex dictionary:

```
data = {
    'name': 'Anna',
    'city': 'Vienna',
    'languages': ['english', 'python'],
    'age': 25,
}
```

```python
code = json.dumps(data)

with open('data.json', 'w') as file:
    print(code, file=file)

with open('data.json', 'rb') as file:
    code = file.read()

data = json.loads(code)
```

Other types of files

- You can also use other type of files to store your values
- TOML
- YAML
- Need to use special libraries to work with these types
- csv files - `import csv`

```
with open('text_file.txt') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=';')
```

- csv files are handled better with pandas library

Exercises

---

1) Write program which will print out text from file poem.txt in CAPITAL LETTERS
2) Write a Python program to append text to a file and display the text
3) Write a Python program to count the frequency of words in a file

Resources and materials

- advent of code - adventofcode.org
- hackerrank - hackerrank.com
- Django Girls - django tutorial (in April in Vienna)
- https://www.practicepython.org
- Nice Python exercices at one place https://github.com/tystar86/python_exercises/tree/master/Tasks
- https://automatetheboringstuff.com
- https://diveintopython3.problemsolving.io

Next topics

Graphics

Games in Python

Testing

Flask

fill the form please :) → https://forms.gle/UtfgVGe6AhhRwx539

Thank you and see you next time

Coding session - 29.1.2020

Next workshop - 15.2.2020