Exercises - Part Bonus

October 19, 2020

0.0.1 StackOverflow

You might have noticed that whenever you try to search the internet for a question about your code, `StackOverflow` shows up high in the search results. Every month, over 120 million people visit the site. It is the most important Q&A forum for coding. When you have an account, you can ask your own question. You will probably have an answer within a couple hours, if not minutes.

Exercise: - Sign up for an account at https://stackoverflow.com/users/signup

0.0.2 GitHub

`GitHub` is a platform for developers. You can host (and showcase) your code, manage projects, and build software together with other developers. It is *the* place for open source software. With an account, you can upload your code and show your work to others. You can also "star" your favourite projects, so you can receive updates about them.

Exercise: - Sign up for an account at https://github.com

0.0.3 Exercises on the web

The following websites have lots of great exercises which will help you level up. You can monitor your progress and participate in challenges. An excellent and fun way of improving your skills!

- PracticePython
- HackerRank
- LeetCode
- Codingame
- Exercism

0.0.4 Try Except

With `try / except` you can handle errors. This is especially useful when dealing with external or user data.

```
# Example
try:
    x = int(input("Please enter a number: "))
    print("You entered:", x)
except ValueError:
    print("Oops!  That was no valid number.  Try again...")
```

Exercise: 1. Follow the example to fix the bug in the code below. If the user gives `0` as input, the program crashes. It should print a warning like `"Entered a zero. Try again"`. *Hint*: note that this is not a `ValueError`, but another one.

```
x = int(input("Please enter a (non-zero) number: "))
print("The outcome is" , 100 / x)
```

Extra challenge: currently you need to run the cell every time if you want to test various user inputs. Create a loop that keeps running until you have the input you need and then stops (breaks).

[ ]:
```
# Exercise 1
x = int(input("Please enter a (non-zero) number: "))
print("The outcome is" , 100 / x)
```

[ ]:
```
# Extra challenge
```

0.0.5 Datetime

Working with dates and times can be tricky sometimes. Calculations and timezones are not always obvious for a computer. In this exercise you will work with Python's `datetime` object, which has some convenient capabilities.

```
from datetime import datetime
now = datetime.now()
year_start = datetime(2020,1,1)
print("Time elapsed since the start of the year:", now - year_start)
```

Exercises: 1. Start with the import: `from datetime import datetime` 1. Create a variable `birthday` with your birthday as a datetime object. Print `birthday` for inspection. 2. Calcuate and print the time elapsed since your birthday

Extra [difficult] challenge: calculate your age in years

[ ]:
```
# Exercise 1
```

[ ]:
```
# Exercise 2
```

[ ]:
```
# Exercise 3
```

[ ]:
```
# Extra challenge
```

0.0.6 Open files

```
# Example writing
with open("zenofpython1.txt", "w") as f:
    f.write("Beautiful is better than ugly.")

# Example reading
with open("zenofpython1.txt", "r") as f:
    text = f.read()
print(text)
```

Exercise: 1. Create a program that creates a new text file called `zenofpython3.txt` and containing the following sentence: `"Simple is better than complex."` 2. Inspect the file is in the same folder as your Jupyter Notebook 3. Read the file again and print it's contents to the screen

Extra challenge: besides the options `"r"` and `"w"`, the `open()` function has other modes for opening files. Check out the documentation and try them out.

```
[ ]: # Exercise 1
```

```
[ ]: # Exercise 2
```

```
[ ]: # Exercise 3
```

0.0.7 List Comprehension

```
# Example
ingredients = ["SPAM", "BaCon", "haM", "EggS"]

ingredients_lowercase = []

for i in ingredients:
    ingredients_lowercase.append(i.lower())
```

Exercise: 1. Assign the variable `names = ["stark", "nelson", "foster", "rogers", "wilson", "hardy", "brewer", "ray", "rich"]` 2. Use a for loop to create a new list `names_capitalized` that contain all the names starting with a capital ("Stark", "Nelson", "Foster", etc) and print the new list to the screen 3. Create again the new list `names_capitalized`, but now using list comprehension. Print the result to the screen.

Extra challenge: use `list comprehension` to create a list of all the even numbers between 0 and 100. Hint: search the internet on how to use the `modulo` operator.

```
[ ]: # Exercise 1
```

```
[ ]: # Exercise 2
```

```
[ ]: # Exercise 3
```

```
[ ]: # Extra challenge
```

0.0.8 Pandas: Tabular data

```
# Example
import pandas as pd
df = pd.read_csv("https://raw.githubusercontent.com/pythonsherpa/data/master/payments.csv")
df.head()
```

Exercise: 1. Run the code (from the example) to create a variable `df` with payments data. 2. Inspect the first rows with `df.head()` 3. Use the `.info()` method to get more information about the dataset 4. Use the `.describe()` method to get descriptive statistics 5. Run the cell to print a bar chart of the total invoiced amounts per payment provider

Documentation: read_csv

Extra challenge: follow the "10 minutes to pandas" tutorial on: https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html

Extra challenge 2: use `pandas` read_excel to read an Excel file on your computer. The code is given, you only need to change the path to a file on your computer. Use methods like `.head()`, `.info()` and `.describe()` to inspect the data. Try out some of the techniques described in the "10 minutes to pandas" tutorial.

```
[ ]: # Exercise 1
     import pandas as pd
     df = pd.read_csv("https://raw.githubusercontent.com/pythonsherpa/data/master/
      ↪payments.csv")
```

```
[ ]: # Exercise 2
```

```
[ ]: # Exercise 3
```

```
[ ]: # Exercise 4
```

```
[ ]: # Exercise 5
     %matplotlib inline
     df.groupby('payment_method')["invoice_amount"].sum().nlargest().
      ↪plot(kind="bar");
```

```
[ ]: # Extra challenge
     import pandas as pd
     df = pd.read_excel("C:/Users/John/sample_data.xlsx")  # change this to a file␣
      ↪on your computer
     df.head()
```

0.0.9 Pandas: Scraping Wikipedia

```
# Example
import pandas as pd
url = "https://en.wikipedia.org/wiki/List_of_S%26P_500_companies"
tables = pd.read_html(url)
snp500 = tables[0]
snp500.head(10)
```

Exercises 1. Import the `pandas` package 2. Assign `url = "https://en.wikipedia.org/wiki/EURO_STOXX_50"` 3. Use Pandas' read_html function to get the data: `tables = pd.read_html(url)` 4. Your variable `tables` is a list of `dataframes`. There are actually 5 tables on the website. Find the desired table by using indexing (like `tables[1]`). Assign the table to a new variable `eurostoxx50`. 5. Inspect the result with `eurostoxx50.head()`

Documentation: pandas

```
[ ]: # Exercise 1
```

```
[ ]: # Exercise 2
```

```
[ ]: # Exercise 3
```

```
[ ]: # Exercise 4
```

```
[ ]: # Exercise 5
```