04_Assignment_Python

January 18, 2023

1 Assignment 04

Python Basics IV - Functions, Unity Testing, and Logging This tutorial was written by Terry L. Ruas (University of Göttingen). The references for external contributors for which this material was anyhow adapted/inspired are in the Acknowledgments section (end of the document).

This notebook will cover the following tasks:

1. Lambda functions
2. List comprehensions
3. Unity Test
4. HTTP Request
5. Logging
6. Download File

1.1 Task 01 – Lambda functions

Python supports lambda functions as a handy way to define small, anonymous, i.e., unnamed, functions inline. The basic syntax for lambda functions is:

lambda parameter1, parameter2, ... : **expression**

Use a lambda function only to retain the even values in an array of integers. Test your function with an input array of your choosing. Print the input array and the filtered output array to stdout.

[3]:
```
a = range(2,20)
result = list(filter(lambda x: x% 2 == 0, a))
print(result)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18]
```

1.2 Task 02 – List comprehensions

Python supports list comprehension. The basic syntax of list comprehensions is:

L = [<elem> for <elem> <Condition>]

Use list comprehensions to write a Python function *remove_long_words()* that: - accepts a sentence s and an integer n as input parameters - uses the *split()* function of String objects to split the sentence into words - stores the individual words in a list - removes all words that are longer than n characters from the list, thereby creating a new list - prints the list to stdout

[16]:
```
#remove long words list

s = "Der Hubschrauberlandeplatz ist besetzt"
n = 3

s.split(" ")
```

[16]:
```
['Der', 'Hubschrauberlandeplatz', 'ist', 'besetzt']
```

[19]:
```
s = "Der Hubschrauberlandeplatz ist besetzt"
n = 4

sentence = s.split(" ")

my_list = [x for x in sentence if len(x) <n]

print(s)
print(my_list)
```

```
Der Hubschrauberlandeplatz ist besetzt
['Der', 'ist']
```

1.3 Task 03 – Unity Test

The following algorithm in Python converts numbers in decimal representation to binary. 1. Develop a unit test that checks for values in the interval [-1,3] whether the algorithm returns the expected results. 2. Adjust the algorithm, so it passes the unit test developed in 1). Rename the function to *decimal_to_binary_correct()*

[23]:
```
import math
def decimal2binary(n):
    # function to convert decimal integers to binary
    x = []
    while n > 0:
        x.append(n % 2)
        n = math.floor(n/2)
    return x[::-1]
```

[ ]:
```

```

1.4 Task 04 – HTTP Request

Working with HTTP connections is essential for many data gathering tasks. The Python library *urllib* provides all functionality we need. Write a Python function *open_url(url)* that: - uses urllib to establish a HTTP connection to an arbitrary website - retrieves and prints the first 200 characters of the html resource, i.e. the html source code, of the chosen website - handles the exceptions thrown by the *urllib.request* function

FYI: The basic syntax for exception handling in Python is as follows:

```
try:
    ...
    return ...
except SomeError1 as e:
    # error-specific exception handling
except SomeError2 as e:
    # error-specific exception handling
except
    # general exception handling
```

[19]:
```
import requests
url = "https://www.instagram.com"

requests.get(url)
```

[19]:
```
<Response [200]>
```

[18]:
```
import urllib.request

def open_url(url):
    try:
        request_url = urllib.request.urlopen(url)
        content = request_url.read()
        string = content.decode("utf-8")
        print(string[:200])
    except Exception as e:
        print(f"Exception: {e}")
        print(f"URL: {url}")

open_url(input("Please enter a URL: "))
```

```
Enter a URL please:  www.intagram.com

Exception: unknown url type: 'www.intagram.com'
URL: www.intagram.com
```

1.5 Task 05 – Logging

The logging module in Python provides functionality for logging and debugging purposes. Use the logging module to extend the error handling for the function that you implemented to establish a HTTP connection (Task 4). All exceptions thrown by your function shall be logged as errors.

To accomplish the task: - write a Python function *init_log(file_name, file_mode, level, format, date_format)* that initializes a custom log file to which all debugging information and errors are appended using a format that includes the date, time, level and the message of the logging event - log occurring errors by calling *logging.error(…)* - close the log after completing your task by calling *logging.shutdown()*

If you choose not to complete Tasks 4, test the logging functionality with a few examples of your own.

[1]:
```
import logging
import urllib.request

def init_log(file_name, file_mode, level, format, date_format):
    logging.basicConfig(level=logging.ERROR, filename="log.log", filemode="w",
                        format="%(asctime)s - %(levelname)s - %(message)s")


logging.debug("debug")
logging.info("info")
logging.warning("warning")
logging.error("error")
logging.critical("critical")

def open_url(url):
    try:
        request_url = urllib.request.urlopen(url)
        content = request_url.read()
        string = content.decode("utf-8")
        print(string[:200])
    except Exception as e:
        print(f"Exception: {e}")
        logger.error(e)
        print(f"URL: {url}")

if __name__ == "__main__":
    logger = init_log("malog.txt", "w", "DEBUG", "message", "asctime")
    open_url(input("Please enter an URL: "))
```

```
WARNING:root:warning
ERROR:root:error
CRITICAL:root:critical

Please enter an URL:  www.instatam.com

Exception: unknown url type: 'www.instatam.com'
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [1], line 17, in open_url(url)
     16 try:
---> 17     request_url = urllib.request.urlopen(url)
     18     content = request_url.read()

File /opt/conda/lib/python3.10/urllib/request.py:216, in urlopen(url, data,␣
 ↪timeout, cafile, capath, cadefault, context)
```

```
    215     opener = _opener
--> 216 return opener.open(url, data, timeout)

File /opt/conda/lib/python3.10/urllib/request.py:503, in OpenerDirector.
 ↪open(self, fullurl, data, timeout)
    502 if isinstance(fullurl, str):
--> 503     req = Request(fullurl, data)
    504 else:

File /opt/conda/lib/python3.10/urllib/request.py:322, in Request.__init__(self,
 ↪url, data, headers, origin_req_host, unverifiable, method)
    319 def __init__(self, url, data=None, headers={},
    320              origin_req_host=None, unverifiable=False,
    321              method=None):
--> 322     self.full_url = url
    323     self.headers = {}

File /opt/conda/lib/python3.10/urllib/request.py:348, in Request.full_url(self,
 ↪url)
    347 self._full_url, self.fragment = _splittag(self._full_url)
--> 348 self._parse()

File /opt/conda/lib/python3.10/urllib/request.py:377, in Request._parse(self)
    376 if self.type is None:
--> 377     raise ValueError("unknown url type: %r" % self.full_url)
    378 self.host, self.selector = _splithost(rest)

ValueError: unknown url type: 'www.instatam.com'

During handling of the above exception, another exception occurred:

AttributeError                            Traceback (most recent call last)
Cell In [1], line 28
     26 if __name__ == "__main__":
     27     logger = init_log("malog.txt", "w", "DEBUG", "message", "asctime")
---> 28     open_url(input("Please enter an URL: "))

Cell In [1], line 23, in open_url(url)
     21 except Exception as e:
     22     print(f"Exception: {e}")
---> 23     logger.error(e)
     24     print(f"URL: {url}")

AttributeError: 'NoneType' object has no attribute 'error'
```

[6]:
```
import logging
import urllib.request

logging.basicConfig(level=logging.ERROR, filename="log.log", filemode="w",
                    format="%(asctime)s - %(levelname)s - %(message)s")

logging.debug("debug")
logging.info("info")
logging.warning("warning")
logging.error("error")
logging.critical("critical")

def open_url(url):

    try:
        request_url = urllib.request.urlopen(url)
        content = request_url.read()
        string = content.decode("utf-8")

        print(string[:200])

    except Exception as e:
        logger.error(e)
        print(f"URL: {url}")

if __name__ == "__main__":
    logger = init_log("malog.txt", "w", "DEBUG", "message", "asctime")
    open_url(input("Please enter an URL: "))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [6], line 27
     24         print(f"URL: {url}")
     26 if __name__ == "__main__":
---> 27     logger = init_log("malog.txt", "w", "DEBUG", "message", "asctime")
     28     open_url(input("Please enter an URL: "))

NameError: name 'init_log' is not defined
```

1.6 Task 06 – Download File

In Task 4, you used the *urllib* library to establish a http connection. You can also use the *urllib* library to perform simple file downloads.

Write a Python function *download_file(url, path)* that: - checks whether the input URL points to a .txt file - if the input URL points to a .txt file, uses the *urllib* library to download and write the text file to the given path on your machine - logs an error “No text file found at given URL,

download aborted!" to the log file created in Task 5 if the input URL does not point to a .txt file. - properly handles exceptions

Use the *download_file()* function to download William Shakespeare's drama Macbeth as a plain text file from: Macbeth

[4]:
```
import urllib.request
from os import execlp

def download_file(url, path):
    try:
        if url [-4:] == ".txt":
            urllib.request.urlretrieve(url, path)
        else:
            raise NameError("Incorrect URL")
    except:
        return "No text file found"

download_file("https://ia802707.us.archive.org/1/items/macbeth02264gut/0ws3410.
 ↪txt", "Macbeth")
```

[ ]:
```

```