**Dictionaries, Stacks And Queues**

**Before Class**

1. Familiarise yourself with dictionary data structure. Explain the concepts: key and value.
2. Watch the video on using dictionaries in Python:
3. <https://youtube.com/playlist?list=PLi01XoE8jYohWFPpC17Z-wWhPOSuh8Er>-
4. A stack is a linear data structure in which data is added to the top of the stack and is retrieved from the top of the stack. Familiarize yourself in mode detail with this data structure. Explain the concept of LIFO.
5. A queue is a linear data structure in which new data is added to the end of the queue, and data is retrieved from the beginning of the queue for further processing. Familiarize yourself in mode detail with this data structure. Explain the concept of FIFO.
6. JSON is a lightweight text format for computer data interchange. Find the file in JSON format on the Internet. Pay attention to the description of the data contained in the file. How are they structured?
7. Familiarize yourself in mode detail with JSON:
8. <https://realpython.com/python-json/>
9. <https://docs.python.org/3/library/json.html>
10. <http://json.org/>
11. Watch the video and read the article on how to deal with JSON in Python:
12. <https://youtube.com/playlist?list=PLi01XoE8jYohWFPpC17Z-wWhPOSuh8Er>-
13. <https://realpython.com/python-json/>

**During Class**

**Dictionary**

1. Create a dictionary as in the example below. Note the structure of the dictionary (key-value) and the value types in the example below. What type of value was used in each of the six key-value pairs?
2. person = {  
    "name": "Marek",  
    "surname": "Banach",  
    "age": 25,  
    "hobby": ["swimming","excursions"],  
    "married": True,  
    "phone":{"landline":"123444321","mobile":"777888999"}  
    }
3. Then, create a program and do the following tasks:
4. Display contents of the dictionary
5. Display name
6. Display hobby
7. Change surname to Nowak
8. Change person's marriage status
9. Add gender: male
10. Add a new hobby: bicycle
11. Add work phone to existing phones: 313131444
12. Create a dictionary describing a mobile phone. Use at least 6 key-value pairs of data. Use different value types. Then, using 'for' loop, display the contents of the dictionary. To read a key and value, use the items() method.
13. Create an array with 5 dictionaries, each containing a country and its population. Then, using ‘while’ loop, display the array contents.

**JSON**

1. Find any JSON file on the Internet and download it to your computer. Open the file in any character editor and read its contents. Then write a program that displays the contents of the JSON file. Use the program code below.
2. import json  
     
   with open("filename.json") as file:  
    data = json.load(file)  
     
   for k,v in data.items():  
    print(k,":",v)
3. Create a dictionary that describes your favourite book or film with at least five key-value pairs. Then create a program that writes the dictionary data to the favourite.json file. Use the dump() method. Note the formatting of the data in the json file. Use the 'indent' parameter in the dump() method.
4. Write a program where you create a dictionary containing student data. Try to describe a student in detail, using different data types that can be used in the dictionary. Then save the data about student in the file student.json, in a readable form.

**After Class**

**Dictionary and JSON**

1. Write a program that spells any text entered from the keyboard, using ICAO spelling alphabet (<https://en.wikipedia.org/wiki/NATO_phonetic_alphabet>). Create a dictionary where you put all the letters and the corresponding words. Then try to spell your name and three other words. Sample result:
2. Enter text: uek  
   Spelled text: Uniform Echo Kilo
3. Write a program that writes to a file ICAO.txt the contents of a dictionary containing ICAO spelling alphabet. Sample file content:
4. A Alfa  
   B Bravo  
   C Charlie  
   D Delta  
   …  
   …  
   Z Zulu
5. Using the website [https://mockaroo.com](https://mockaroo.com/), generate a list of 500 students, containing the following data: name, surname, student ID, gender, age, year of study, email. Write the data to the students.json file. Then write a program that creates a limited.json file with the copy of the list of students, limited to data: first name, last name, student id.
6. The website [http://api.nbp.pl](http://api.nbp.pl/) contains data on exchange rates published by the National Bank of Poland. The service provides data in json or xml formats. Display the last ten Euro exchange rates in json format in the browser window. Save the data to the euro.json file. Then write a program that displays the data from the euro.json file in the following format:
7. Date Buying Rate Selling Rate  
   ============================================  
   2019-10-25 3.8150 3.9820  
   ... ... ...

**Stack and Queue**

1. The following functions are necessary to handle the stack: push(), pop() and empty(). Below is a simple implementation of the stack using a Python list. Note the definition of the listed functions. What action do these functions perform? Copy and paste the program code below into a module named stack.py.
2. #####  
   # Stack definition  
   ##  
     
   stack = []  
     
   # add value at the top of the stack  
   def push(value):  
    stack.append(value)  
      
   # remove the topmost element of the stack  
   # and return its value   
   def pop():  
    if not empty():  
    return stack.pop()  
    else:  
    return None  
      
   # return true if the stack is empty  
   def empty():  
    return len(stack) == 0  
     
   # display stack  
   def display():  
    for i in range(len(stack)-1,-1,-1):  
    print(stack[i])  
    print()
3. Write a program, in which, import the module stack.py. Then do the following:
4. Display stack
5. Put the number 2 on the stack
6. Put the number 14 on the stack
7. Put the number 9 on the stack
8. Display stack
9. Get element from stack
10. Display stack
11. Put the number 31 on the stack
12. Put the number 6 on the stack
13. Display stack
14. Get two elements from stack
15. Display stack
16. Write a program that converts any natural number to a binary number. Use the stack. To convert a number, divide the number by 2, each time taking the remainder of the division and putting the remainder on the stack. Repeat the division until the number you are dividing is zero. Then pop and display all values from the stack. Sample result for number 18:

|  |  |
| --- | --- |
| Division | Remainder |
| 18 / 2 = 9 | 0 |
| 9 / 2 = 4 | 1 |
| 4 / 2 = 2 | 0 |
| 2 / 2 = 1 | 0 |
| 1 / 2 = 0 | 1 |

1. Natural number: 18  
   Binary number: 10010
2. Search he Internet and familiarise yourself with RPN (Reverse Polish Notation). Then, write a program that calculates RPN expressions. RPN can be conveniently evaluated using a stack structure. A user can enter by the keyboard any number, an operator (+ - \* / ) or the equal sign (=).
3. If the entered value is a number, push the number on to the stack
4. If the entered value is an operator, pop two items from the top of the stack, do the calculation, and push the result of the operation on to the stack.
5. If the entered value is an equal sigh, pop the final result from the stack and display the result of calculation.
6. Using the program, calculate the value of RPN expressions:

|  |  |
| --- | --- |
| Expression | RPN (Reverse Polish Notation) |
| 2 + 3 = | 2 3 + = |
| 2 \* (4 + 1) | 2 4 1 + \* = |
| (2 + 3) \* ( 4 + 5) = | 2 3 + 4 5 + \* = |
| 8 / (3 + 1) \* (3 - 2 + 4) = | 8 3 1 + / 3 2 – 4 + \* = |

1. Following the example of stack.py, create a queue.py module in which define queue handling. Then write a program that imports the queue.py module. Add and remove values from the queue. Display its content.