**HOMEWORK WEEK 3**

**Kristina Mituzaite**

Attachments: 20220318 HW WK3 Kristina Mituzaite.py

# TASK 1 (Conditional flow)

## Question 1

Create a program that tells you whether or not you need an umbrella when you leave the house.

The program should:

1. Ask you if it is raining using input()
2. If the input is 'y', it should output 'Take an umbrella'
3. If the input is 'n', it should output 'You don't need an umbrella'

|  |
| --- |
| Solution  Added while loop to avoid invalid answers |
| rain = input('Is it raining today? ').lower().strip()  while True:      if rain == "y":          print('Take an umbrella')          break      elif rain == "n":          print("You dont need an umbrella")          break      else:          print('Please answer only with "y" for yes or "n" for no')          rain = input('Is it raining today? ').lower().strip() |

## Question 2

I'm on holiday and want to hire a boat. The boat hire costs £20 + a refundable £5 deposit. I've written a program to check that I can afford the cost, but something doesn't seem right. Have a look at my program and work out what I've done wrong

my\_money = input('How much money do you have? ') boat cost = 20 + 5

if my\_money < boat\_cost:

print('You can afford the boat hire')

else :

print('You cannot afford the board hire'

|  |
| --- |
| Solution |
| my\_money = input('How much money do you have? ')  boat\_cost = 20 + 5  if int(my\_money) > boat\_cost:      print('You can afford the boat hire')  else :      print('You cannot afford the board hire') |

## Question 3

Your friend works for an antique book shop that sells books between **1800** and **1950** and wants to quickly categorise books by the century and decade that they were written. Write a program that takes a year (e.g. 1872) and outputs the century and decade (e.g. "Eighteenth Century, Seventies")

|  |
| --- |
| Solution |
| book\_year = input("Enter the publishing year: ")  century = book\_year[:2]  decade = int(book\_year[2:])  res\_cent = ''  res\_dec =''  century\_mapping = {      1800: "nineteenth century",      1900: "twentieth century"}  decade\_mapping = {      00: "first",      10: "tens",      20: "twenties",      30: "thirdies",      40: "fourties",      50: "fifties",      70: "seventies",      80: "eighties",      90: "nineties"}  if int(book\_year) >= 1800 and int(book\_year) <= 1950:  #    print('book year is between 1800 & 1950') #testing      if century == "18":          res\_cent = century\_mapping[1800]      else: # century == "19": #because there are only two options          res\_cent = century\_mapping[1900]        for key in decade\_mapping:          if decade >= key:              res\_dec= decade\_mapping[key]      print (res\_cent, ', ', res\_dec)  else:      print('book year must be between 1800 & 1950') |

# TASK 2 (Lists and Dictionaries)

## Question 1

I have a list of things I need to buy from my supermarket of choice.

shopping\_list = ["oranges", "cat food", "sponge cake", "long-grain rice", "cheese board",]

print(shopping\_list[1]

I want to know what the ﬁrst thing I need to buy is. However, when I run the program it shows me a different answer to what I was expecting? What is the mistake? How do I ﬁx it.

|  |
| --- |
| Solution |
| shopping\_list = ["oranges", "cat food", "sponge cake", "long-grain rice", "cheese board",]  print(shopping\_list[0]) #or  print(shopping\_list[:1])  # since the index starts at 0, ist he first item at 0 or everything before 1 |

## Question 2

I'm setting up my own market stall to sell chocolates. I need a basic till to check the prices of different chocolates that I sell. I've started the program and included the chocolates and their prices. Finish the program by **asking the user to input an item and then output its price**.

chocolates = {

'white': 1.50,

'milk': 1.20,

'dark': 1.80,

'vegan': 2.00,

}

|  |
| --- |
| Solution |
| available\_types = list(chocolates.keys())  sold\_choco = input("What type of chocolate are you buying? ").lower().strip()  if sold\_choco in chocolates.keys():      print(chocolates[sold\_choco])  else:      print("Please select one of our available chocolates {}".format(available\_types)) |

## Question 3

Write a program that simulates a lottery. The program should have a list of seven numbers that represent a lottery ticket. It should then generate seven random numbers. After comparing the two sets of numbers, the program should output a prize based on the number of matches:

* £20 for three matching numbers
* £40 for four matching numbers
* £100 for ﬁve matching numbers
* £10000 for six matching numbers
* £1000000 for seven matching numbers

|  |
| --- |
| Solution  \*there is a logic issue in the matching because of duplicate values |
| import random as rd  user\_numbers = []  count = 0 # max7  user\_selection = int(input("Select 7 times a digit 1-9: "))  #print(type(user\_numbers), type(count), type(user\_selection))  print(user\_selection)  while count < 7:      user\_numbers.append(int(user\_selection))      if int(user\_selection) >0 and int(user\_selection) < 10 :          count += 1          user\_selection = input("Select 7 times a digit 1-9: ")          #print("\n  You selected {} number(s), they are as follows{} and {}  \n ".format(count,user\_numbers, user\_selection))      else:          print("Sorry, you made an incorrect entry. Make sure your number is between 1 and 9 and start again")          quit()    print("\nYour 7 selected numbers are: {} \n".format(user\_numbers))  ## for testing  ## user\_numbers = [1, 5, 6, 7, 9, 8, 1]  winning\_numbers =[ rd.randint(1,9) for i in range (0,7)] #list  print ('The winning numbers are {}! \n \n'.format(winning\_numbers))  match =0  for number in user\_numbers:      if number in winning\_numbers:          match +=1  prise = 0  if match == 7:      prise = 1000000  elif match == 6:      prise = 10000  elif match == 5:      prise = 100  elif match == 4:      prise = 40  elif match == 3:      prise = 20  else:      print ("Try your luck once more, you missed it this time.")  ## The matching at the moment has a logic issue with duplicate numbers  print( "You got {} matching numbers! \nThat means, you won  {} € \n \n".format(match, prise) ) |

# TASK 3 (Read and Write files)

## Question 1

**You're having coffee/tea/beverage of your choice with a friend that is learning to program in Python. They're curious about why they would use pip. Explain what pip is and one beneﬁt of using pip.**

|  |
| --- |
| **Answer** |
| Pip manages the packages for Python. You can expand your python libraries with external modules that needs to be installed (or afterwards uninstalled). Pip is a tool to do that like a Google play store for python.  With “pip list” you can see what packages you have currently on your machine and could use in your code.  You can find all the available packaged modules on <https://pypi.org/search/?q=&o=> and search for topics, frameworks, license or similar |

## Question 2

**This program should save my data to a ﬁle, but it doesn't work when I run it. What is the problem and how do I ﬁx it?**

**poem = 'I like Python and I am not very good at poems' with open('poem.txt', 'r') as poem\_ﬁle:**

**poem\_ﬁle.write(poem)**

|  |
| --- |
| **Solution** |
| poem = 'I like Python and I am not very good at poems'  with open('poem.txt', 'w') as poem\_ﬁle:      poem\_ﬁle.write(poem) |

## Question 3

Here is a snippet of Elton John’s song “I’m Still Standing”

You could never know what it's like

Your blood like winter freezes just like ice

And there's a cold lonely light that shines from you

You'll wind up like the wreck you hide behind that mask you use

And did you think this fool could never win? Well look at me, I'm coming back again

I got a taste of love in a simple way

And if you need to know while I'm still standing, you just fade away

Don't you know I'm still standing better than I ever did Looking like a true survivor, feeling like a little kid

I'm still standing after all this time

Picking up the pieces of my life without you on my mind

I'm still standing (Yeah, yeah, yeah) I'm still standing (Yeah, yeah, yeah)

**Tasks:**

1. Write the lyrics to a new ﬁle called song.txt
2. Check that a ﬁle has been created successfully.
3. The read lines from this ﬁle and print out ONLY those lines that have a word ‘still’ in them.

|  |
| --- |
| Solution |
| with open("song.txt", 'w') as songtext:      songtext.write(lyrics)  ###check if new file exists  import os.path  creation\_test = os.path.exists("song.txt")  print(creation\_test)  ###read lines that have a word ‘still’  still\_lines = []  with open('song.txt', 'r') as songtext2:      lines =songtext2.readlines()      for line in lines:          if 'still' in line:              still\_lines.append(line)      print(still\_lines) |

# TASK 4 (API)

## Question 1

In this session you used the Pokémon API to retrieve a single Pokémon.

I want a program that can retrieve multiple Pokémon and save their names and moves to a file.

Use a list to store about 6 Pokémon IDs. Then in a for loop call the API to retrieve the data for each Pokémon. Save their names and moves into a file called 'pokemon.txt'

|  |
| --- |
| Solution  \*added emptylines and “name” & “moves” for readability |
| import requests  from pprint import pprint  pokemon\_gang= [1,5 ,9,15,30,500]  with open('pokemon.txt', 'w') as pokedoc:      for pokemon\_number in pokemon\_gang:          url= 'https://pokeapi.co/api/v2/pokemon/{}/'.format(pokemon\_number)          response = requests.get(url)          # print(response) #API call works          pokemon =response.json()          pokedoc.writelines( 'name: '+pokemon["name"] + '\nmoves: '  )          moves=pokemon["moves"]          for move in moves:              pokedoc.writelines(move['move']['name'])          pokedoc.write('\n\n') |

## Question 2 (optional)

Here is a link to a really cool API:<https://opentdb.com/> Answer the following questions:

* What is the name of this API?
* What does it do?
* Example URL to make a call to this API?
* Example output?

|  |
| --- |
| Answer |
| The name is: open trivia database  It provides verified questions for tests  Example url: <https://opentdb.com/api.php?amount=10&category=10&difficulty=easy&type=multiple>  Example output:# reduced to question & correct answer  import requests  from pprint import pprint as pp  url= "https://opentdb.com/api.php?amount=10&category=10&difficulty=easy&type=multiple"  response = requests.get(url)  print(response)  book\_test = (response.json())  print(type( book\_test['results']))  question\_pool=book\_test['results']  for question in question\_pool:      print(question['question'])      print(question['correct\_answer']+ "\n") |