(coventry university, 2021).,spyder live coding.

Task1

```
 # this code run to give output of the values given to the variables# when the
function e is called

# all errors corrected

 def  e(a):

  b = ''
  d = ''
  for i in range(1,len(a)-1):
    if (b==''):
      c = i
      c += 1
    elif (a[i]==d):
        c = 1
    else:
      b = 'b+str(c)+d'
    d = a[i]
  b = 'b+str(c)+d'
  return(b)

print(e('a'))
#runfile('C:/Users/cex/.spyder-py3/corrected code.py', wdir='C:/Users/cex/.spyder-
py3')#b+str(c)+d
```

Task 1 a

"""# generally code has good indentation.# the def function has syntax error.# the function e that was defined was not called,# argument passed to function e was not used e.g e(a)) and has no string# function b and d has no input string passed to them# a b d are variables that can be called in the def function e # but not properly described# generally the code lack comment# the entire code when corrected can only produce empty output and return b because the variables are all empty

```
 def  e(a)
      b = ''
      d = ''
```

for loop has synthax error, # it is use to count(iterate) the characters in the stated strings# it loops over each sequence in the strings or values passed to variables, the for loop as semantic error

```
 for i in range(1,len(a)-1)
```

# b is suppose to be used to define unknown c # c+=1 this will return b character counts at an increasing rate # and stop when the length of the character is exhausted.# the code c = 1, c was not known# and it suppose to increase each loop by one

```
    if (b=='')
        c = 1
        # elif has syntax error
 elif (a[i]==d)
    c = 1



 # else also has synthax error
 # the variables passed to b here is not "" to make it a string b="b+str(c)+d",
 # c = 1 is an unknown variable
 # in the else statement here we have made d to be equal the list of a
 else
 b = b+str(c)+d
 d = a[i]

 # the variable b+str(c)+d has to be put in a string using ""
 # b will always be returned for every print statement under the def function
 # b is the unreadable line of code
 b = b+str(c)+d
 return(b)
```

Task 1

```
# giving appropriate function and variable
def length_encoded(abcd):
    numb = ''
    numbers_encoded = ''
    new_character =''
    character_count = 0

    for j in range(1,len(numb)):
        if new_character =='':
            character_count = 1
        elif numb[j] == new_character:
            character_count += 1
        else:
            numbers_encoded = str(character_count) + new_character
            character_count = 1
            new_character = numb[j]

            numbers_encoded += str(character_count) + new_character
            return numbers_encoded
print(length_encoded("abcd"))
```

Task 1
# using values in my variables

```
 def length_encoded(greetings):
    numb = 'please'
    numbers_encoded = 'magic' +'word'
    new_character = 'thank' + 'you'

    character_count = 0
    for j in range(1,len(numb)):
        print(j)
        if new_character =='thank' + 'you':
            character_count = 1
            print(new_character)
        if  numbers_encoded =='magic' + 'word':
              character_count += 1
              print(numbers_encoded)
        if numb == 'please':
               character_count = 1
               print(numb)
        elif numb[j] == new_character:
             character_count += 1
             print(numb[j])
        else:
            numbers_encoded += str(character_count) + new_character
            character_count = 1
            new_character = numb[j]

            numbers_encoded += 'str(character_count) + new_character'
            return numbers_encoded
        print(length_encoded('please' + 'magic' +'word' + 'thank' + 'you'))


        # runfile('C:/Users/cex/.spyder-py3/appropriate variable and function.py',
wdir='C:/Users/cex/.spyder-py3')
         None
         1
         thankyou
         magicword
         please
         2
         thankyou
         magicword
         please
         3
         thankyou
         magicword
         please
```

```
4
thankyou
magicword
please
5
thankyou
magicword
please
None
```

```
# task 1
def  reduced(input_str):
    numbers_encoded = ''
    new_char = ''
    character_count = 0
    for i in range(len(input_str)):
        if new_char == '':
           character_count = 1
        elif input_str[i] == new_char:
             character_count += 1
        else:
            numbers_encoded += str(character_count) + new_char
            character_count = 1
        new_char = input_str[i]
    numbers_encoded += str(character_count) + new_char
    return numbers_encoded print(reduced('aaaaabbbbbbbbbbbcaaadd'))
```

```
runfile('C:/Users/cex/.spyder-py3/untitled5.py', wdir='C:/Users/cex/.spyder-
py3')5a11b1c3a2
```

Coventry university(2021)., spyder code
Task 1

```
# this for loop work by going around each sequence in the string# which adds to the empty variable the number is seen as a character too# e.g a is added to decode_string four times and a is added to number string once which make it 5 a's


def decoded_str(s):
    # Initialize variables
    decoded_str = ""
    num_str = ""

    # Iterate over the input string
    for i in range(len(s)):
        # If the current character is a digit, add it to num_str
        if s[i].isdigit():
            num_str += s[i]
        # If the current character is a letter, add it to the decoded string
        elif s[i].isalpha():
            decoded_str += s[i]
        # If the current character is not a digit or a letter, repeat the previous letter
num_str times
        else:
            decoded_str += decoded_str[-1] * int(num_str)
            num_str = ""

    # If there are any remaining characters in num_str, repeat the previous letter
num_str times
    if num_str:
        decoded_str+= decoded_str[-1] * int(num_str)

    return decoded_str
input_string = "4a 10b 0c 2a 1d"
decoded_string = decode_string(input_string)
print(decoded_string)
 #runfile('C:/Users/cex/.spyder-py3/untitled7.py', wdir='C:/Users/cex/.spyder-
py3')#aaaaabbbbbbbbbbbcaaadd
```

Task 2

```
Created on Fri Feb 24 01:46:35 2023
@author: cex"""
import random
def beetle_game():
    body = 0
    head = 0
    antenna = 0
    eye = 0
    mouth = 0
    leg = 0

    while leg < 6:
        roll = random.randint(1,6)
        print("You got a", roll)
        if roll == 1:
            if body == 0:
                body += 1
                print("You got a Beetle body!")
            else:
                print("You already have a Beetle body!")
        elif roll == 2:
            if body == 0:
                print("You need to get a Beetle body first!")
            elif head == 0:
                head += 1
                print("You got a Beetle head!")
            else:
                print("You already have a Beetle head!")
        elif roll == 3:
            if head == 0:
                print("You need to get a Beetle head first!")
            elif antenna < 2:
                antenna += 1
                print("You got an antenna!")
            else:
                print("You already have two antennae!")
        elif roll == 4:
            if head == 0:
                print("You need to get a Beetle head first!")
            elif eye < 2:
                eye += 1
                print("You got an eye!")
            else:
                print("You already have two eyes!")
        elif roll == 5:
            if head == 0:
                print("You need to get a Beetle head first!")
            elif mouth == 0:
                mouth += 1
                print("You got a mouth!")
            else:
                print("You already have a mouth!")
        else:
            if body == 0:
                print("You need to get a Beetle body first!")
            else:
                leg += 1
                print("You got a leg!")
    print("Congratulations! You have completed the Beetle!")
    print(beetle_game())
```

```
runfile('C:/Users/cex/.spyder-py3/untitled2 beetle.py', wdir='C:/Users/cex/.spyder-
py3')
Congratulations! You have completed the Beetle!
You got a 4
You need to get a Beetle head first!
You got a 1
You got a Beetle body!
You got a 1
You already have a Beetle body!
You got a 3
You need to get a Beetle head first!
You got a 4
You need to get a Beetle head first!
You got a 3
You need to get a Beetle head first!
You got a 1
You already have a Beetle body!
You got a 1
You already have a Beetle body!
You got a 5
You need to get a Beetle head first!
You got a 2
You got a Beetle head!
You got a 4
You got an eye!
You got a 4
You got an eye!
You got a 3
You got an antenna!
You got a 6
You got a leg!
You got a 2
You already have a Beetle head!
You gota 2
You already have a Beetle head!
You got a 6
You got a leg!
You got a 1
You already have a Beetle body!
You got a 3
You got an antenna!
You got a 2
You already have a Beetle head!
You got a 5
You got a mouth!
You got a 1
You already have a Beetle body!
You got a 5
You got have a mouth!
You got a 3
You already have two antennae!
You got a 6
You got a leg!
You got a 5
You already have a mouth!
You got a 3
You already have two antennae!
You got a 2
You already have a Beetle head!
You got a 2
You already have a Beetle head!
You got a 1
You already have a Beetle body!
You got a 4
You already have two eyes!
You got a 2
You already have a Beetle head!
You got a 5
You already have a mouth!
You got a 6
You got a leg!
You got a 2
You already have a Beetle head!
You got a 3
You already have two antennae!
You got a 4
You already have two eyes!
You got a 4
You already have two eyes!
You got a 3
You already have two antennae!
You got a 6
You got a leg!
You got a 5
You already have a mouth!
You got a 1
You already have a Beetle body!
You got a 1
You already have a Beetle body!
You got a 4
You already have two eyes!
You got a 2
You already have a Beetle head!
You got a 5
You already have a mouth!
You got a 3
You already have two antennae!
You got a 5
You already have a mouth!
You got a 5
You already have a mouth!
You got a 6
You got a leg!
None
```

```
Task 2
Created on Fri Feb 24 01:46:35 2023
@author: cex"""# mutant beetle with a second head gotten with another throw of a
two but not # consecutively
import random
def mutant_beetle_game():
    body = 0
    head = 0
    sec_head = 0
    antenna = 0
    eye = 0
    mouth = 0
    leg = 0

    while leg < 6:
        roll = random.randint(1,6)
        print("You got a", roll)
        if roll == 1:
            if body == 0:
                body += 1
                print("You got a Beetle body!")
            else:
                print("You already have a Beetle body!")
        elif roll == 2:
            if body == 0:
                print("You need to get a Beetle body first!")
            elif head == 0:
                head += 1
                print("You got a Beetle head!")
                if body == 0:
                    print("you already got a beetle body")
            elif sec_head == 0:
                sec_head += 1
                print("you got a second beetle head")
            else:
                print("You already have a Beetle head!")
        elif roll == 3:
            if head == 0:
                print("You need to get a Beetle head first!")
            elif antenna < 2:
                antenna += 1
                print("You got an antenna!")
            else:
                print("You already have two antennae!")
        elif roll == 4:
            if head == 0:
                print("You need to get a Beetle head first!")
            elif eye < 2:
                eye += 1
                print("You got an eye!")
            else:
                print("You already have two eyes!")
        elif roll == 5:
            if head == 0:
                print("You need to get a Beetle head first!")
            elif mouth == 0:
                mouth += 1
                print("You got a mouth!")
            else:
                print("You already have a mouth!")
        else:
            if body == 0:
                print("You need to get a Beetle body first!")
            else:
                leg += 1
                print("You got a leg!")
    print("Congratulations! You have completed the Beetle!")
    print(mutant_beetle_game())
```

```
runfile('C:/Users/cex/.spyder-py3/untitled2 beetle.py', wdir='C:/Users/cex/.spyder-py3')
Congratulations! You have completed the Beetle!
You got a 3
You need to get a Beetle head first!
You got a 1
You got a Beetle body!
You got a 4
You need to get a Beetle head first!
You got a 1
You already have a Beetle body!
You got a 3
You need to get a Beetle head first!
You got a 6
You got a leg!
You got a 6
You got a leg!
You got a 2
You got a Beetle head!
You got a 3
You got an antenna!
You got a 4
You got an eye!
You got a 1
You already have a Beetle body!
You got a 5
You got a mouth!
You got a 4
You got an eye!
You got a 4
You already have two eyes!
You got a 5
You already have a mouth!
You got a 6
You got a leg!
You got a 1
You already have a Beetle body!
You got a 1
You already have a Beetle body!
You got a 2
you got a second beetle head
You got a 3
You got an antenna!
You got a 1
You already have a Beetle body!
You got a 3
You already have two antennae!
You got a 6
You got a leg!
You got a 2
You already have a Beetle head!
You got a 1
You already have a Beetle body!
You got a 3
You already have two antennae!
You got a 1
You already have a Beetle body!
You got a 1
You already have a Beetle body!
You got a 5
You already have a mouth!
You got a 6
You got a leg!
You got a 6
You got a leg!
None
```

Task2
Summary table for beetle game and mutant beetle

Count	Original Beetle Game	Mutant Beetle Game
Total Rolls	50	31
Total Turns	8	7
Total Score	16	17
Bodies got	9	9
Heads got	3	8
Antennae got	6	9
Eyes got	4	7
Mouths got	3	9
Legs got	6	16
Second head	No	1

Total rolls = number dice roll , Total score = number of body part collected to complete the game ,
Total score = number times the right number is got

From the table above the mutant beetle game took fewer dice roll to complete the game. The mutant beetle game also required another complete head to be got. Overall, the mutant beetle game seems to be slightly easier to complete than the original beetle game. The mutant beetle added a challenge to build the game with two heads.

```
(Coventry university,2021).,python live coding
Task 2 roll count
Created on Fri Feb 24 01:46:35 2023
@author: cex"""
import random
def add_count_game():
    body = 0
    head = 0
    antenna = 0
    eye = 0
    mouth = 0
    leg = 0
    roll_count = 0


    while leg < 6:
        roll_count = 1
        roll = random.randint(1, 6)
        print("You got a", roll)
        if roll == 1:
            if body == 0:
                body += 1
                print("You got a Beetle body!")
            else:
                print("You already have a Beetle body!")
        elif roll == 2:
            if body == 0:
                print("You need to get a Beetle body first!")
            elif head == 0:
                head += 2
                print("You got a Beetle head!")
                if body == 0:
                    print("you already got a beetle body")
            else:
                print("You already have a Beetle head!")
        elif roll == 3:
            if head == 0:
                print("You need to get a Beetle head first!")
            elif antenna < 2:
                antenna += 1
                print("You got an antenna!")
            else:
                print("You already have two antennae!")
        elif roll == 4:
            if head == 0:
                print("You need to get a Beetle head first!")
            elif eye < 2:
                eye += 1
                print("You got an eye!")
            else:
                print("You already have two eyes!")
        elif roll == 5:
            if head == 0:
                print("You need to get a Beetle head first!")
            elif mouth == 0:
                mouth += 2
                print("You got a mouth!")
            else:
                print("You already have a mouth!")
        else:
            if body == 0:
                print("You need to get a Beetle body first!")
            else:
                leg += 1
                print("You got a leg!")
            if roll_count == 3:
                return roll_count print("game count satisfied")
           print("Congratulations! You have completed the Beetle!")
      print(add_count_game())
```

```
runfile('C:/Users/cex/.spyder-py3/untitled2 beetle.py', wdir='C:/Users/cex/.spyder-
py3')
game count satisfied
Congratulations! You have completed the Beetle!
You got a 1
You got a Beetle body!
You got a 4
You need to get a Beetle head first!
You got a 3
You need to get a Beetle head first!
You got a 1
You already have a Beetle body!
You got a 1
You already have a Beetle body!
You got a 5
You need to get a Beetle head first!
You got a 5
You need to get a Beetle head first!
You got a 3
You need to get a Beetle head first!
You got a 4
You need to get a Beetle head first!
You got a 1
You already have a Beetle body!
You got a 2
You got a Beetle head!
You got a 4
You got an eye!
You got a 1
You already have a Beetle body!
You got a 2
You already have a Beetle head!
You got a 3
You got an antenna!
You got a 4
You got an eye!
You got a 6
You got a leg!
You got a 6
You got a leg!
You got a 6
You got a leg!
You got a 4
You already have two eyes!
You got a 1
You already have a Beetle body!
You got a 6
You got a leg!
You got a 4
You already have two eyes!
You got a 2
You already have a Beetle head!
You got a 4
You already have two eyes!
You got a 4
You already have two eyes!
You got a 6
You got a leg!
You got a 4
You already have two eyes!
You got a 2
You already have a Beetle head!
You got a 5
You got a mouth!
You got a 4
You already have two eyes!
You got a 1
You already have a Beetle body!
You got a 5
You already have a mouth!
You got a 2
You already have a Beetle head!
You got a 6
You got a leg!
```

Task 3
Question 3c
Use your experience from parts (1) and (2) to critically assess the Python plotting libraries you have used in terms of the following criteria: difficulty of coding, adaptability of the code, level of control over the plot layout or labelling, and quality of the graphical plot produced. Present your assessment in an appropriate table.

Here's a table that presents an assessment of the Python plotting libraries used in parts (1) and (2) based on the criteria mentioned:

Criteria	matplotlib	Seaborn	Plotly
Difficulty of coding	Easy	Moderate	Moderate
Code adaptability	High	Low	High
Labeling	High	Moderate	High
Quality of graph	Good	Good	Excellent

Based on the above table, we can see that:
Matplotlib: are relatively easy to use for creating basic plots and provide a high level of control over the plot layout and labeling. However, they may require more effort to customize the plots and may not produce visually appealing plots by default.
Seaborn: is a higher-level library that provides a range of statistical visualizations out of the box, which can be useful for exploratory data analysis. However, it has limited adaptability and customization options compared to Matplotlib, and may not be suitable for more complex visualizations.
Plotly: is a modern and powerful library that provides a wide range of interactive and animated visualizations, which are particularly useful for web-based applications. However, it has a steeper learning curve and may require more effort to customize the plots compared to other libraries.
Overall, the choice of the library depends on the specific requirements of the visualization task, and users should consider the trade-offs between ease of use, adaptability, and quality of the graphical plot produced when selecting a library.

Task 4

Question 1:
Briefly describe (with small examples) ways in which a dataset can be "anonymised" (or "pseudonymised"), considering both quantitative and categorical data.

Answer:
Firstly, anonymisation simply means not wanting to be known or keeping identity secret. Anonymisation or pseudonymisation in data collection are ways used to safeguard sensitive data by extracting or replacing a piece of information that identifies someone or an organisation. Data can be anonymised in several ways, some of which are:

Generalisation: simply depicts changing a particular data about someone with something general, such as replacing the actual age of a patient with generalised age ranges (e.g., 30-40.). with this the patient can be treated using a random number.

Masking: this simply means exchanging certain identifying information with what it not while keeping the rest of the data intact, such as masking the last four digits of a social security number. Example: replacing the first five digits of a national insurance number with a masking letter (e.g., XXX-XX-1234B) instead of showing the full number.

Perturbation: it is a way of using random noise to data so that each value in the data cannot be traced back to a specific individual. Example: Adding random noise to the height and weight of individuals in a dataset, so the values are still usable for analysis but cannot be directly traced back to a specific person.

Tokenisation: it is a way of replacing data that can identify an individual with a unique identifier or token. Example: Instead of using a patient's name, a unique identifier or token is assigned to each patient to keep track of their data.

Question 2
Briefly describe a recent UK example of "deductive disclosure" (otherwise known as the "jigsaw effect" or the "mosaic effect") that had some adverse impact (or had the potential for some adverse impact) upon people. Critically evaluate how the relevant principles in GDPR and the UK Government Data Ethics Framework could (or should) have prevented or mitigated this impact. You must clearly cite and reference any sources you have used. These techniques can be applied to both quantitative and categorical data, depending on the specific data and the goals of the anonymisation process.
Answer:
one recent UK example of deductive disclosure is the personal data breach of the Department for Education in 2021. The department's failure to properly edit data in a Freedom of Information request led to the disclosure of sensitive information about thousands of adopted children and their families, including their names, addresses, and potentially their birth parents' names. The breach had the potential to cause harm to the children and their families, as well as violate their privacy rights.
The GDPR and the UK Government Data Ethics Framework have several principles that could have prevented or mitigated the impact of this breach. The GDPR's principle of data reduction requires that individual data should be enough, updated, and restrained to what is useful for the purpose for which it is processed. The Department for Education could have applied this principle by redacting only the necessary information in the Freedom of Information request, instead of providing more information than required.
The GDPR also requires organisations to implement the right technical and organisational measures to ensure the protection of individual data. The Department for Education could have implemented proper redaction procedures, such as using software to automatically redact sensitive information or having a designated redaction team to review all Freedom of Information requests.
The UK Government Data Ethics Framework also emphasises the importance of ethical considerations in data use, such as ensuring transparency, fairness, and accountability. The Department for Education could have been more transparent about the breach and taken accountability for its actions. They could have informed the affected individuals about the breach and provided them with support, such as counseling or credit monitoring services.
Another recent example of deductive disclosure in the UK was the NHS Test and Trace data breach in November 2020. In this incident, the personal data of thousands of people who had tested positive for COVID-19 was published on a publicly accessible website. The data included postcode, age, ethnicity, and the date of the positive test, which could be used to identify individuals. This information could be combined with other publicly available data, such as social media posts or news articles, to deduce the identity of individuals and potentially cause harm, such as harassment or discrimination.
The GDPR and the UK Government Data Ethics Framework provide guidelines and principles to mitigate the impact of such incidents. The GDPR requires organisations to implement appropriate technical and organisational measures to ensure the security of personal data, including the use of pseudonymisation and encryption. The UK Government Data Ethics Framework also emphasises the importance of ensuring the

privacy and security of personal data and highlights the need for ethical considerations in the collection and use of data.
However, in the case of the NHS Test and Trace data breach, it appears that these guidelines were not fully implemented or followed. The Information Commissioner's Office (ICO) found that the breach occurred due to a "human error" and that there was a lack of proper controls and checks in place to prevent such incidents. The ICO also highlighted the need for organisations to conduct regular risk assessments and to ensure that data processing activities are regularly reviewed and updated to address new risks and threats.
Overall, while the GDPR and the UK Government Data Ethics Framework provide important guidelines and principles to protect personal data and prevent deductive disclosure, it is ultimately up to organisations to implement these measures effectively and to take responsibility for the privacy and security of the data they collect and process. In the case of the NHS Test and Trace data breach, more stringent measures and oversight could have prevented or mitigated the impact of the incident.

Question 4
In the age of Big Data, anonymisation, and informed consent are important tools for safeguarding an individual's right to privacy, but they may not be sufficient on their own. Anonymisation can help protect individuals' identities by removing or masking personal information from data sets, but as discussed in Narayanan and Shmatikov's (2010) article, "Myths and Fallacies of Personally Identifiable Information," it is often difficult to achieve true anonymisation because even seemingly innocuous data points can be used to re-identify individuals. Informed consent, which involves providing individuals with notice and choice about the collection and use of their data, is also important but may not always be feasible or effective in practice.
Answer:
To fully safeguard an individual's right to privacy in the age of Big Data, it is necessary to take a multi-faceted approach that goes beyond just anonymisation and informed consent. This approach should involve
Privacy by design: This means building privacy protections into the design and development of technologies and data systems from the outset, rather than trying to retrofit them later. This includes minimizing the collection and use of personal data to only what is necessary, implementing strong security measures, and conducting privacy impact assessments.
Transparency and accountability: Organizations should be transparent about their data collection and use practices, and should be held accountable for any privacy breaches or violations. This includes providing clear and accessible privacy policies, establishing mechanisms for individuals to exercise their privacy rights, and conducting regular audits and assessments to ensure compliance.
Data governance: Robust data governance frameworks can help ensure that personal data is collected, used, and shared responsibly and ethically. This includes establishing clear guidelines and standards for data management, implementing appropriate data sharing agreements, and involving stakeholders and the public in decision-making processes.
Education and awareness: Finally, educating individuals about their privacy rights and the risks and benefits of sharing their data is crucial. This includes providing clear and accessible information about data collection and use practices, as well as educating individuals about how to protect their privacy online and how to recognize and respond to privacy breaches.
The debate over privacy in the age of Big Data is complex and multifaceted. While both informed consent and anonymisation can play a role in safeguarding individuals' right to privacy, neither alone is sufficient.
Informed consent, which involves providing individuals with clear and concise information about how their data will be used and obtaining their explicit permission, is an important first step in protecting privacy. However, studies have shown that individuals often do not fully understand the implications of their consent, particularly in complex contexts such as data sharing for research purposes (Mittelstadt et al., 2016). Additionally, there is often an unequal power dynamic between individuals and data collectors, particularly in cases where data collection is a condition of using a service or product.
Anonymisation, which involves removing or altering personally identifiable information to protect individuals' identities, is another important tool in protecting privacy. However, as Narayanan and Shmatikov (2010) argue, complete

anonymisation is often impossible, particularly when data is combined with other sources or contextual information. Additionally, even supposedly anonymised data can often be re-identified using advanced techniques such as machine learning (Sweeney, 2000).

Therefore, a more comprehensive approach to privacy protection is needed, one that takes into account the limitations of both informed consent and anonymisation. This could include measures such as differential privacy, which involves adding noise to data to protect individuals' identities while still allowing for useful analysis (Dwork et al., 2006), or a "data minimisation" approach that limits the amount of data collected and stored in the first place.

Ultimately, protecting privacy in the age of Big Data will require a multifaceted and evolving approach that takes into account both individual rights and the wider societal benefits of data sharing and analysis.

Sweeney, L. (2000). Simple demographics often identify people uniquely. Health (San Francisco), 671, 676.

Anonymisation and informed consent are important measures for protecting individual privacy in the age of Big Data, but they may not be sufficient on their own. Anonymisation techniques can be vulnerable to re-identification attacks, where a person's identity can still be inferred from the anonymised data by combining it with other publicly available data sources. Informed consent, while necessary, can be difficult to obtain in some situations, such as when dealing with large datasets with diverse populations.

To safeguard an individual's right to privacy, it is important to adopt a holistic approach that includes both technical and non-technical measures. This includes implementing privacy-preserving technologies, such as different people privacy, which adds noise(distraction) to data to prevent re-identification attacks, and also promoting ethical data use practices, such as transparency and accountability. It is also important to educate individuals about their privacy rights and provide them with more control over their data.

In conclusion, while anonymisation and informed consent are important tools for safeguarding privacy in the age of Big Data, they are not sufficient on their own. A multi-faceted approach that includes privacy by design, transparency and accountability, data governance, and education and awareness is necessary to fully protect individuals' privacy rights.

References:
·Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In TCC (pp. 265-284)·Mittelstadt, B. D., Fairweather, N. B., Shaw, M., & McBride, N. (2016). The ethics of algorithms: Mapping the debate. Big Data & Society, 3(2), 2053951716679679.

Snowden, E. (2016). Arguing that you don't care about the right to privacy because you have nothing to hide is no different than saying you don't care about free speech because you have nothing to say. Business Insider. https://www.businessinsider.com/edward-snowden-privacy-argument-2016-9

Narayanan, A. and Shmatikov, V. (2010). Myths and fallacies of personally identifiable information. Communications of the ACM, 53(6), 24-26. https://dl.acm.org/doi/10.1145/1743546.1743558

Universal Declaration of Human Rights. (1948). United Nations. https://www.un.org/en/about-us/universal-declaration-of-human-rights

Information Commissioner's Office. (2021). NHS Test and Trace: Data protection impact assessment. https://ico.org.uk/media/about-the-ico/documents/2619661/nhs-test-and-trace-dpia-20201118.pdf

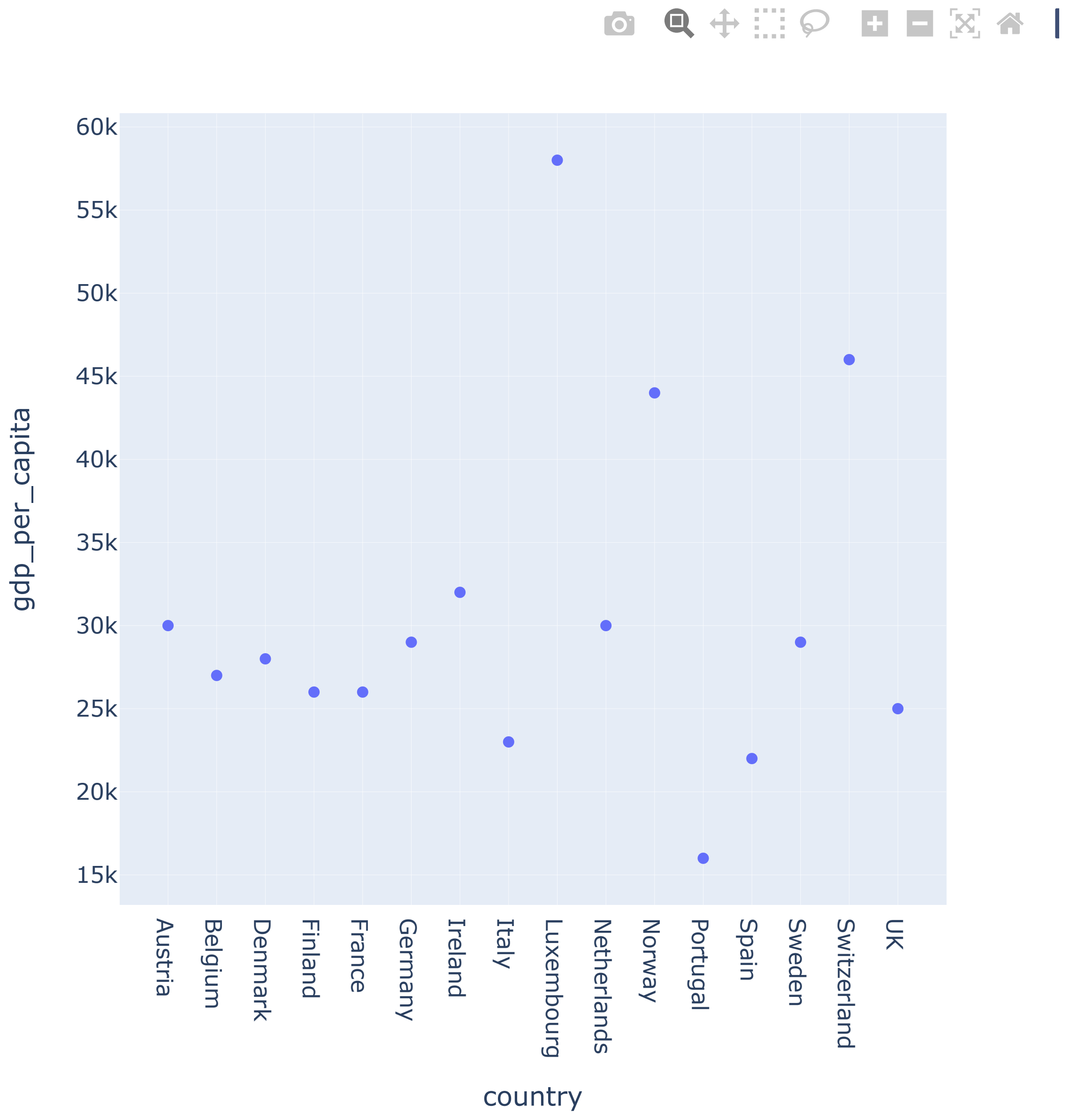
UK Government. (2021). Data Ethics Framework. https://www.gov.uk/government/publications/data-ethics-framework/data-ethics-framework

25

In [13]:

```
#https://k21academy.com/datascience-blog/python/data-visualization-using-plotly/
# Creating the Figure instance
# using the dataset
import plotly.express as px
df = px.data=({'country':  ['Austria', 'Belgium', 'Denmark', 'Finland', 'France',
            'Germany', 'Ireland', 'Italy', 'Luxembourg', 'Netherlands',
            'Norway', 'Portugal', 'Spain', 'Sweden', 'Switzerland', 'UK' ],

        'gdp_per_capita': [30000, 27000, 28000, 26000, 26000, 29000, 32000, 23000,
                  58000, 30000, 44000, 16000, 22000, 29000, 46000, 25000 ]})
print(df)
# plotting the scatter chart
fig = px.scatter(df, x='country', y="gdp_per_capita")
# showing the plot
print(fig)
fig.update_layout(
autosize=False,
width=600,
height=600,)
fig.show()
# printing the figure instance
```

```
{'country': ['Austria', 'Belgium', 'Denmark', 'Finland', 'France', 'German
y', 'Ireland', 'Italy', 'Luxembourg', 'Netherlands', 'Norway', 'Portugal',
'Spain', 'Sweden', 'Switzerland', 'UK'], 'gdp_per_capita': [30000, 27000,
28000, 26000, 26000, 29000, 32000, 23000, 58000, 30000, 44000, 16000, 2200
0, 29000, 46000, 25000]}
Figure({
    'data': [{'hovertemplate': 'country=%{x}<br>gdp_per_capita=%{y}<extra>
</extra>',
              'legendgroup': '',
              'marker': {'color': '#636efa', 'symbol': 'circle'},
              'mode': 'markers',
              'name': '',
              'orientation': 'v',
              'showlegend': False,
              'type': 'scatter',
              'x': array(['Austria', 'Belgium', 'Denmark', 'Finland', 'Fra
nce', 'Germany',
                          'Ireland', 'Italy', 'Luxembourg', 'Netherlands',
'Norway', 'Portugal',
                          'Spain', 'Sweden', 'Switzerland', 'UK'], dtype=o
bject),
              'xaxis': 'x',
              'y': array([30000, 27000, 28000, 26000, 26000, 29000, 32000,
23000, 58000, 30000,
                          44000, 16000, 22000, 29000, 46000, 25000], dtype
=int64),
              'yaxis': 'y'}],
    'layout': {'legend': {'tracegroupgap': 0},
               'margin': {'t': 60},
               'template': '...',
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'t
ext': 'country'}},
               'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'t
ext': 'gdp_per_capita'}}}
})
```


60k
55k
50k
45k
40k
35k
30k
25k
20k
15k
gdp_per_capita
Austria
Belgium
Denmark
Finland
France
Germany
Ireland
Italy
Luxembourg
Netherlands
Norway
Portugal
Spain
Sweden
Switzerland
UK
country

In [47]:

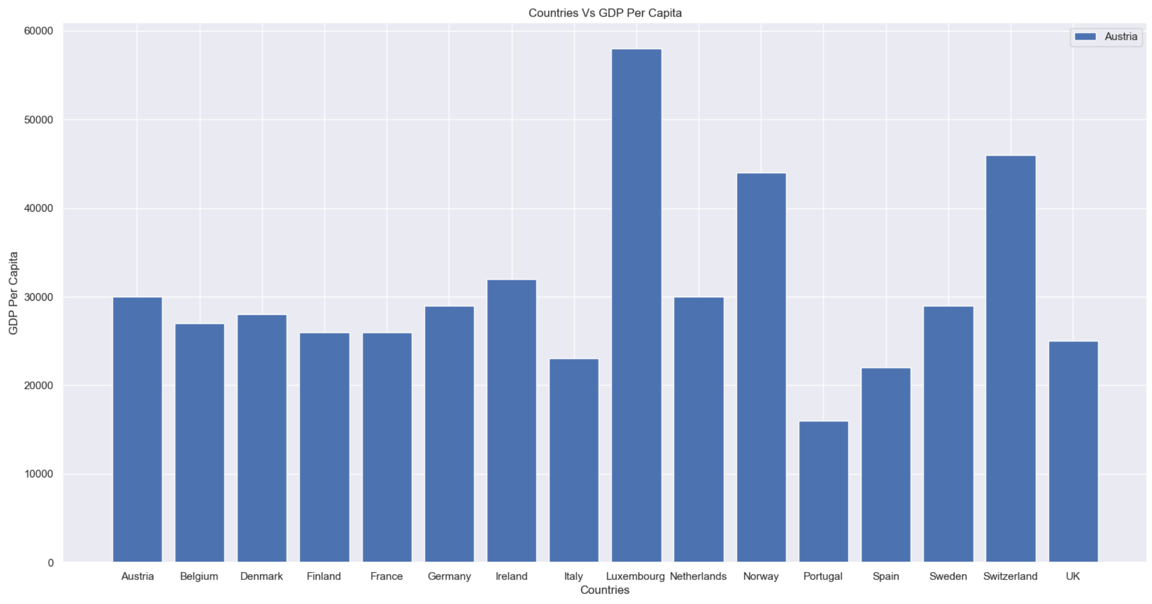
```
#https://datatofish.com/bar-chart-python-matplotlib/
#https://www.google.com/images/search?q=country+vs+gdp+per+capita+graph+in+bbc+news&form=
#https://towardsdatascience.com/data-visualization-using-seaborn-fc24db95a850
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns

data = {'country':  ['Austria', 'Belgium', 'Denmark', 'Finland', 'France',
            'Germany', 'Ireland', 'Italy', 'Luxembourg', 'Netherlands',
            'Norway', 'Portugal', 'Spain', 'Sweden', 'Switzerland', 'UK' ],

        'gdp_per_capita': [30000, 27000, 28000, 26000, 26000, 29000, 32000, 23000,
                  58000, 30000, 44000, 16000, 22000, 29000, 46000, 25000 ]}
print(data)
df = pd.DataFrame(data)
print(df)
```

```
{'country': ['Austria', 'Belgium', 'Denmark', 'Finland', 'France', 'German
y', 'Ireland', 'Italy', 'Luxembourg', 'Netherlands', 'Norway', 'Portugal',
'Spain', 'Sweden', 'Switzerland', 'UK'], 'gdp_per_capita': [30000, 27000,
28000, 26000, 26000, 29000, 32000, 23000, 58000, 30000, 44000, 16000, 2200
0, 29000, 46000, 25000]}
        country  gdp_per_capita
0       Austria           30000
1       Belgium           27000
2       Denmark           28000
3       Finland           26000
4        France           26000
5       Germany           29000
6       Ireland           32000
7         Italy           23000
8    Luxembourg           58000
9   Netherlands           30000
10       Norway           44000
11     Portugal           16000
12        Spain           22000
13       Sweden           29000
14  Switzerland           46000
15           UK           25000
```

In [48]:

```
plt.figure(figsize=(20, 10))
plt.figure(figsize=(20, 10))
plt.bar(countries, gdp_per_capita)
plt.title('Countries Vs GDP Per Capita')
plt.xlabel('Countries')
plt.ylabel('GDP Per Capita')
plt.legend(countries)
plt.show()
```

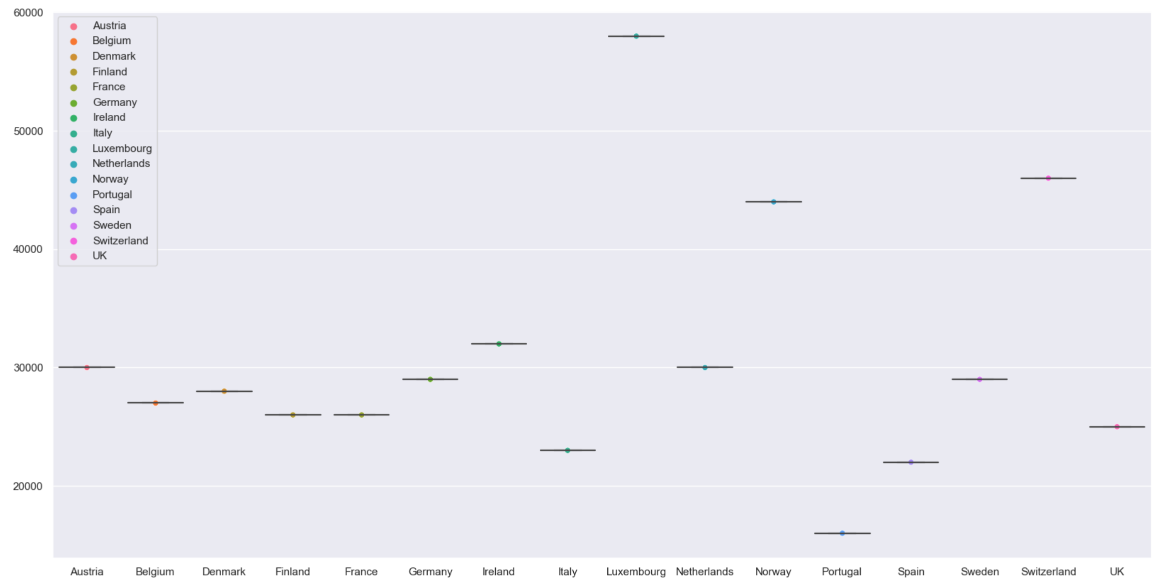
```
<Figure size 2000x1000 with 0 Axes>
```



In [46]:

```
sns.scatterplot(countries, gdp_per_capita, hue=countries)
sns.set(rc={'figure.figsize': (20, 10)})
sns.boxplot(countries, gdp_per_capita)
sns.set(style='darkgrid')
```

```
C:\Users\cex\anaconda3\New folder\lib\site-packages\seaborn\_decorators.p
y:36: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.

C:\Users\cex\anaconda3\New folder\lib\site-packages\seaborn\_decorators.p
y:36: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
```



In [ ]:

In [45]:

```
#https://k21academy.com/datascience-blog/python/data-visualization-using-plotly/
# Creating the Figure instance
fig = px.line(df,countries, gdp_per_capita)
color='g-',

print(fig)
fig.show()
# printing the figure instance
```

```
Figure({
    'data': [{'hovertemplate': 'x=%{x}<br>y=%{y}<extra></extra>',
              'legendgroup': '',
              'line': {'color': '#636efa', 'dash': 'solid'},
              'marker': {'symbol': 'circle'},
              'mode': 'lines',
              'name': '',
              'orientation': 'v',
              'showlegend': False,
              'type': 'scatter',
              'x': array(['Austria', 'Belgium', 'Denmark', 'Finland', 'Fra
nce', 'Germany',
                          'Ireland', 'Italy', 'Luxembourg', 'Netherlands',
'Norway', 'Portugal',
                          'Spain', 'Sweden', 'Switzerland', 'UK'], dtype=o
bject),
              'xaxis': 'x',
              'y': array([30000, 27000, 28000, 26000, 26000, 29000, 32000,
23000, 58000, 30000,
                          44000, 16000, 22000, 29000, 46000, 25000], dtype
=int64),
              'yaxis': 'y'}],
    'layout': {'legend': {'tracegroupgap': 0},
               'margin': {'t': 60},
               'template': '...',
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'t
ext': 'x'}},
               'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'t
ext': 'y'}}}
})
```

```
In [ ]:
```

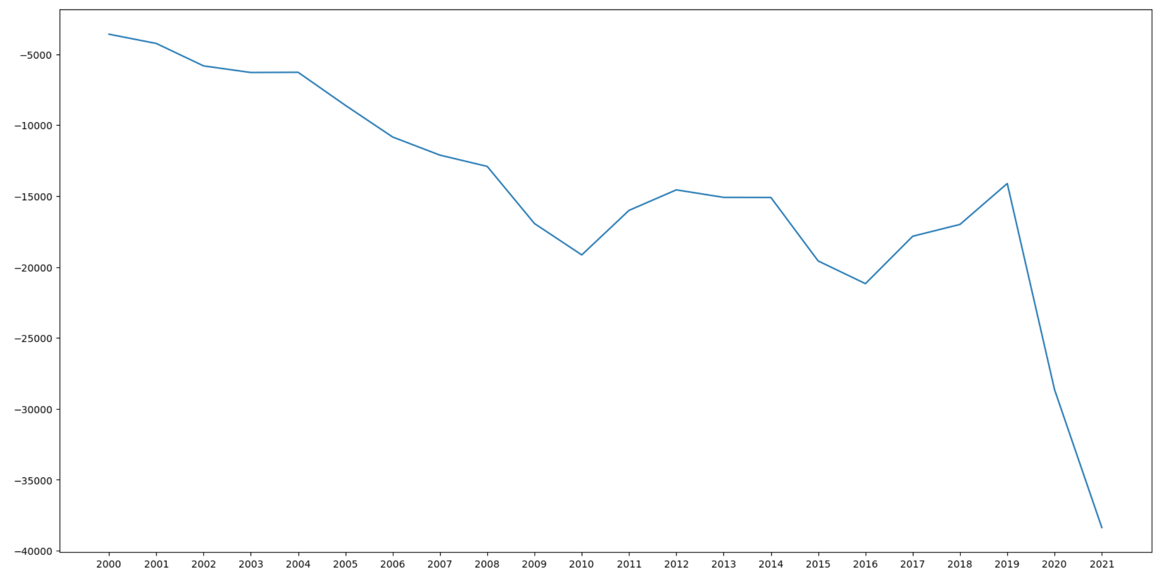
In [1]:

```
# i skipped the first 8 rows and removed the quarterly and average rows to have a clear g
# https://www.ons.gov.uk/economy/nationalaccounts/balanceofpayments/timeseries/ajfb/mret
import pandas as pd
import numpy as np
df = pd.read_csv('series-250223.csv', skiprows=8, parse_dates=True).iloc[0:22]

df.columns= ['year', 'average']
print(df)
```

```
    year  average
0   2000    -3593
1   2001    -4244
2   2002    -5826
3   2003    -6292
4   2004    -6278
5   2005    -8607
6   2006   -10840
7   2007   -12116
8   2008   -12911
9   2009   -16940
10  2010   -19150
11  2011   -16010
12  2012   -14565
13  2013   -15094
14  2014   -15100
15  2015   -19577
16  2016   -21176
17  2017   -17834
18  2018   -17002
19  2019   -14116
20  2020   -28649
21  2021   -38370
```

In [14]:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(20, 10))
xaxis= df.year
yaxis= df.average
plt.plot(xaxis, yaxis)
plt.show()
```



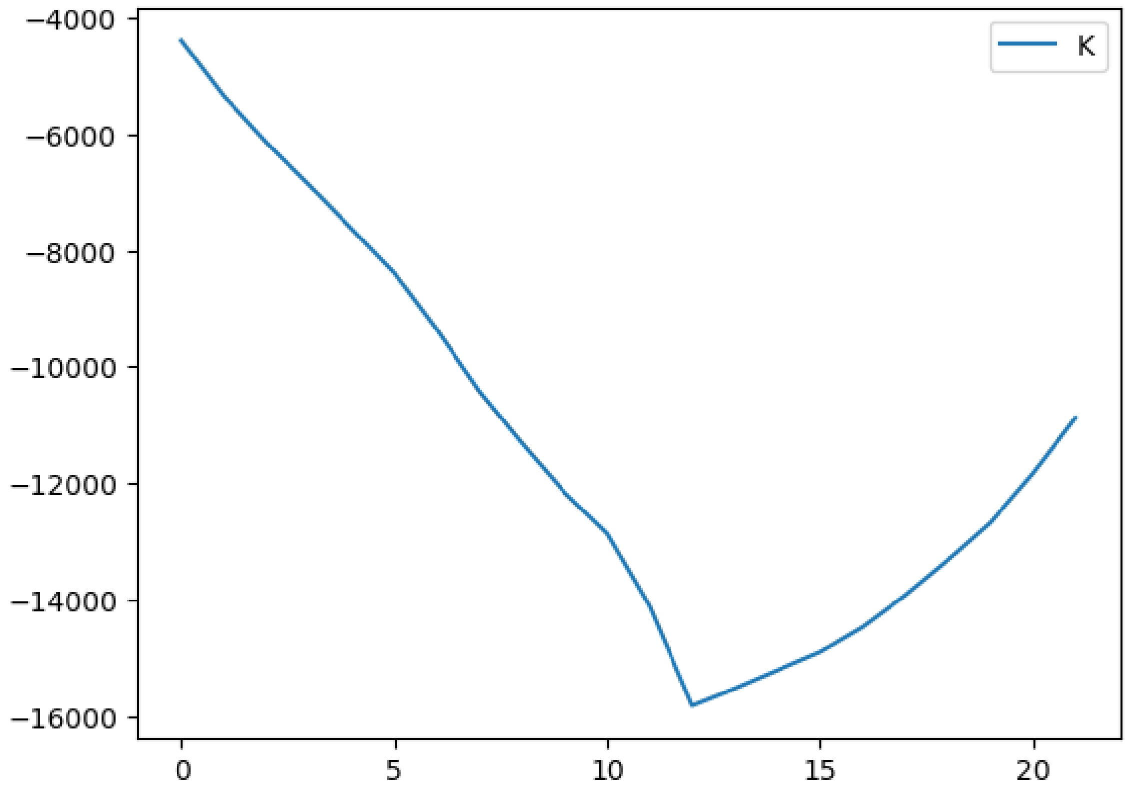
In [5]:

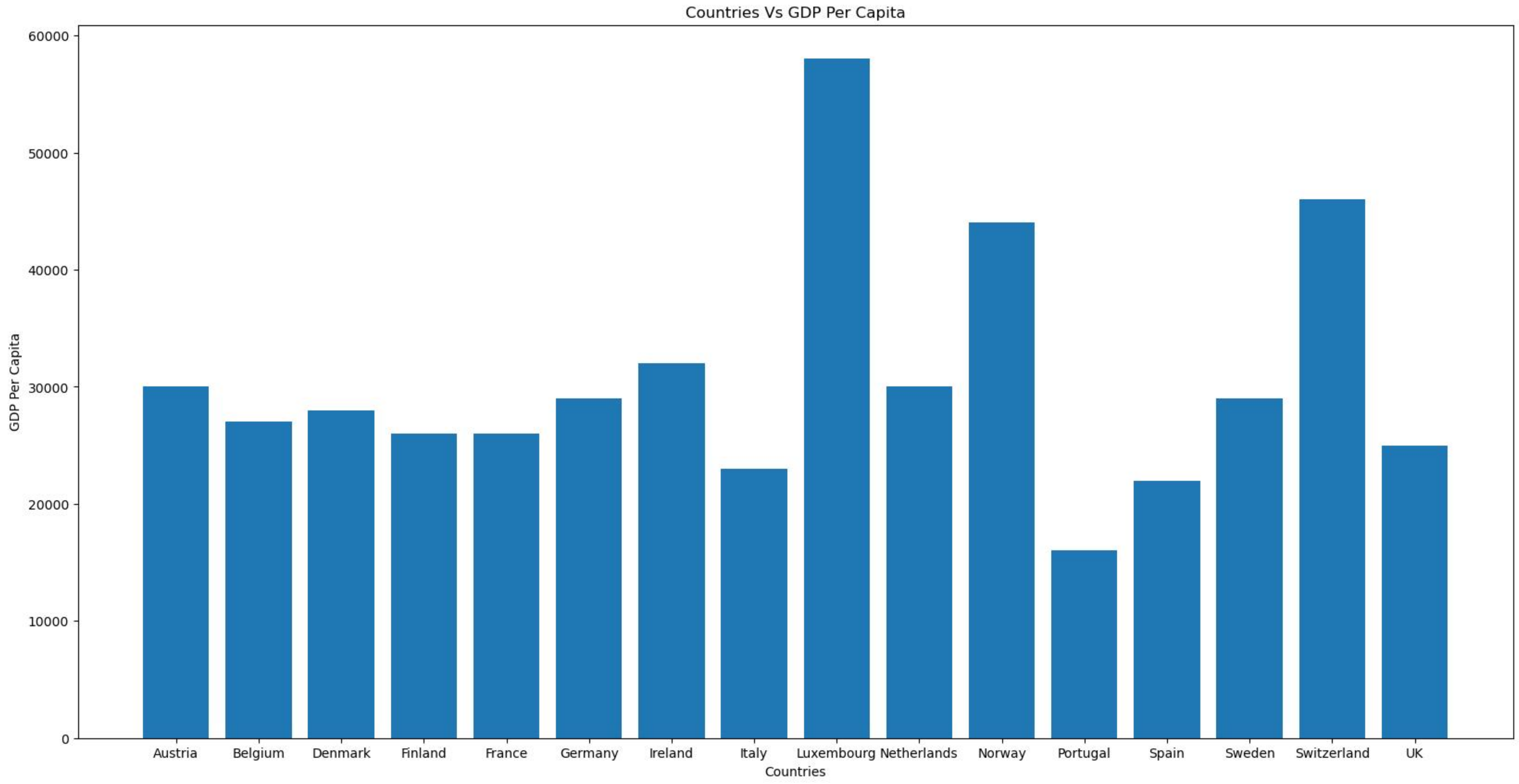
```
# python code rolling average analysis
# https://danielmuellerkomorowska.com/2020/06/02/smoothing-data-by-rolling-average-with-n
import numpy as np
import matplotlib.pyplot as plt
sss = df.average
print(sss)
kernel_size = 10
kernel = np.ones(kernel_size) / kernel_size
sss_convolved = np.convolve(sss, kernel, mode='same')

kernel_size = 20
kernel = np.ones(kernel_size) / kernel_size
sss_convolved_20 = np.convolve(sss, kernel, mode='same')

plt.plot(sss_convolved_20)
plt.legend("Kernel Size 10", )
```

```
0      -3593
1      -4244
2      -5826
3      -6292
4      -6278
5      -8607
6     -10840
7     -12116
8     -12911
9     -16940
10    -19150
11    -16010
12    -14565
13    -15094
14    -15100
15    -19577
16    -21176
17    -17834
18    -17002
19    -14116
20    -28649
21    -38370
Name: average, dtype: int64
```

Out[5]:

```
<matplotlib.legend.Legend at 0x26c8854a610>
```


K
−4000
−6000
−8000
−10000
−12000
−14000
−16000
0
5
10
15
20



Countries Vs GDP Per Capita
GDP Per Capita
Countries
0
10000
20000
30000
40000
50000
60000
Austria
Belgium
Denmark
Finland
France
Germany
Ireland
Italy
Luxembourg
Netherlands
Norway
Portugal
Spain
Sweden
Switzerland
UK