Task1

```
# this code runs 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 syntax error, # it is used 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 supposed 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 a syntax error
 elif (a[i]==d)
     c = 1


 # else also has a syntax error
 # the variables passed to b here are not in a string" 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 compress(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(compress('aaaaabbbbbbbbbbbcaaadd'))

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

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 makes it 5 a's


def decode_string(s):
    # Initialize variables
    decoded_string = ""
    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_string += s[i]
        # If the current character is not a digit or a letter, repeat the previous letter num_str times
        else:
            decoded_string += decoded_string[-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_string += decoded_string[-1] * int(num_str)

    return decoded_string
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')#aaaaabbbbbbbbbbcaaadd
```

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 rolled 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 rolled a 4
You need to get a Beetle head first!
You rolled a 1
You got a Beetle body!
You rolled a 1
You already have a Beetle body!
You rolled a 3
You need to get a Beetle head first!
You rolled a 4
You need to get a Beetle head first!
You rolled a 3
You need to get a Beetle head first!
You rolled a 1
You already have a Beetle body!
You rolled a 1
You already have a Beetle body!
You rolled a 5
You need to get a Beetle head first!
You rolled a 2
You got a Beetle head!
```

```
You rolled a 4
You got an eye!
You rolled a 4
You got an eye!
You rolled a 3
You got an antenna!
You rolled a 6
You got a leg!
You rolled a 2
You already have a Beetle head!
You rolled a 2
You already have a Beetle head!
You rolled a 6
You got a leg!
You rolled a 1
You already have a Beetle body!
You rolled a 3
You got an antenna!
You rolled a 2
You already have a Beetle head!
You rolled a 5
You got a mouth!
You rolled a 1
You already have a Beetle body!
You rolled a 5
You already have a mouth!
You rolled a 3
You already have two antennae!
You rolled a 6
You got a leg!
You rolled a 5
You already have a mouth!
You rolled a 3
You already have two antennae!
You rolled a 2
You already have a Beetle head!
You rolled a 2
You already have a Beetle head!
You rolled a 1
You already have a Beetle body!
You rolled a 4
You already have two eyes!
You rolled a 2
You already have a Beetle head!
```

```
You rolled a 5
You already have a mouth!
You rolled a 6
You got a leg!
You rolled a 2
You already have a Beetle head!
You rolled a 3
You already have two antennae!
You rolled a 4
You already have two eyes!
You rolled a 4
You already have two eyes!
You rolled a 3
You already have two antennae!
You rolled a 6
You got a leg!
You rolled a 5
You already have a mouth!
You rolled a 1
You already have a Beetle body!
You rolled a 1
You already have a Beetle body!
You rolled a 4
You already have two eyes!
You rolled a 2
You already have a Beetle head!
You rolled a 5
You already have a mouth!
You rolled a 3
You already have two antennae!
You rolled a 5
You already have a mouth!
You rolled a 5
You already have a mouth!
You rolled 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 rolled 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 rolled a 3
You need to get a Beetle head first!
You rolled a 1
You got a Beetle body!
You rolled a 4
You need to get a Beetle head first!
You rolled a 1
You already have a Beetle body!
You rolled a 3
You need to get a Beetle head first!
You rolled a 6
You got a leg!
```

```
You rolled a 6
You got a leg!
You rolled a 2
You got a Beetle head!
You rolled a 3
You got an antenna!
You rolled a 4
You got an eye!
You rolled a 1
You already have a Beetle body!
You rolled a 5
You got a mouth!
You rolled a 4
You got an eye!
You rolled a 4
You already have two eyes!
You rolled a 5
You already have a mouth!
You rolled a 6
You got a leg!
You rolled a 1
You already have a Beetle body!
You rolled a 1
You already have a Beetle body!
You rolled a 2
you got a second beetle head
You rolled a 3
You got an antenna!
You rolled a 1
You already have a Beetle body!
You rolled a 3
You already have two antennae!
You rolled a 6
You got a leg!
You rolled a 2
You already have a Beetle head!
You rolled a 1
You already have a Beetle body!
You rolled a 3
You already have two antennae!
You rolled a 1
You already have a Beetle body!
You rolled a 1
You already have a Beetle body!
```

```
You rolled a 5
You already have a mouth!
You rolled a 6
You got a leg!
You rolled 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 Rolled	9	9
Heads Rolled	3	8
Antennae Rolled	6	9
Eyes Rolled	4	7
Mouths Rolled	3	9
Legs Rolled	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 of times the right number is rolled

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 rolled. 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.

```
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 rolled 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 rolled a 1
You got a Beetle body!
You rolled a 4
You need to get a Beetle head first!
You rolled a 3
You need to get a Beetle head first!
You rolled a 1
You already have a Beetle body!
You rolled a 1
You already have a Beetle body!
You rolled a 5
You need to get a Beetle head first!
You rolled a 5
```

```
You need to get a Beetle head first!
You rolled a 3
You need to get a Beetle head first!
You rolled a 4
You need to get a Beetle head first!
You rolled a 1
You already have a Beetle body!
You rolled a 2
You got a Beetle head!
You rolled a 4
You got an eye!
You rolled a 1
You already have a Beetle body!
You rolled a 2
You already have a Beetle head!
You rolled a 3
You got an antenna!
You rolled a 4
You got an eye!
You rolled a 6
You got a leg!
You rolled a 6
You got a leg!
You rolled a 6
You got a leg!
You rolled a 4
You already have two eyes!
You rolled a 1
You already have a Beetle body!
You rolled a 6
You got a leg!
You rolled a 4
You already have two eyes!
You rolled a 2
You already have a Beetle head!
You rolled a 4
You already have two eyes!
You rolled a 4
You already have two eyes!
You rolled a 6
You got a leg!
You rolled a 4
You already have two eyes!
You rolled a 2
```

```
You already have a Beetle head!
You rolled a 5
You got a mouth!
You rolled a 4
You already have two eyes!
You rolled a 1
You already have a Beetle body!
You rolled a 5
You already have a mouth!
You rolled a 2
You already have a Beetle head!
You rolled a 6
You got a leg!
None
```

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: the 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	Numpy/Matplotlib	Pandas/Matplotlib	Seaborn Plotly	Plotnine
The difficulty of coding	Easy	Moderate	Moderate	Difficult
Code adaptability	High	Low	High	High
Labeling control level	High	Moderate	High	High
Quality of graph	Good	Good	Excellent	Good

Based on the above table, we can see that:

Numpy/Matplotlib and Pandas/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.

Plotnine is a library that is based on the popular R's ggplot2 library, and provides a declarative approach to creating visualizations. It can be a good choice for users who are familiar with ggplot2, but may be difficult to use for those who are not.

Overall, the choice of 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.

Firstly, anonymous 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 an information that identifies someone or an organisation. Data can be anonymised in several ways, some of which are:

- **Generalisation:** simply depict 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 actually 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 a way of using random noise to data so that each values 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 a way of replacing a 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 of 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 minimisation requires that personal data should be adequate, relevant, and limited to what is necessary for the purpose for which it is processed (ICO ACT,2021). 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 appropriate technical and organisational measures to ensure the security of personal 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 prevent or 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 3

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.

However, 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 for 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 differential privacy, which adds noise 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

Information Commissioner's Office. (2021). https://www.treasurers.org/system/files/ACT%20ICO%20standard%20contractual%20clauses%20controller%20to%20controller%20May%202021%20for%20external%20purposes%20%28DH%2030-06-21%29%20%282022%29.pdf?cv=1