program6

January 3, 2025

1.Write a function that accepts a positive integer as a parameter and then returns a representation of that number in binary (base 2).

[3]:
```
def to_binary(n):
    if n <= 0:
        return "Input must be a positive integer"
    return bin(n)[2:]

number = int(input("Enter the number you want to convert into binary digits: "))
print(to_binary(number))
```

```
Enter the number you want to convert into binary digits:  67
```

```
1000011
```

2. Write and test a function that takes an integer as its parameter and returns the factors of that integer.

[7]:
```
def factor_int(num):
    if num <= 0:
        return "Input must be a positive integer"
    factors = []

    for i in range(1, num + 1):
        if num % i == 0:
            factors.append(i)
    return factors

number = int(input("Enter the number for factorisation: "))
print(factor_int(number))
```

```
Enter the number for factorisation:  12
```

```
[1, 2, 3, 4, 6, 12]
```

3.Write and test a function that determines if a given integer is a prime number. A prime number is an integer greater than 1 that cannot be produced by multiplying two other integers.

[9]:
```
from math import sqrt

def prime_number(num):
```

```
    if num<=1:
        return "please enter a value greater than 1"

    for i in range(2, int(sqrt(num)) +1):
        if num%i == 0:
            return f"{num} is not a prime number."

    return f"{num} is a prime number."

prime = int(input("Enter a number to check whether it is a prime number: "))
print(prime_number(prime))
```

```
Enter a number to check whether it is a prime number:  42
```

```
42 is not a prime number.
```

4. Computers are commonly used in encryption. A very simple form of encryption (more accurately "obfuscation") would be to remove the spaces from a message and reverse the resulting string. Write, and test, a function that takes a string containing a message and "encrypts" it in this way.

[10]:
```
def encrypt_message(mssg):
    encrypted_msg = mssg.replace(" ", "")[::-1]
    return encrypted_msg

message = input("Type word or sentence for encryption: ")
print(encrypt_message(message))
```

```
Type word or sentence for encryption:  Psychology hack
```

```
kcahygolohcysP
```

5. Another way to hide a message is to include the letters that make it up within seemingly random text. The letters of the message might be every fifth character, for example. Write and test a function that does such encryption. It should randomly generate an interval (between 2 and 20), space the message out accordingly, and should fill the gaps with random letters. The function should return the encrypted message and the interval used. For example, if the message is "send cheese", the random interval is 2, and for clarity the random letters are not random: send cheese s e n d c h e e s e sxyexynxydxy cxyhxyexyexysxye

[24]:
```
import random

def encrypt_message_with_key(message):
    interval = random.randint(2, 20)
    encrypted_message = []
    key = []  # Store all original characters that are replaced

    for i in range(len(message)):
        if i % interval == 0:
            encrypted_message.append(message[i])
        else:
```

```
            encrypted_message.append('x')
            key.append(message[i])  # Save the replaced character

    encrypted_message_str = ''.join(encrypted_message)
    return encrypted_message_str, interval, key

# Example usage
message = input("Enter the message for encryption: ")
encrypted_message, interval, key = encrypt_message_with_key(message)

print(f"Original Message: {message}")
print(f"Encrypted Message: {encrypted_message}")
print(f"Interval: {interval}")
print(f"Key: {key}")
```

```
Enter the message for encryption:  Computer

Original Message: Computer
Encrypted Message: Cxxxxxxx
Interval: 10
Key: ['o', 'm', 'p', 'u', 't', 'e', 'r']
```

6. Write a program that decrypts messages encoded as above.

[25]:
```
def decrypt_message_with_key(encrypted_message, interval, key):
    decrypted_message = []
    key_index = 0

    for i in range(len(encrypted_message)):
        if i % interval == 0:
            decrypted_message.append(encrypted_message[i])  # Use the retained␣
 ↪character
        else:
            decrypted_message.append(key[key_index])  # Use characters from the␣
 ↪key
            key_index += 1

    return ''.join(decrypted_message)

encrypted_message = input("Enter the encrypted message: ")
interval = int(input("Enter the encryption interval: "))
key = input("Enter the key (comma-separated): ").split(',')

original_message = decrypt_message_with_key(encrypted_message, interval, key)

print(f"Encrypted Message: {encrypted_message}")
print(f"Decrypted (Original) Message: {original_message}")
```

```
Enter the encrypted message:  Cxxxxxxx
```

```
Enter the encryption interval:  10
Enter the key (comma-separated):  o, m, p, u, t, e, r

Encrypted Message: Cxxxxxxx
Decrypted (Original) Message: Co m p u t e r
```

[ ]: