week6_programs

January 15, 2025

[2]:
```
"""Write a function that accepts a positive integer as a parameter and then␣
 ↪returns a
representation of that number in binary (base 2)."""
def integer_to_binary(number):
    if number <= 0:
        raise ValueError("The number must be a positive integer.")
    binary_num = bin(number)[2:] #covvert 0b10 into 10
    return binary_num

def user_input():
    try:
        number = int(input("Enter a positive integer: "))
        if number <= 0:
            raise ValueError
        return number
    except ValueError:
        print("Invalid input. Please enter a positive integer.")
        return user_input()

def display(number):
    binary_representation = integer_to_binary(number)
    print(f"Binary representation of {number}: {binary_representation}")

def main():
    number = user_input()
    display(number)

main()
```

```
Enter a positive integer:  -9
Invalid input. Please enter a positive integer.
Enter a positive integer:  2
Binary representation of 2: 10
```

[19]:
```
"""Write and test a function that takes an integer as its parameter and returns␣
 ↪the
```

```
factors of that integer. (A factor is an integer which can be multiplied by␣
 ↪another to
yield the original)."""
def factors(number):
    factor_number = []
    for i in range(1, number + 1):
        if number % i == 0:
            factor_number.append(i)
    return factor_number

def user_input():
    number = int(input("Enter a number to find its factors: "))
    return number
def display():
    number=user_input()
    finalfactors=factors(number)
    print(f"The factors of {number} are: {finalfactors}")
display()
```

```
Enter a number to find its factors:  9
```

```
The factors of 9 are: [1, 3, 9]
```

[4]:
```
"""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."""
def is_prime(number):
    if number <= 1:
        return False
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            return False
    return True

def user_input():
    try:
        number = int(input("Enter a number: "))
        return number
    except ValueError:
        print("Invalid input. Please enter an integer.")
        return user_input()

def main():
    number = user_input()
    if is_prime(number):
        print(f"{number} is a prime number.")
    else:
```

```
        print(f"{number} is not a prime number.")

if __name__ == "__main__":
    main()
```

```
Enter a number:  2.5

Invalid input. Please enter an integer.

Enter a number:  2

2 is a prime number.
```

[ ]:
```
"""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."""
def encryption(message):
    message_no_spaces = message.replace(" ", "")
    encrypted_message = message_no_spaces[::-1]
    return encrypted_message

def user_input():
    message = input("Enter a message to encrypt: ")
def display():
    message=user_input()
    encrypted_message = encryption(message)
    print(f"Encrypted message: {encrypted_message}")
display()
```

[10]:
```
"""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"""
import random
import string

def encrypt_message(message):
    interval = random.randint(2, 20)
```

```
    encrypted_message = []

    for i, char in enumerate(message):
        if i > 0 and i % interval == 0:
            random_char = random.choice(string.ascii_lowercase)
            encrypted_message.append(random_char)
        encrypted_message.append(char)
    encrypted_message_str = ''.join(encrypted_message)

    return encrypted_message_str, interval


def main():
    message = input("Enter a message to encrypt: ")

    encrypted_message, interval = encrypt_message(message)

    print(f"Encrypted message: {encrypted_message}")
    print(f"Interval used: {interval}")

if __name__ == "__main__":
    main()
```

```
Enter a message to encrypt:  e

Encrypted message: e
Interval used: 18
```

[14]:
```
"""Write a program that decrypts messages encoded as above."""
def decrypt_message(encrypted_message, interval):
    decrypted_message = ""
    for i in range(0, len(encrypted_message), interval):
        decrypted_message += encrypted_message[i]

    return decrypted_message

def main():
    encrypted_message = input("Enter the encrypted message: ")
    interval = int(input("Enter the interval used during encryption: "))
    decrypted_message = decrypt_message(encrypted_message, interval)
    print(f"Decrypted message: {decrypted_message}")

if __name__ == "__main__":
    main()
```

```
Enter the encrypted message:  hello python to this world
Enter the interval used during encryption:  3

Decrypted message: hlph  iwl
```