Program 6

January 16, 2025

[1]:
```
'''1. Write a function that accepts a positive integer as a parameter and then␣
 ↪returns a
representation of that number in binary (base 2).'''

def bin_representation(n):
    if n < 0:
        print("Can't be a Negative Number")
    else:
      binary_str = ""
      while n > 0:
          binary_str = str(n % 2) + binary_str
          n //= 2
      return binary_str
bin_representation(3)
```

[1]:
```
'11'
```

[5]:
```
'''2. 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(n):
   print("The factors of",n,"are:")
   for i in range(1, n + 1):
       if n % i == 0:
           print(i)

print_factors(12)
print_factors(7)
```

```
The factors of 12 are:
1
2
3
4
6
```

```
12
The factors of 7 are:
1
7
```

[11]:
```
'''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.'''

def Prime_finder(n):
    if n > 1:
        for i in range(2, (n // 2) + 1):
            if (n % i) == 0:
                print(n, "is not a prime number")
                break
        else:
            print(n, "is a prime number")
    else:
        print(n, "is not a prime number")

Prime_finder(49)
Prime_finder(13)
```

```
49 is not a prime number
13 is a prime number
```

[22]:
```
'''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. '''

def encryption_fun(message):
    remove_space = ''.join(message.split())
    encrypted_message = remove_space[::-1]
    return encrypted_message


def message():
    code_langauge = "Let us take unsigned integers (32 bits), which consists of␣
 ↪0-31 bits."
    print(f"Original message: {code_langauge}")
    print(f"Encrypted message: {encryption_fun(code_langauge)}\n")

message()
```

```
Original message: Let us take unsigned integers (32 bits), which consists of
0-31 bits.
Encrypted message: .stib13-0fostsisnochcihw,)stib23(sregetnidengisnuekatsuteL
```

[1]:
```
'''5. Another way to hide a message is to include the letters that make it up␣
 ↪withinseemingly random text.
The letters of the message might be every fih 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

def encrypt_message(message):
    random_letters = ['x', 'y', 'z', 'a', 'b', 'c', 'd', 'e', 'f']

    message_without_spaces = message.replace(" ", "")
    interval = random.randint(2, 20)
    encrypted_message = ""

    for letter in message_without_spaces:
        for _ in range(interval - 1):
            encrypted_message += random.choice(random_letters)
        encrypted_message += letter

    return encrypted_message, interval

def main():
    message = "send cheese"
    encrypted_message, interval = encrypt_message(message)

    print(f"Original Message: {message}")
    print(f"Random Interval: {interval}")
    print(f"Encrypted Message: {encrypted_message}")

main()
```

```
Original Message: send cheese
Random Interval: 17
Encrypted Message: yfzbdzxayxaydfcbscbxcazefazdydybfeaydefyacaadcedbcnzddddedyxy
xcyyzcddbeyzfbaxyexdzbdcdcazceycdadfxazdhbebzfyabeaezcbbyezyybdebxzaybcyycecyyfx
```

```
yzyecceyxfzsebfddfbyebzzydbee
```

[1]:
```python
'''6. decript it'''

import random

def encrypt_message(message):
    random_letters = ['x', 'y', 'z', 'a', 'b', 'c', 'd', 'e', 'f']

    message_without_spaces = message.replace(" ", "")
    interval = random.randint(2, 20)
    encrypted_message = ""

    for letter in message_without_spaces:
        for _ in range(interval - 1):
            encrypted_message += random.choice(random_letters)
        encrypted_message += letter

    return encrypted_message, interval

def decrypt_message(encrypted_message, interval):
    # Extract letters at the correct interval
    decrypted_message = ""
    for i in range(interval - 1, len(encrypted_message), interval):
        decrypted_message += encrypted_message[i]
    return decrypted_message

def main():
    message = "send cheese"
    encrypted_message, interval = encrypt_message(message)

    print(f"Original Message: {message}")
    print(f"Random Interval: {interval}")
    print(f"Encrypted Message: {encrypted_message}")

    decrypted_message = decrypt_message(encrypted_message, interval)
    print(f"Decrypted Message: {decrypted_message}")

main()
```

```
Original Message: send cheese
Random Interval: 18
Encrypted Message: edfaacyaafbyzdxbxsbcacbxfyfdceabbxeeaecfbydcdzbaaeeacncbbfdax
yedexcbbbydeadzdezzdeeyddcyccxxyxfdzydfbcyydachacxeccazaeddycfyyefbexfzxdbzfyxzf
ecebbxyfaezcaecdefbesyycfyzdyecfbzedfze
Decrypted Message: sendcheese
```

[ ]: