```
def greet():
  print("Hello world!")
  print("How do you do?")
  print("How's the weather?")
greet()
```

```
Hello world!
How do you do?
How's the weather?
```

```
#Function that allows for input

def greet_with_name(name):
  print(f"Hello {name}")
  print(f"How do you do {name}?")

greet_with_name("Salad")
```

```
Hello Salad
How do you do Salad?
```

```
def greet_with(name,location):
  print(f"Hello {name}")
  print(f"What is like in {location}?")
greet_with(location="Saladland",name="Salad")
```

```
Hello Salad
What is like in Saladland?
```

```
import math
#Paint can calculator
coverage=5
def paint(height, width, coverage):
  num_of_cans=math.ceil(height*width/coverage)
  print(f"You'll need {num_of_cans} cans of paint")
paint(7,13,5)
```

```
You'll need 19 cans of paint
```

```
def prime_checker(number):
  is_prime=True
  for i in range(2,number):
    if number %i == 0:
      is_prime=False
  if is_prime ==True:
    print("It's a prime number.")
  else:
```

```
        print("It's not a prime number.")
prime_checker(47)
```

```
 It's a prime number.
```

```
alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', '

direction = input("Type 'encode' to encrypt, type 'decode' to decrypt:\n")
text = input("Type your message:\n").lower()
shift = int(input("Type the shift number:\n"))
def caesar(plain_text,shift_amount, direction):
  cipher_text=""
  for letter in plain_text:
    position=alphabet.index(letter)
    if direction=="encode":
      new_position=position+shift_amount
      if new_position>26:
        new_position=new_position-26
        new_letter=alphabet[new_position]
      else:
        new_letter=alphabet[new_position]
      cipher_text += new_letter
    elif direction=="decode":
      real=position-shift_amount
      if real<0:
        real=real+26
        real_letter=alphabet[real]
      else:
        real_letter=alphabet[real]
      cipher_text += real_letter
  print(f"The {direction}d text is {cipher_text}")

caesar(text,shift,direction)
```

```
Type 'encode' to encrypt, type 'decode' to decrypt:
encode
Type your message:
wwwe 3
Type the shift number:
2
---------------------------------------------------------------------------
--
ValueError                                Traceback (most recent call
```

```
def caesar(start_text, shift_amount, cipher_direction):
  end_text = ""
  if cipher_direction == "decode":
    shift_amount *= -1
  for char in start_text:
    #TODO-3: What happens if the user enters a number/symbol/space?
    #Can you fix the code to keep the number/symbol/space when the text is encoded/decoded?
    #e.g. start_text = "meet me at 3"
    #end_text = "•••• •• •• 3"
    if char in alphabet:
      position = alphabet.index(char)
      new_position = position + shift_amount
      end_text += alphabet[new_position]
    else:
      end_text += char
  print(f"Here's the {cipher_direction}d result: {end_text}")

#TODO-1: Import and print the logo from art.py when the program starts.
#from art import logo
#print(logo)

#TODO-4: Can you figure out a way to ask the user if they want to restart the cipher program?
#e.g. Type 'yes' if you want to go again. Otherwise type 'no'.
#If they type 'yes' then ask them for the direction/text/shift again and call the caesar() fu
#Hint: Try creating a while loop that continues to execute the program if the user types 'yes
should_end = False
while not should_end:

  direction = input("Type 'encode' to encrypt, type 'decode' to decrypt:\n")
  text = input("Type your message:\n").lower()
  shift = int(input("Type the shift number:\n"))
  #TODO-2: What if the user enters a shift that is greater than the number of letters in the
  #Try running the program and entering a shift number of 45.
  #Add some code so that the program continues to work even if the user enters a shift number
  #Hint: Think about how you can use the modulus (%).
  shift = shift % 26

  caesar(start_text=text, shift_amount=shift, cipher_direction=direction)

  restart = input("Type 'yes' if you want to go again. Otherwise type 'no'.\n")
  if restart == "no":
    should_end = True
    print("Goodbye")
```

8s completed at 11:24