DAY 8: Function Parameters & Caesar Cipher

22/01/2025
Today we're going to be looking at functions that allow you to give them inputs.
https://appbrewery.github.io/python-day8-demo/

Functions with Inputs

We learned that functions are a way of packing together a set of instructions, and when we need these instructions we can just call the function.

task.py

```
1  def greet():          #We set the def keyboard with the name  1 usage
2                        # of your function and set of parentheses
3      print("Hello")   #We create the print statement
4      print("World")
5      print(":)")
6
7  greet()           #We call the function
```

How do we start giving our functions some inputs? Now in order to actually pass this value, when I call my_function, I have to add the data inside the parentheses.

```
3  #Functions that allows for inputs
4  def greet_with_name(name):  1 usage
5      print(f" Hello {name}")  # We create the print statement
6      print(f" How are you {name}?")
7      print(":)")
8  greet_with_name("Valentina") #When we called the function, between the parentheses we need to put our name
```

```
"C:\Users\Veronica\PycharmProjects\100
 Hello Valentina
 How are you Valentina?
:)

Process finished with exit code 0
```

If I change the piece of data to call a different name, now in programming lingo, you'll hear this being referred to as the Parameter (name) , and this piece of data being referred to as the Argument.(the piece of data) Now the argument is the actual piece of data that's going to be passed over to this function when it's being called, whereas the parameter is the name of that data.

Coding Exercise : Life in Weeks

Create a function called `life_in_weeks()` using maths and f-Strings that tells us how many weeks we have left, if we live until 90 years old.

```
def life_in_weeks(age):
    years_remaining = 90 - age
    weeks_remaining = years_remaining * 52
    print(f"You have {weeks_remaining} weeks left.")


life_in_weeks(12)
```

Positional vs. Keyword Arguments

I want to create a function that allows multiple inputs. We can add a second one just by adding a coma inside the parentheses with the first parameter. Like this:

```
greet_with_name("Jack Bauer")

#function with more than one parameter
def greet_with(name, location):  1 usage
    print(f"Hello {name}")
    print(f"What is t like in {location}")
greet_with( name: "Valentina",  location: "Norway")
```

Now here's a question: what happens if I call the same function greet_with(), but I switch the order of the data that I give it. And what's actually happened here is it takes the position of the data, looks at both of these arguments, and the first argument gets assigned to the first parameter, the second argument gets assigned to the second parameter. Now we know that the parameters follow an order.

Now what if you wanted to be more clear when you actually call the function so you don't ever encounter this problem? Well, you could use something called Keyword Arguments instead.

So now instead of just adding the arguments into the function call like this, we can actually add each of the parameter names and an equal sign to say that the first parameter a = 1, b = 2, and c = 3. And now when we actually change the order around, it doesn't matter how we order it, it's still going to abide by these bindings. So c will still be 3 and a will still be 1.

```
#function with more than one parameter
def greet_with(a, b):  1 usage
    print(f"Hello {a}")
    print(f"What is t like in {b}")
greet_with(b= "Norway" a= "Valentina")
```

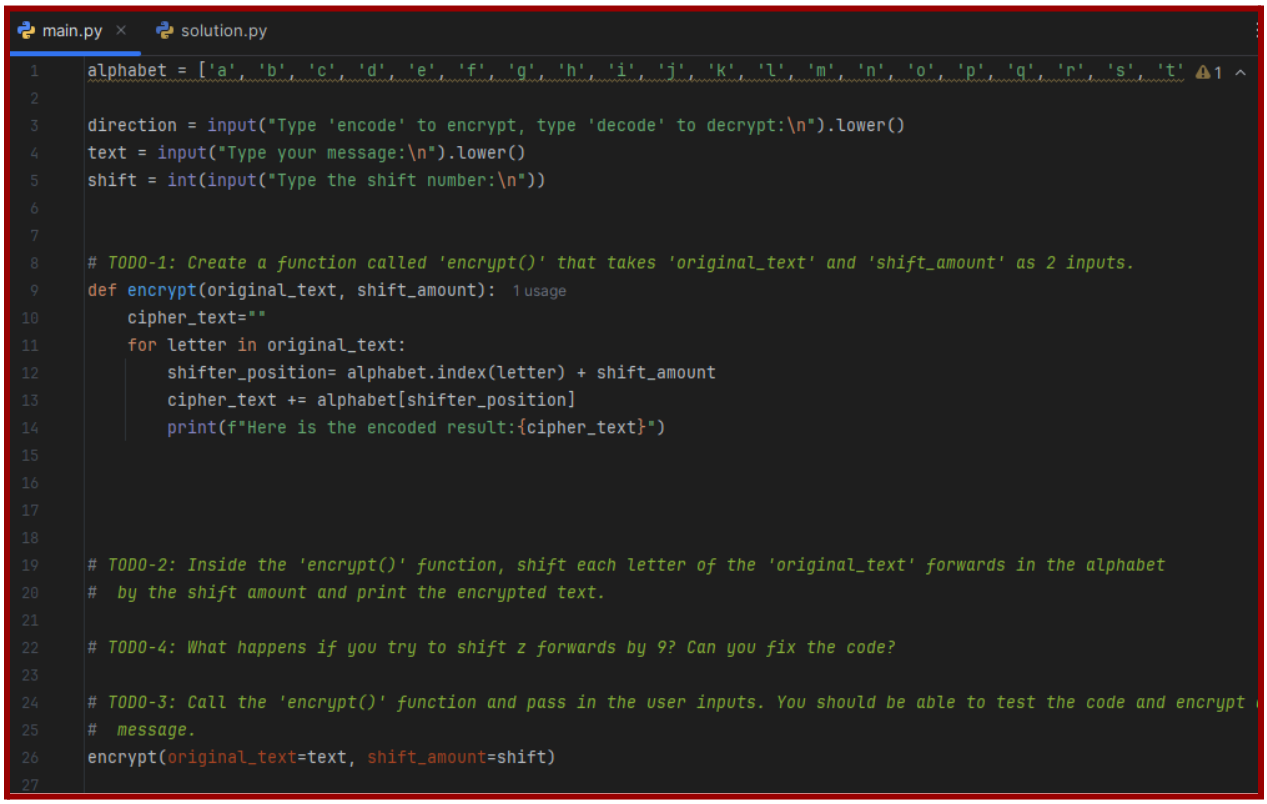
If we switch positions it no longer matters.

Coding Exercise 8: Love Calculator

You are going to write a function called `calculate_love_score()` that tests the compatibility between two names.

```
1
2   love_letters = ["l", "o", "v", "e"]
3   true_letters = ["t", "r", "u", "e"]
4   total_score = 0
5
6   def calculate_love_score(name1, name2):  1 usage
7       global total_score
8       score1 = 0
9       score2 = 0
10
11      for letter in name1:
12          if letter in love_letters or letter in true_letters:
13              score1 += 1
14
15      for letter in name2:
16          if letter in love_letters or letter in true_letters:
17              score2 += 1
18
19      total_score = score1 + score2
20
21
22  calculate_love_score( name1: "ramona",  name2: "tito")
23
24  print(total_score)
```

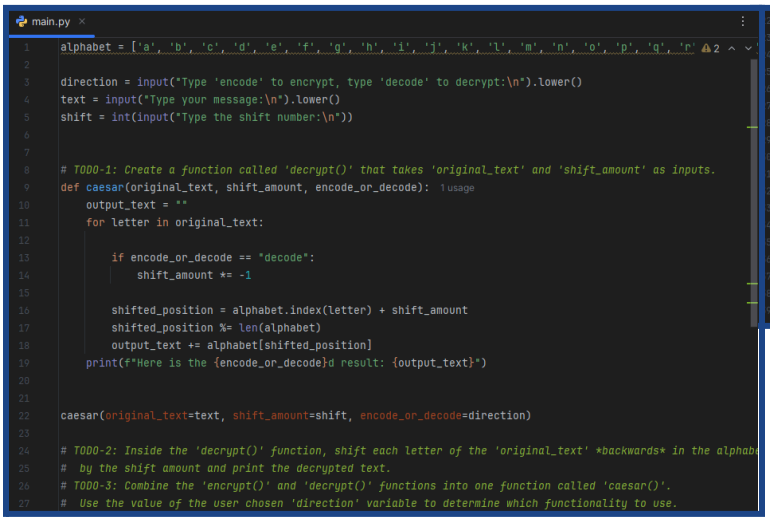
Caesar Cipher Part 1 - Encryption



main.py × solution.py

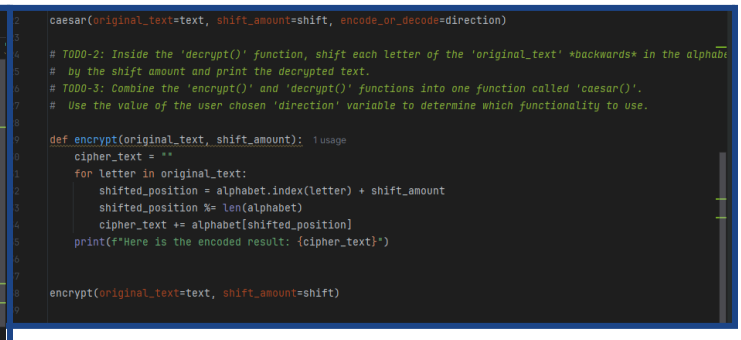
```
alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't'

direction = input("Type 'encode' to encrypt, type 'decode' to decrypt:\n").lower()
text = input("Type your message:\n").lower()
shift = int(input("Type the shift number:\n"))


# TODO-1: Create a function called 'encrypt()' that takes 'original_text' and 'shift_amount' as 2 inputs.
def encrypt(original_text, shift_amount):  1 usage
    cipher_text=""
    for letter in original_text:
        shifter_position= alphabet.index(letter) + shift_amount
        cipher_text += alphabet[shifter_position]
        print(f"Here is the encoded result:{cipher_text}")




# TODO-2: Inside the 'encrypt()' function, shift each letter of the 'original_text' forwards in the alphabet
#  by the shift amount and print the encrypted text.

# TODO-4: What happens if you try to shift z forwards by 9? Can you fix the code?

# TODO-3: Call the 'encrypt()' function and pass in the user inputs. You should be able to test the code and encrypt
#  message.
encrypt(original_text=text, shift_amount=shift)
```

Caesar Cipher Part 2 - Decryption



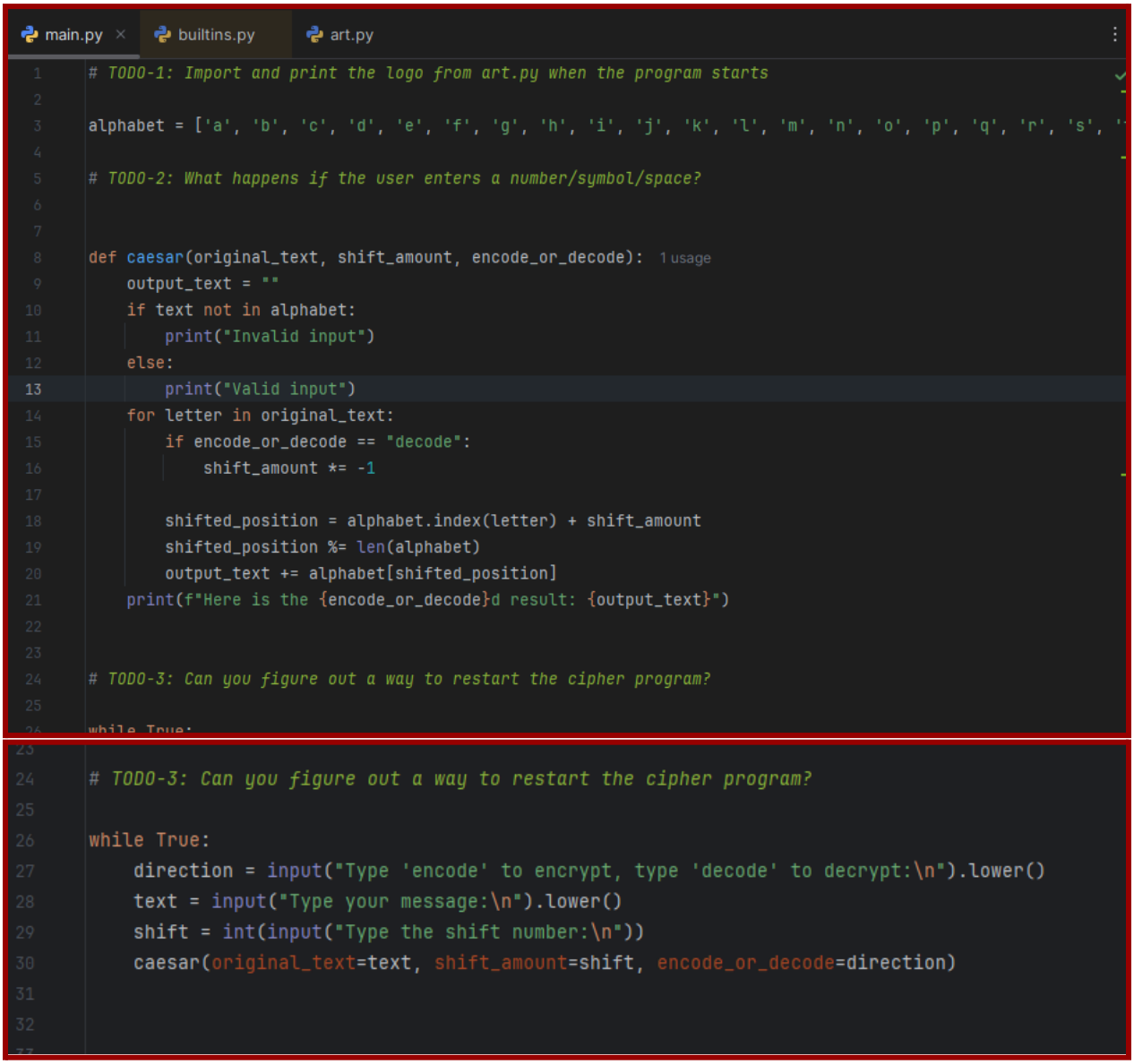
main.py ×

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

direction = input("Type 'encode' to encrypt, type 'decode' to decrypt:\n").lower()
text = input("Type your message:\n").lower()
shift = int(input("Type the shift number:\n"))


# TODO-1: Create a function called 'decrypt()' that takes 'original_text' and 'shift_amount' as inputs.
def caesar(original_text, shift_amount, encode_or_decode):  1 usage
    output_text = ""
    for letter in original_text:

        if encode_or_decode == "decode":
            shift_amount *= -1

        shifted_position = alphabet.index(letter) + shift_amount
        shifted_position %= len(alphabet)
        output_text += alphabet[shifted_position]
    print(f"Here is the {encode_or_decode}d result: {output_text}")


caesar(original_text=text, shift_amount=shift, encode_or_decode=direction)

# TODO-2: Inside the 'decrypt()' function, shift each letter of the 'original_text' *backwards* in the alphab
#  by the shift amount and print the decrypted text.
# TODO-3: Combine the 'encrypt()' and 'decrypt()' functions into one function called 'caesar()'.
#  Use the value of the user chosen 'direction' variable to determine which functionality to use.
```



```
caesar(original_text=text, shift_amount=shift, encode_or_decode=direction)

# TODO-2: Inside the 'decrypt()' function, shift each letter of the 'original_text' *backwards* in the alphab
#  by the shift amount and print the decrypted text.
# TODO-3: Combine the 'encrypt()' and 'decrypt()' functions into one function called 'caesar()'.
#  Use the value of the user chosen 'direction' variable to determine which functionality to use.

def encrypt(original_text, shift_amount):  1 usage
    cipher_text = ""
    for letter in original_text:
        shifted_position = alphabet.index(letter) + shift_amount
        shifted_position %= len(alphabet)
        cipher_text += alphabet[shifted_position]
    print(f"Here is the encoded result: {cipher_text}")


encrypt(original_text=text, shift_amount=shift)
```

Caesar Cipher Part 3 - Reorganising our Code



main.py | builtins.py | art.py

```python
# TODO-1: Import and print the logo from art.py when the program starts

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

# TODO-2: What happens if the user enters a number/symbol/space?


def caesar(original_text, shift_amount, encode_or_decode):  1 usage
    output_text = ""
    if text not in alphabet:
        print("Invalid input")
    else:
        print("Valid input")
    for letter in original_text:
        if encode_or_decode == "decode":
            shift_amount *= -1

        shifted_position = alphabet.index(letter) + shift_amount
        shifted_position %= len(alphabet)
        output_text += alphabet[shifted_position]
    print(f"Here is the {encode_or_decode}d result: {output_text}")


# TODO-3: Can you figure out a way to restart the cipher program?

while True:
```

```python
# TODO-3: Can you figure out a way to restart the cipher program?

while True:
    direction = input("Type 'encode' to encrypt, type 'decode' to decrypt:\n").lower()
    text = input("Type your message:\n").lower()
    shift = int(input("Type the shift number:\n"))
    caesar(original_text=text, shift_amount=shift, encode_or_decode=direction)
```