Task 5

Exercise 5.4: Alien Colors #2.

In [ ]:

```
alienColor = 'purple'

if alienColor == 'green':
    print("You just earned 5 points!")
else:
    print("You just earned 10 points!")
```

```
You just earned 10 points!
```

Exercise 5.5: Alien Colors #3.

In [ ]:

```
alienColor='red'

if alienColor == 'green':
    print("You just earned 5 points!")
elif alienColor == 'yellow':
    print("You just earned 10 points!")
else:
    print("You just earned 15 points!")
```

```
You just earned 15 points!
```

Exercise 5.6: Stages of Life.

In [ ]:

```
age = 12

if age < 2:
    print("You are a baby")
elif age < 4:
    print("You are a toddler")
elif age < 13:
    print("You are a kid")
elif age < 20:
    print("You are a teenager")
elif age < 65:
    print("You are an adult")
```

```
You are a kid
```

Exercise 5.7: Favorite Fruit.

In [ ]:

```
fruits=['apple','banana','orange','grape','watermelon']

if 'apple' in fruits:
    print("You really like apples!")
```

```
if 'banana' in fruits:
    print("You really like bananas!")
```

```
You really like apples!
You really like bananas!
```

Exercise 5.8: Hello Admin.

In [ ]:

```
users=['admin','user1','user2','user3','user4']

for user in users:
    if user == 'admin':
        print("Hello admin, would you like to see a status report?")
    else:
        print("Hello "+user+", thank you for logging in again.")
```

```
Hello admin, would you like to see a status report?
Hello user1, thank you for logging in again.
Hello user2, thank you for logging in again.
Hello user3, thank you for logging in again.
Hello user4, thank you for logging in again.
```

Exercise 5.10: Checking Usernames.

In [ ]:

```
currentUsers=['admin','user1','user2','user3','user4']
newUsers=['admin','user4','user5','user6','user7']

currentUsersLower=[user.lower() for user in currentUsers]

for newUser in newUsers:
    if newUser.lower() in currentUsersLower:
        print("Sorry, "+newUser+" is taken.")
    else:
        print(newUser+" is available.")
```

```
Sorry, admin is taken.
Sorry, user4 is taken.
user5 is available.
user6 is available.
user7 is available.
```

Exercise 5.11: Ordinal Numbers.

In [ ]:

```
numbers=list(range(1,10))

for number in numbers:
    if number==1:
        print(str(number)+"st")
    elif number==2:
        print(str(number)+"nd")
    elif number==3:
        print(str(number)+"rd")
    else:
        print(str(number)+"th")
```

```
1st
2nd
3rd
4th
5th
6th
7th
8th
9th
```

Exercise 7.1: Rental Car.

In [ ]:

```
car=input("What kind of car would you like to rent?")
print("Let me see if I can find you a "+car)
```

```
Let me see if I can find you a Subaru
```

Exercise 7.2: Restaurant Seating.

In [ ]:

```
partySize = input("How many people are in your dinner party tonight? ")
partySize = int(partySize)

if partySize > 8:
    print("I'm sorry, you'll have to wait for a table.")
else:
    print("Your table is ready.")
```

```
Your table is ready.
```

Exercise 7.3: Multiples of Ten.

In [ ]:

```
number=input("Please enter a number: ")
number=int(number)

if number%10==0:
    print(str(number)+" is a multiple of 10.")
else:
    print(str(number)+" is not a multiple of 10.")
```

```
20 is a multiple of 10.
```

Exercise 7.4: Pizza Toppings.

In [ ]:

```
prompt="What toppings would you like on your pizza?\nEnter 'quit' when you are fini

while True:
    topping=input(prompt)
    if topping=='quit':
        break
    else:
        print("I'll add "+topping+" to your pizza.")
```

```
I'll add Pepperoni to your pizza.
I'll add Sausage to your pizza.
I'll add Olives to your pizza.
```

Enumerate()

Write a program that takes a list of strings and converts it into a dictionary where each key is a string from the list and the value is the index of the string in the list. For example, if the input list is ["apple", "banana", "cherry"], the output dictionary should be {"apple": 0, "banana": 1, "cherry": 2}.

In [ ]:

```
testList=['a','b','c','d','e']
enum=enumerate(testList)
resultDict={}
for index, element in enum:
    resultDict[element]=index

print(resultDict)
```

```
{'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4}
```

Timing your code.

a. Write a program that takes a list of numbers and returns a new list with the squares of each number. Compare the time it takes to run this function with a lopp versus list comprehension.

In [ ]:

```
%%timeit
numList=[1,2,3,4,5,6,7,8,9,10]
squareList=[num**2 for num in numList]
```

```
456 ns ± 2.74 ns per loop (mean ± std. dev. of 7 runs, 1,000,000 loops each)
```

In [ ]:

```
%%timeit
numList=[1,2,3,4,5,6,7,8,9,10]
squareList=[]
for num in numList:
    squareList.append(num**2)
```

```
458 ns ± 4.99 ns per loop (mean ± std. dev. of 7 runs, 1,000,000 loops each)
```

b. Write a program that takes a list of numbers and returns the maximum value. Compare the time it takes to run this function using the built-in max() function versus a loop.

In [ ]:

```
%%timeit
numList=[1,2,3,4,5,6,7,8,9,10]
max(numList)
```

```
163 ns ± 1.28 ns per loop (mean ± std. dev. of 7 runs, 10,000,000 loops each)
```

In [ ]:

```
%%timeit
numList=[1,2,3,4,5,6,7,8,9,10]
maxNum=0
for num in numList:
    if num>maxNum:
        maxNum=num
```

```
267 ns ± 2.11 ns per loop (mean ± std. dev. of 7 runs, 1,000,000 loops each)
```

Sets

a. Intersection: Write a program that takes two lists and returns a new list containing only the elements that are common between the lists.

In [ ]:

```
list1=[1,2,3,4,5,6,7,8,9,10]
list2=[6, 8, 10, 12, 14, 16, 18, 20, 22, 24]

set1=set(list1)
set2=set(list2)

print(set1.intersection(set2))
```

```
{8, 10, 6}
```

b. Union: Write a program that takes two lists and returns a new list containing all the elements from both lists.

In [ ]:

```
list1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
list2 = [6, 8, 10, 12, 14, 16, 18, 20, 22, 24]

set1 = set(list1)
set2 = set(list2)

print(set1.union(set2))
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20, 22, 24}
```

c. Difference: Write a program that takes two lists and returns a new list containing only the elements that are different between the lists.

In [ ]:

```
list1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
list2 = [6, 8, 10, 12, 14, 16, 18, 20, 22, 24]

set1 = set(list1)
set2 = set(list2)

print(set1.difference(set2))
```

```
{1, 2, 3, 4, 5, 7, 9}
```

d. Symmetric Difference: Write a program that takes two lists and returns a new list containing only the elements that appear in either the first or second list, but not both.

In [ ]:

```
list1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
list2 = [6, 8, 10, 12, 14, 16, 18, 20, 22, 24]

set1 = set(list1)
set2 = set(list2)

print(set1.symmetric_difference(set2))
```

```
{1, 2, 3, 4, 5, 7, 9, 12, 14, 16, 18, 20, 22, 24}
```

Making Data Unique With Sets

a. Write a program that takes a list of numbers and returns a new list containing only the unique elements from the original list.

In [ ]:

```
recList=[1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,6,6,6]
uniqueList=[list(set(recList))]
uniqueList
```

Out[ ]:

```
[[1, 2, 3, 4, 5, 6]]
```

b. Write a program that takes a list of strings and returns a new list containing only the unique elements from the original list.

In [ ]:

```
recList=["apple", "apple", "banana", "orange", "orange", "orange", "grapes"]
uniqueList=[list(set(recList))]
uniqueList
```

Out[ ]:

```
[['grapes', 'apple', 'orange', 'banana']]
```