Collections, loops and if-conditions:

In [ ]:
```
L = [[5, 8, 7], ['hello', 'hi', 'hola'], [6.6, 1.54, 3.99], ['small', 'large']]

# Test if 'hola' is in the list L. Save to variable name test1

test1 = 'hola' in L

# Test if [5, 8, 7] is in the list L. Save to variable name test2

test2 = [5, 8, 7] in L

# Test if 6.6 is in the third element of list L. Save to variable name test3

test3 = 6.6 in L[2]



print(test1)
print(test2)
print(test3)
```

```
False
True
True
```

In [ ]:
```
nested = {'data': ['finding', 23, ['exercises', 'hangout', 34]], 'window': ['par


# Check to see if the string data is a key in nested, if it is, assign True to t

data = 'data' in nested

# Check to see if the integer 24 is in the value of the key data, if it is then

twentyfour = 24 in nested['data']

# Check to see that the string 'whole' is not in the value of the key window. If

whole = 'whole' not in nested['window']

# Check to see if the string 'physics' is a key in the dictionary nested. If it

physics = 'physics' in nested

# OUTPUT

print(data)
print(twentyfour)
print(whole)
print(physics)
```

```
True
False
False
False
```

In [ ]:
```
sports = {'swimming': ['butterfly', 'breaststroke', 'backstroke', 'freestyle'],
          'track': ['sprint', 'distance', 'jumps', 'throws'], 'gymnastics': {'wo
                                                                             'me

# Assign the string 'backstroke' to the variable name v1

v1 = sports['swimming'][2]

# Assign the string 'platform' to the variable name v2

v2 = sports['diving'][1]

# Assign the list ['vault', 'floor', 'uneven bars', 'balance beam'] to the varia

v3 = sports['gymnastics']['women']

# Assign the string 'rings' to the variable name v4

v4 = sports['gymnastics']['men'][3]



# Output
print(v1)
print(v2)
print(v3)
print(v4)
```

```
backstroke
platform
['vault', 'floor', 'uneven bars', 'balance beam']
rings
```

In [ ]:
```
# Given the dictionary `nested_d`, save the medal count for the USA from all thr


nested_d = {'Beijing':{'China':51, 'USA':36, 'Russia':22, 'Great Britain':19}, '
            'Rio':{'USA':35, 'Great Britain':22, 'China':20, 'Germany':13}}

us_count = [
    nested_d['Beijing']['USA'],
    nested_d['London']['USA'],
    nested_d['Rio']['USA']
]





# OUtput
print(us_count)
```

```
[36, 46, 35]
```

In [ ]:
```
# The same as before but figure out what is the difference and how to do it!!!


nested_d = {'Beijing':[('China', 51), ('USA', 36), ('Russia', 22), ('Great Brita
            'Rio':"""USA=35, Great_Britain=22, China=20, Germany=13"""}
```

```
us_count = []

# From Beijing
for country, medals in nested_d['Beijing']:
    if country == 'USA':
        us_count.append(medals)

# From London
us_count.append(nested_d['London']['USA'])

# From Rio
for entry in nested_d['Rio'].split(','):
    if 'USA' in entry:
        us_count.append(int(entry.split('=')[1]))

print(us_count)
```

```
[36, 46, 35]
```

In [ ]:

```
# Given below is a list of lists of athletes. Create a list `t`, that saves only
# If it doesn't contain the letter "t", save the athlete name into list `other`.


athletes = [['Phelps', 'Lochte', 'Schooling', 'Ledecky', 'Franklin'], ['Felix',

# Flatten the list of lists and classify names based on presence of 't'
t = []
other = []

for sublist in athletes:
    for name in sublist:
        if 't' in name.lower():
            t.append(name)
        else:
            other.append(name)




print(t)
print(other)
```

```
['Lochte', 'Bolt', 'Eaton', 'Dalton']
['Phelps', 'Schooling', 'Ledecky', 'Franklin', 'Felix', 'Gardner', 'Biles', 'Doug
las', 'Hamm', 'Raisman', 'Mikulak']
```

Let's level up

- List comprehension and lambda function
- functional programming (map, filter, reduce)

In [ ]:

```
# Write code to assign to the variable map_testing all the elements in lst_check
# while adding the string "Fruit" to the beginning of each element usig mapping


lst_check = ['plums', 'watermelon', 'kiwi', 'strawberries', 'blueberries', 'peac

lst_check = [
    'plums', 'watermelon', 'kiwi', 'strawberries', 'blueberries',
    'peaches', 'apples', 'mangos', 'papaya'
]

# Map each fruit to the format "Fruit: <name>"
map_testing = list(map(lambda fruit: f"Fruit: {fruit}", lst_check))

print(map_testing)
```

```
['Fruit: plums', 'Fruit: watermelon', 'Fruit: kiwi', 'Fruit: strawberries', 'Frui
t: blueberries', 'Fruit: peaches', 'Fruit: apples', 'Fruit: mangos', 'Fruit: papa
ya']
```

In [ ]:
```
# Below, a list of countries, Use filter to produce a list called b_countries th


countries = ['Canada', 'Mexico', 'Brazil', 'Chile', 'Denmark', 'Botswana', 'Spai
             'Argentina', 'Belarus', 'Laos', 'Australia', 'Panama', 'Egypt', 'Mo


# Filter countries starting with 'B'
b_countries = list(filter(lambda country: country.startswith('B'), countries))

print(b_countries)
```

```
['Brazil', 'Botswana', 'Britain', 'Bangladesh', 'Belarus', 'Belgium']
```

In [ ]:
```
# Below, a list of tuples contain the names of Game of Thrones characters. Using
# `first_names` that contains only the first names of everyone in the original l


people = [('Snow', 'Jon'), ('Lannister', 'Cersei'), ('Stark', 'Arya'), ('Stark',
          ('Tyrell', 'Margaery'), ('Stark', 'Eddard'), ('Lannister', 'Tyrion'),




# Extract first names using list comprehension
first_names = [first for _, first in people]

print(first_names)
```

```
['Jon', 'Cersei', 'Arya', 'Robb', 'Jamie', 'Daenerys', 'Sansa', 'Margaery', 'Edda
rd', 'Tyrion', 'Joffrey', 'Ramsey', 'Peter']
```

In [ ]:
```
# Below, a list of tuples that contain students' names and their final grades, U
# that contains the names of students who passed the class (had a final grade of


students = [('Tommy', 95), ('Linda', 63), ('Carl', 70), ('Bob', 100), ('Raymond'




# Keep names of students with grade >= 70
passed = [name for name, grade in students if grade >= 70]

print(passed)
```

```
['Tommy', 'Carl', 'Bob', 'Sue']
```

In [ ]:
```
# Write code using zip and filter so that these lists (l1, l2) are combined into
# to the variable `opposites` if they are both longer than 3 characters each.
```

```
l1 = ['left', 'up', 'front']
l2 = ['right', 'down', 'back']



# Zip lists and filter pairs where both strings are longer than 3 characters
opposites = list(filter(lambda pair: len(pair[0]) > 3 and len(pair[1]) > 3, zip(

print(opposites)
```

```
[('left', 'right'), ('front', 'back')]
```

In [ ]:
```
# Below two lists `species` and `population`. Use zip to combine these lists int
# From this list, create a new list called `endangered` that contains the names


species = ['golden retriever', 'white tailed deer', 'black rhino', 'brown squirr
           'blue whale', 'water moccasin', 'giant panda', 'green turtle', 'blue

population = [10000, 90000, 1000, 2000000, 500000, 500, 1200, 8000, 12000, 2300,


# Combine species and population into list of tuples
pop_info = list(zip(species, population))

# Extract endangered species (population < 2500)
endangered = [name for name, pop in pop_info if pop < 2500]

print(pop_info)
print(endangered)
```

```
[('golden retriever', 10000), ('white tailed deer', 90000), ('black rhino', 100
0), ('brown squirrel', 2000000), ('field mouse', 500000), ('orangutan', 500), ('s
umatran elephant', 1200), ('rainbow trout', 8000), ('black bear', 12000), ('blue
whale', 2300), ('water moccasin', 7500), ('giant panda', 100), ('green turtle', 1
800), ('blue jay', 9500), ('japanese beetle', 125000)]
['black rhino', 'orangutan', 'sumatran elephant', 'blue whale', 'giant panda', 'g
reen turtle']
```

Functions:

Question-1

Write a function called check_nums that takes a list as its parameter, and contains a while loop that only stops once the element of the list is the number 7. What is returned is a list of all of the numbers up until it reaches 7.

e.g;
print(check_nums([0,2,4,9,2,3,6,8,12,14,7,9,10,8,3])) ==> [0, 2, 4, 9, 2, 3, 6, 8, 12, 14]

In [ ]:
```
def check_nums(lst):
    """
    Returns a list of numbers from lst until the number 7 is encountered.
    The number 7 itself is not included in the result.
    """
    result = []
    i = 0

    while i < len(lst):
        if lst[i] == 7:
            break
        result.append(lst[i])
        i += 1

    return result


# Example usage:
print(check_nums([0, 2, 4, 9, 2, 3, 6, 8, 12, 14, 7, 9, 10, 8, 3]))
```

```
[0, 2, 4, 9, 2, 3, 6, 8, 12, 14]
```

Question-2

Write a function, test, that takes in three parameters: a required integer, an optional boolean whose default value is True, and an optional dictionary, called dict1, whose default value is {2:3, 4:5, 6:8}. If the boolean parameter is True, the function should test to see if the integer is a key in the dictionary. The value of that key should then be returned. If the boolean parameter is False, return the boolean value “False”.

e.g;
print(test(2)) ==> 3
print(test(4,False)) ==> False

In [ ]:
```
def test(num, flag=True, dict1={2: 3, 4: 5, 6: 8}):
    """
    Checks if num exists as a key in dict1 when flag is True.
    Returns the corresponding value if found, otherwise returns None.
    If flag is False, returns False.
    """
    if not flag:
        return False

    return dict1.get(num)


# Example usage:
print(test(2))
print(test(4, False))
```

```
3
False
```

Question-3:

Write a function called `checkingIfIn` that takes three parameters. The first is a required parameter, which should be a string. The second is an optional parameter called `direction` with a default value of True. The third is an optional parameter called d that has a default value of {'apple': 2, 'pear': 1, 'fruit': 19, 'orange': 5, 'banana': 3, 'grapes': 2, 'watermelon': 7}. Write the function checkingIfIn so that when the second parameter is True, it checks to see if the first parameter is a key in the third parameter; if it is, return True, otherwise return False. But if the second paramter is False, then the function should check to see if the first parameter is not a key of the third. If it's not, the function should return True in this case, and if it is, it should return False.

e.g;
print(checkingIfIn('grapes')) # True

In [ ]:
```
def checkingIfIn(item, direction=True, d={'apple': 2, 'pear': 1, 'fruit': 19, 'c
    if direction:
        # Check if item is a key in the dictionary
        return item in d
    else:
        # Check if item is NOT a key in the dictionary
        return item not in d

# Example usage:
print(checkingIfIn('grapes'))
print(checkingIfIn('mango'))
print(checkingIfIn('mango', direction=False))
print(checkingIfIn('apple', direction=False))
```

```
True
False
True
False
```