2.3.3

```
>>> n = int(input("Enter three digits (each digit for one pig):"))
Enter three digits (each digit for one pig):124
>>> a,b,c = n%10, (n//10)%10, n//100
>>> print(a+b+c, (a+b+c)//3, (a+b+c)%3, (a+b+c)%3==0)
7 2 1 False
```

3.2.1

```
>>>print("\"Shuffle, Shuffle, Shuffle\", say it together!\nChange colors and directions,\nDon't back down and stop the player!\n\t\tDo you want to play Taki?\n\t\tPress y\n")

"Shuffle, Shuffle, Shuffle", say it together!
Change colors and directions,
Don't back down and stop the player!
        Do you want to play Taki?
        Press y
```

3.3.3

```
>>> encrypted_message = "!XgXnXiXcXiXlXsX XnXoXhXtXyXpX XgXnXiXnXrXaXeXlX XmXaX Xl"
>>>encrypted_message[-1::-2]
'I am learning python slicing!'
```

3.4.2

```
 print(str[::-1].replace(str[0], 'e', str.count(str[0])-1)[::-1])
```

3.4.3

```
 print(str[:len(str)//2].lower()+str[len(str)//2:].upper())
astrONAUT
```

4.2.2

```
if(str==str[-1::-1]):
        print("OK")
```

```
else:
        print("NOT")
```

4.2.3

```
temp = input("Insert the temperature you would like to convert: ")
num = int(temp[:-1])
if(temp[-1]=='F' or temp[-1]=='f'):
    to_c = (num * 5 - 160)//9
    print(str(to_c) + 'C')
elif(temp[-1]=='C' or temp[-1]=='g'):
    to_f =  (num * 9 + (32*5))//5
    print(str(to_f) + 'F')
else:
    print("error in input")
```

4.2.4

```
import calendar

date = input("Enter a date: ")
day = int(date[:2])
month = int(date[3:5])
year = int(date[6:])

an = calendar.weekday(year, month, day)

if(an == 0):
    print("monday")
elif(an == 1):
    print("Tuesday")
elif(an == 2):
    print("Wednesday")
elif(an == 3):
    print("Thursday")
elif(an == 4):
    print("Friday")
elif(an == 5):
    print("Saturday")
else:
    print("Sunday")
```

5.3.4

```
def last_early(my_str):
    return my_str.lower().count(my_str[-1].lower()) > 1
```

5.3.5

```
def distance(num1, num2, num3):
    return (abs(num1 - num2) == 1 or abs(num1 - num3) == 1) and (abs(num1 - num2) >= 2
or abs(num1 - num3) >= 2)
```

5.3.6

```
def filter_teens(a = 13, b = 13, c = 13):
    return fix_age(a) + fix_age(b) + fix_age(c)

def fix_age(age):
    if age >= 13 and age <=19 and age != 15 and age != 16:
        return 0
    return age
```

5.3.7

```
def chocolate_maker(small, big, x):
    if x < 0 or small < 0 or big < 0 :
        return False
    if (x == 1 and small > 0) or (x == 5 and big > 0):
        return True

    return chocolate_maker(small-1,big,x-1) or chocolate_maker(small,big-1, x-5)
```

5.4

```
import math
def func(num1, num2):
    """func calculate the log val of num1 in base num2"""
    return f"log value of {num1} in base {num2} is {math.log(num1, num2)}"


def main():
    print(func(81,9))

if __name__ == "__main__":
    main()
```

6.1.2

```
def shift_left(my_list):
    my_list[0], my_list[1], my_list[2] = my_list[1], my_list[2], my_list[0]
```

```
    return my_list
def main():
    print(shift_left([0, 1, 2]))
    print(shift_left(['monkey', 2.0, 1]))

if __name__ == "__main__":
    main()
```

6.2.3

```
def format_list(my_list):
    return ', '.join(my_list[::2]) + ', and ' + my_list[-1]
```

6.2.4

```
def extend_list_x(list_x, list_y):
    return [*list_y,*list_x]
```

6.3.1

```
def are_lists_equal(list1, list2):
    list1.sort()
    list2.sort()
    return list1 == list2
```

6.3.2

```
def longest(my_list):
    return sorted(my_list, key = len)[-1]
```

7.1.4

```
def squared_numbers(start, stop):
    p=[]
    while start<=stop:
        p.append(start*start)
        start+=1
    return p
```

7.2.1

```
def is_greater(my_list, n):
    p=[]
    for x in my_list:
```

```
        if x > n:
            p.append(x)
    return p
```

7.2.2

```
def numbers_letters_count(my_str):
    p = [0, 0]
    for x in my_str:
        if x.isdigit():
            p[0]+=1
        else:
            p[1]+=1
    return p
```

7.2.4

```
def seven_boom(end_number):
    p = []
    for x in range(0,end_number+1):
        if x%7==0 or '7' in str(x):
            p.append('BOOM')
        else:
            p.append(x)
    return p
```

7.2.5

```
def sequence_del(my_str):
    new_str=""
    p = my_str[0]
    end = True
    for x in range(len(my_str)):
        if my_str[x] == p:
            end = False
            continue
        else:
            new_str += p
            p = my_str[x]
            end = True

    if not end:
        new_str += p
    return new_str
```

7.2.6

```
def print_illegal_prod(ls):
    newl=[]
    for j in ls:
        if len(j)<3 or not j.isalpha():
            newl.append(j)
    return newl
def main():
    ls = input("enter list ,:")
    ls=ls.split(',')
    n = int(input("enter a number 1-9:"))
    while n!=9:
        if n == 1:
            print("product list:", ls)
        elif n == 2:
            print(len(ls), "product in the list")
        elif n == 3:
            prod_name = input("enter prod name to find:")
            if prod_name in ls:
                print("found")
            else:
                print("not found")
        elif n == 4:
            prod_name = input("enter prod name to find coantity:")
            print(ls.count(prod_name))
        elif n == 5:
            prod_name = input("enter prod name to delete:")
            ls.remove(prod_name)
        elif n == 6:
            prod_name = input("enter prod name to add:")
            ls.append(prod_name)
        elif n == 7:
            print(print_illegal_prod(ls))
        elif n == 8:
            ls = list(dict.fromkeys(ls))
        elif n==9:
            break
        n = int(input("enter a number 1-9:"))
```

7.2.7

```
def arrow(my_char, max_length):
    for i in range(max_length+1):
        print(my_char*i)
    for j in range(max_length-1, 0, -1):
```

```
    print(my_char*j)
```

8.2.1

```
data = ("self", "py", 1.543)
  format_string = "Hello %s.%s learner, you have only %.1f units left before you master the course!"
  print(format_string % data)
```

8.2.2

```
def price(tup):
  return float(tup[1])

def sort_prices(list_of_tuples):
  return sorted(list_of_tuples, key = price, reverse = True)
```

8.2.3

```
def mult_tuple(tuple1, tuple2):
  mt = []
  for x in tuple1:
    for y in tuple2:
      mt.append((x, y))
      mt.append((y, x))
  return tuple(mt)
```

8.2.4

```
def sort_anagrams(list_of_strings):
  sorted_list = []
  anagram_groups = []

  for x in list_of_strings:
    sorted_string = ''.join(sorted(x))
    if sorted_string not in sorted_list:
      sorted_list.append(sorted_string)
      anagram_groups.append([x])
    else:
      index = sorted_list.index(sorted_string)
      anagram_groups[index].append(x)

  return anagram_groups
```

8.3.2

```
star_dict = {"first_name": "Mariah", "last_name": "Carey", "birth_date": "27.03.1970",
"hobbies": ["Sing", "Compose", "Act"]}
    n = int(input("enter a number 1-7:"))
    while n != 8:

        if n == 1:
            print("last name: ", star_dict["last_name"])
        elif n == 2:
            print("birth_date month: ", star_dict["birth_date"][3:5])
        elif n == 3:
            print(f" hobbies: {star_dict['hobbies']} ")
        elif n == 4:
            print(f"last hobbie: {star_dict['hobbies'][-1]}")
        elif n == 5:
            star_dict["hobbies"].append("Cooking")
        elif n == 6:
            print("birth_date: %s.%s.%s" %(star_dict["birth_date"][0:2],
star_dict["birth_date"][3:5], star_dict["birth_date"][6:]))
        elif n == 7:
            star_dict["age"] = date.today().year - int(star_dict["birth_date"][6:]) -
((date.today().month, date.today().day) < (int(star_dict["birth_date"][3:5]),
int(star_dict["birth_date"][0:2])))
        n = int(input("enter a number 1-7:"))
        print(star_dict)
```

8.3.3

```
def count_chars(my_str):
    str_dict = {}
    first = True
    for x in my_str:
        if x == ' ':
            continue
        if x not in str_dict.keys():
            str_dict[x] = 1
        else:
            str_dict[x] += 1
    return str_dict
```

8.3.4

```
def inverse_dict(my_dict):
    inv_dict = {}
    d_key = list(my_dict.keys())
    d_val = list(my_dict.values())
```

```
    for i in range(len(my_dict)):
        if d_val[i] in list(inv_dict.keys()):
            inv_dict[d_val[i]].append(d_key[i])
        else:
            inv_dict[d_val[i]] = [d_key[i]]
    return dict(sorted(inv_dict.items()))
```

9.1.2

```
def file_sort(path):
    ls = []
    with open(path,'r') as f1:
        for line in f1:
            for x in line.split(' '):
                if x in ls:
                    continue
                ls.append(x)
    print(sorted(ls))

def file_rev(path):
    with open(path,'r') as f1:
        for line in f1:
            print(line[-1::-1])


def file_last(path, n):
    with open(path,'r') as f1:
        lines = f1.readlines()
        for i in range(n):
            print(lines[len(lines)-i-1])
def main():
    path = input("Enter a file path: ")
    task = input("Enter a task: ")
    if task == 'sort':
        file_sort(path)
    elif task == 'rev':
        file_rev(path)
    elif task == 'last':
        n = int(input("Enter a number: "))
        file_last(path, n)
    else:
        print("invalid task")
```

9.2.2

```
def copy_file_content(source, destination):
```

```
    with open(source,'r') as s1, open(destination,'w') as d1:
        for line in s1:
            d1.write(line)
```

9.2.3

```
def who_is_missing(file_name):
    with open(file_name, 'r') as f1:
        n_ls = f1.readline().split(',')

    for i in range(0, len(n_ls)):
        n_ls[i] = int(n_ls[i])

    n_ls.sort()
    number_missing = ((len(n_ls)+1)*(len(n_ls)+2))//2 - sum(n_ls)

    with open("found.txt", 'w') as f2:
        f2.write(str(number_missing))
    return number_missing
```

9.3.1

```
def my_mp3_playlist(file_path):
    mp3_ls = []
    with open(file_path, 'r') as mp3_p:
        mp3_ls = mp3_p.read().split('\n') #split the lines

    for i in range(len(mp3_ls)):
        mp3_ls[i] = mp3_ls[i].split(';') #split each string to sublist

    long_song_time = max(mp3_ls, key = func)[2]
    artist_counter = {}
    long_song_name = ""
    for i in range(len(mp3_ls)):
        if mp3_ls[i][2] == long_song_time:
            long_song_name = mp3_ls[i][0]
        if mp3_ls[i][1] in list(artist_counter.keys()):
            artist_counter[mp3_ls[i][1]] += 1
        else:
            artist_counter[mp3_ls[i][1]] = 1
    max_artist = max(artist_counter, key=artist_counter.get)

    return long_song_name, len(mp3_ls), max_artist

def func(ls):
   return (ls[2])
```

9.3.2

```
def my_mp4_playlist(file_path, new_song):
    mp4_ls = []
    with open(file_path, 'r') as mp4_p:
        mp4_ls = mp4_p.read().split('\n') #split the lines
    for i in range(len(mp4_ls)):
        mp4_ls[i] = mp4_ls[i].split(';') #split each string to sublist
    l_4=len(mp4_ls)
    if l_4<3:
        for i in range(3-l_4):
            mp4_ls.append(" ; ; ;".split(';'))
    mp4_ls[2][0] = new_song #ותבת לקובץ את המחרוזת המייצגת שם של שיר חדש (new_song)
במקום שם השיר המופיע בשורה השלישית בקובץ
    with open(file_path, 'w') as mp4_p:
        for i in range(len(mp4_ls)):
            mp4_p.write(';'.join(mp4_ls[i]))
            mp4_p.write('\n')

    with open(file_path, 'r') as mp4_p:
        print(mp4_p.read())
```