1.1.2

```
def double_letter(my_str):
    return ''.join(list(map(f,my_str)))
def f(c):
    return c*2
```

1.1.3

```
def four_dividers(number):
    return list(filter(f, range(1, number + 1)))
def f(n):
    return n % 4 == 0
```

1.1.4

```
import functools
def sum_of_digits(number):
    return functools.reduce(f, str(number))
def f(a, b):
    return int(a) + int(b)
```

```
def combine_coins(coin, numbers): return ', '.join(list(map(lambda x: coin + str(x), numbers)))
```

1.3.1

```
def intersection(list_1, list_2):
    return list(set([x for x in list_1 for y in list_2 if x == y]))
```

1.3.2

```
def is_prime(number):
    return (lambda x: len(x) == 2)([x for x in range(1, number+1) if number % x == 0])
```

1.3.3

```
def is_funny(string):
    return set(string) == {'h', 'a'}
```

1.3.4

```
print(''.join([chr((ord(x) + 2)) for x in password]))  #code 9412
```

1.5

```
file_path = r"C:\Users\Alemo\Downloads\found.txt"
   #תוכנית שמדפיסה למסך את השם הארוך ביותר בקובץ
   with open(file_path, 'r') as f1:
      print(functools.reduce(lambda x,y: x if x > y else y, [x for x in f1]))

   #תוכנית שמדפיסה למסך את סכום האורכים של השמות בקובץ
   with open(file_path, 'r') as f1:
      print(functools.reduce(lambda x,y: x+y, [len(x.strip()) for x in f1]))

   #תוכנית שמדפיסה למסך את השמות הכי קצרים בקובץ, כל שם בשורה נפרדת
   with open(file_path, 'r') as f1:
      min_len = len(functools.reduce(lambda x,y: x if len(x) < len(y) else y, [x.strip() for x in
f1]))
   with open(file_path, 'r') as f1:
      [print(x.strip()) for x in f1 if len(x.strip()) == min_len]

   #תוכנית שיוצרת קובץ חדש בשם name_length.txt המכיל את האורך של כל שם בקובץ names.txt,
לפי הסדר, אחד בכל שורה.
   with open(file_path, 'r') as f1, open("name_length.txt", 'w') as nlf2:
      ls = [len(x.strip()) for x in f1]
      [nlf2.write(str(x) + '\n') for x in ls]

   #תוכנית שקולטת מהמשתמש מספר המייצג אורך של שם ומדפיסה את כל השמות בקובץ names.txt
שהם באורך הזה.
   n = int(input("Enter name length: "))
   with open(file_path, 'r') as f1:
      [print(x.strip()) for x in f1 if len(x.strip()) == n]
```

2.2.2

```
class Whale:
   def __init__(self):
      self.name = 'Moby Dick'
      self.age = 75
      self.type = 'sperm whale'

   def birthday(self):
      self.age += 1

   def get_age(self):
      return self.age

def main():
   a = Whale()
   b = Whale()
```

```
    a.birthday()
    print("a:",a.get_age()," b:",b.get_age())
```

2.3.3

```
class Whale:
    count_animals = 0
    def __init__(self, name, age, type = 'blue whale'):
        self._name = name
        self._age = age
        self._type = type
        Whale.count_animals += 1
    def birthday(self):
        self._age += 1
    def get_age(self):
        return self._age
    def get_name(self):
        return self._name
    def get_type(self):
        return self._type
    def set_name(self, name):
        self._name = name

def main():
    a = Whale('Moby Dick', 78, 'Sperm Whale')
    b = Whale('Whalo', 100)
    a.birthday()
    print("a:", a.get_age(), " b:", b.get_age())
    print("a:", a.get_type(), " b:", b.get_type())
    a.set_name('Magadon')
    print('a new name:', a.get_name())
    print('animals count:', Whale.count_animals)
```

2.3.4

```
class Pixel:
    count_animals = 0
    def __init__(self, x = 0, y = 0, red = 0, green = 0, blue = 0):
        self._x = x
        self._y = y
        self._red = red
        self._green = green
        self._blue = blue
    def set_coords(self, x, y):
        self._x = x
        self._y = y
    def set_grayscale(self):
        gray = (self._red + self._green + self._blue) // 3
        self._red = gray
        self._green = gray
        self._blue = gray

    def print_pixel_info(self):
        a = ''
        if self._red > 50 and self._green == 0 and self._blue == 0: a = 'Red'
        elif self._red == 0 and self._green > 50 and self._blue == 0: a = 'Green'
        elif self._red == 0 and self._green == 0 and self._blue > 50: a = 'Blue'
        print(f'X: {self._x}, Y: {self._y}, Color: {(self._red, self._green, self._blue)} {a}')
```

2.4.2

```
class BigThing:
    def __init__(self, arg):
        self._arg = arg

    def size(self):
        if isinstance(self._arg, int) or isinstance(self._arg, float):
            return self._arg
        else:
            return len(self._arg)
class BigCat(BigThing):
    def __init__(self, arg, weight):
        super().__init__(arg)
        self._weight = weight
    def size(self):
        if self._weight > 20:
            return "Very Fat"
        elif self._weight > 15:
            return "Fat"
        else:
            return "OK"
```

2.5

```
class Animal:
    zoo_name = "Hayaton"
    def __init__(self, name, hunger = 0):
        self._name = name
        self._hunger = hunger

    def get_name(self):
        return self._name

#המחזירה ערך בוליאני המתאר האם החיה רעבה או לא, חיה רעבה היא חיה שערך מידת הרעב שלה גדול מאפס
    def is_hungry(self):
        return self._hunger > 0
    def feed(self):
        self._hunger -= 1
    def talk(self):
        pass



class Dog(Animal):
    def talk(self):
        print('woof woof')
    def __str__(self):
        return f"Type: {'Dog'}, Name: {super().get_name()}"
    #special method
    def fetch_stick(self):
        print('There you go, sir!')

class Cat(Animal):
    def talk(self):
        print('meow')
    def __str__(self):
        return f"Type: {'Cat'}, Name: {super().get_name()}"
     #special method
    def chase_laser(self):
        print('Meeeeow')

class Skunk(Animal):
    def __init__(self, name, hunger = 0, count = 6):
        super().__init__(name, hunger)
        self._stink_count = count
    def talk(self):
        print('tsssss')
    def __str__(self):
        return f"Type: {'Skunk'}, Name: {super().get_name()}"
     #special method
    def stink(self):
        print('Dear lord!')

class Unicorn(Animal):
    def talk(self):
        print('Good day, darling')
    def __str__(self):
        return f"Type: {'Unicorn'}, Name: {super().get_name()}"
     #special method
    def sing(self):
        print('I'm not your toy...')

class Dragon(Animal):
    def __init__(self, name, hunger = 0, color = "Green"):
        super().__init__(name, hunger)
        self._color = color
    def talk(self):
        print('Raaaawr')
    def __str__(self):
        return f"Type: {'Dragon'}, Name: {super().get_name()}"
     #special method
    def breath_fire(self):
        print('$@#$#@$')

def main():
    dog = Dog('Brownie', 10)
    cat = Cat('Zelda', 3)
    skunk = Skunk('Stinky')
    unicorn = Unicorn('Keith', 7)
    dragon = Dragon('Lizzy', 1450)
    zoo_lst = [dog, cat, skunk, unicorn, dragon]

    dog = Dog('Doggo', 80)
    cat = Cat('Kitty', 80)
    skunk = Skunk('Stinky Jr.', 80)
    unicorn = Unicorn('Clair', 80)
    dragon = Dragon('McFly', 80)
    zoo_lst += [dog, cat, skunk, unicorn, dragon]

    for animal in zoo_lst:
        if animal.is_hungry():
            print(animal)
            while animal.is_hungry():
                animal.feed()
        animal.talk()
        if isinstance(animal, Dog):
            animal.fetch_stick()
        elif isinstance(animal, Cat):
            animal.chase_laser()
        elif isinstance(animal, Skunk):
            animal.stink()
        elif isinstance(animal, Unicorn):
            animal.sing()
        elif isinstance(animal, Dragon):
            animal.breath_fire()
    print(Animal.zoo_name)

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

3.2.5
```
def read_file(file_name):
    a = "__CONTENT_START__\n"
    try:
        f = open(file_name)
        try:
            a += f.read()+'\n'
        finally:
            f.close()
    except IOError:
        a += "__NO_SUCH_FILE__\n"
    finally:
        return a + "__CONTENT_END__"
```

3.3.2
```
class UnderAge(Exception):
    def __init__(self, age):
        self._age = age
    def __str__(self):
        return f"user is under age {self._age}, in {18 - self._age} years will be invited"

def send_invitation(name, age):
    try:
        if int(age) < 18:
            raise UnderAge(age)
    except UnderAge as e:
        print(e)
    else:
        print("You should send an invite to " + name)
```

3.4

```
import string

class UsernameContainsIllegalCharacter(Exception):
    def __init__(self, char, index):
        self._char = char
        self._index = index
    def __str__(self):
        return f'The username contains an illegal character "{self._char}" at index {self._index}'
class UsernameTooShort(Exception):
    def __str__(self):
        return "The username is too short"
class UsernameTooLong(Exception):
    def __str__(self):
        return "The username is too long"
```

```
class PasswordMissingCharacter(Exception):
    def __init__(self):
        pass
    def __str__(self):
        return "The password is missing a character"
class PasswordMissingUppercase(PasswordMissingCharacter):
    def __str__(self):
        return super().__str__() + ' (Uppercase)'

class PasswordMissingLowercase(PasswordMissingCharacter):
    def __str__(self):
        return super().__str__() + ' (Lowercase)'

class PasswordMissingDigit(PasswordMissingCharacter):
    def __str__(self):
        return super().__str__() + ' (Digit)'

class PasswordMissingSpecial(PasswordMissingCharacter):
    def __str__(self):
        return super().__str__() + ' (Special)'
class PasswordTooShort(Exception):
    def __str__(self):
        return "The password is too short"
class PasswordTooLong(Exception):
    def __str__(self):
        return "The password is too long"


def check_input(username, password):
    pw_validator = {"Lower": 0, "Upper": 0, "Digit":0, "Special":0}
    try:
        if len(username) < 3:
            raise UsernameTooShort
        if len(username) > 16:
            raise UsernameTooLong
        if len(password) < 8:
            raise PasswordTooShort
        if len(password) > 40:
            raise PasswordTooLong

        for i in range(len(username)):
            if username[i] not in string.ascii_letters + string.digits + '_':
                raise UsernameContainsIllegalCharacter(username[i], i)

        for x in password:
            if x in string.ascii_lowercase:
                pw_validator["Lower"] += 1
            elif x in string.ascii_uppercase:
                pw_validator["Upper"] += 1
            elif x in string.digits:
                pw_validator["Digit"] += 1
            elif x in string.punctuation:
                pw_validator["Special"] += 1


        if pw_validator["Lower"] == 0:
            raise PasswordMissingLowercase
        elif pw_validator["Upper"] == 0:
            raise PasswordMissingUppercase
        elif pw_validator["Digit"] == 0:
            raise PasswordMissingDigit
        elif pw_validator["Special"] == 0:
            raise PasswordMissingSpecial

    except UsernameTooShort as e:
        print(e)
    except UsernameTooLong as e:
        print(e)
    except PasswordTooShort as e:
        print(e)
    except PasswordTooLong as e:
        print(e)
    except UsernameContainsIllegalCharacter as e:
        print(e)
    except PasswordMissingCharacter as e:
        print(e)

    else:
        print("OK")
```

4.1.2

```
def translate(sentence):
    words = {'esta': 'is', 'la': 'the', 'en': 'in', 'gato': 'cat', 'casa': 'house', 'el': 'the'}
    st = ''
    gen = (words[w] for w in sentence.split())
    for word in gen:
        st += word + ' '
    return st
```

4.1.3

```
def is_prime(n):
    # Corner case
    if n <= 1:
        return False
    # Check from 2 to n-1
    for i in range(2, n):
        if n % i == 0:
            return False
    return True

def first_prime_over(n):
    prime_generator = (x for x in range(n+1, n*2) if is_prime(x))
    return next(prime_generator)
```

4.2.2

```
def parse_ranges(ranges_string):
    rs_lst = ranges_string.split(',')
    ps_lst = []
    for st_range in rs_lst:
        start,end = st_range.split('-')
        for num in (x for x in range(int(start), int(end) + 1)):
            ps_lst.append(num)
    return ps_lst
```

4.3.4

```
def get_fibo():
    x = 0
    yield x
    y = 1
    yield y
    while True:
        x, y = y, x
        y += x
        yield y
```

4.4

```
def gen_secs():
    for sec in range(60):
        yield sec
def gen_minutes():
    for minute in range(60):
        yield minute
def gen_hours():
    for hour in range(24):
        yield hour
def gen_time():
    for hour in gen_hours():
        for minute in gen_minutes():
            for sec in gen_secs():
                yield("%02d:%02d:%02d"%(hour, minute, sec))


def gen_years(start=2019):
    while True:
        yield start
        start += 1
def gen_months():
    for month in range(1, 13):
        yield month
def gen_days(month, leap_year=True):
    days = 0
    if month in [1,3,5,7,8,10,12]:
        days = 31
    elif month in [4,6,9,11]:
        days = 30
    elif month == 2:
        if leap_year:
            days = 29
        else:
            days = 28
    for day in range(1,days + 1):
        yield day
def is_leap_year(year):
    if year % 4 != 0:
        return False
    elif year % 100 != 0:
        return True
    elif year % 400 != 0:
        return False
    else:
        return True

def gen_date():
    for year in gen_years():
        for month in gen_months():
            for day in gen_days(month, is_leap_year(year)):
                for gt in gen_time():
                    yield("%02d/%02d/%04d %s"%(day, month, year, gt))


def main():
    gen = gen_date()
    for i in range(0, 10000001):
        if i % 1000000 == 0:
            print(next(gen))
        else:
            next(gen)
```

5.1.2

```
import winsound
def main():
    freqs = {"la": 220,
         "si": 247,
         "do": 261,
         "re": 293,
         "mi": 329,
         "fa": 349,
         "sol": 392,
         }
    notes =
"sol,250-mi,250-mi,500-fa,250-re,250-re,500-do,250-re,250-mi,250-fa,250-sol,250-sol,250-s
ol,500"
    notes = notes.split('-')
    gen_note = (note.split(',') for note in notes)
    for note in gen_note:
        winsound.Beep(freqs[note[0]], int(note[1]))
```

5.2.2

```
  numbers = iter(list(range(1, 101)))
    for i in numbers:
        next(numbers)
        next(numbers)
        print(i)
```

5.2.3

```
def main():
    wallet = "20*3,10*5,5*2,1*5"
    w_lst = []
    for g in wallet.split(','):
        bill, amount = g.split('*')
        w_lst += [int(bill)] * int(amount)
    count = 0
    gen_combination = (itertools.combinations(w_lst, r) for r in range(len(w_lst)))

    for g in gen_combination:

       for comb in set(g):
           if sum(comb) == 100:
               print(comb)
               count += 1
    print(count)
```

5.3.2

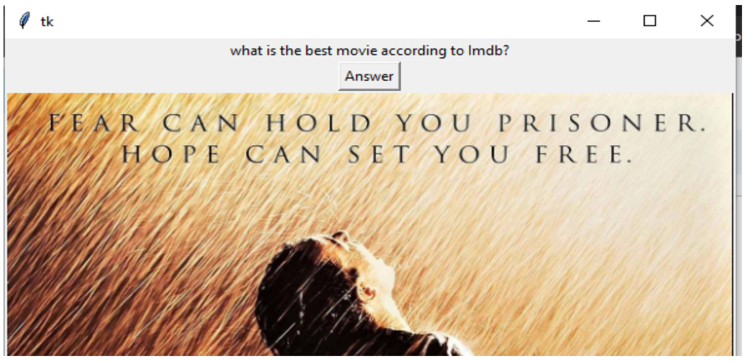
```
class MusicNotes():
    def __init__(self):
        self._notes_lst =[['La', 55], ['Si', 61.74], ['Do', 65.41], ['Re', 73.42], ['Mi', 82.41], ['Fa',
87.31], ['Sol', 98]]
        self.mn_index = 0
        self.oct_index = 0
    def __iter__(self):
        return self
    def __next__(self):
        if self.oct_index > 4:
            raise StopIteration()
        if self.mn_index == len(self._notes_lst):
            self.mn_index = 0
            self.oct_index += 1
        a = self._notes_lst[self.mn_index][1] * (2**(self.oct_index))
        self.mn_index += 1
        return a
```

5.4

```
import functools
def even_mul(num, index):
    """multple by 2 the numbers in a even index.
    :param num: the number to multiple
    :param index: the index
    :type num: int
    :type index: int
    :return: the number after the multiple
    :rtype: int"""
    if index % 2 == 0:
        return num*2
    return num
def num_to_digit_lst(num):
    """Turn a number to a list of his digits.
    :param num: the number to turn
    :type num: int
    :return: list of digits
    :rtype: list"""
    return [int(x) for x in str(num)]
def check_id_valid(id_number):
    """Check if the id number given is valid (by the question definition).
    :param id_number: id number
    :type id_number: int
    :return: True if the number valid, False otherwise
    :rtype: bool"""
    digits_lst = num_to_digit_lst(id_number)
    return (functools.reduce(lambda x, y: x + y,
                            map(lambda x: x if x < 10 else sum(num_to_digit_lst(x)),
                               [even_mul(digits_lst[i],i+1) for i in range(len(digits_lst))]))) % 10 == 0

MAX_NUMBER = 999999999
class IDIterator:
    """
    A class used to represent an ID Iterator
    """
    def __init__(self, id_num):
        """initialize the attribute in the class.
        :param id_num: id number for initializition
        :type id_num: int
        :return: none"""
        self._id = id_num #0 - 999999999

    def __iter__(self):
        """Returns the iterator instance.
        :return: iterator instance"""
        return self

    def __next__(self):
        """Returns the next valid id .
        :return: next valid id
        :rtype: int
        :raise: StopIteration"""
        while self._id <= MAX_NUMBER:
            self._id += 1
            if check_id_valid(self._id):
                return self._id
        raise StopIteration


def id_generator(id_num):
    """Generate the next valid id number.
    :param id_num: the id number
    :type exponent: int
    :return: next valid id number
    :rtype: int"""
    while id_num <= MAX_NUMBER:
            id_num += 1
            if check_id_valid(id_num):
                yield id_num


def main():
    id_number = int(input('Enter ID: '))
    choice = input('Generator or Iterator? (gen/it)? ')
    if choice == 'it':
        id_iter = IDIterator(id_number)
        for i in range(10):
            print(next(id_iter))
    elif choice == 'gen':
        id_gen = id_generator(id_number)
        for i in range(10):
            print(next(id_gen))

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

6.1.3

```
import tkinter
from PIL import Image, ImageTk
def show_picture():
    img = tkinter.PhotoImage(file=r"C:/Users/Alemo/Downloads/Capture.PNG")
    label = tkinter.Label(image=img)
    label.image = img
    label.pack()
def main():
    window = tkinter.Tk()
    tkinter.Label(text = "what is the best movie according to Imdb?").pack()
    tkinter.Button(text = "Answer", command = show_picture).pack()

    window.mainloop()
if __name__ == "__main__":
    main()
```


what is the best movie according to Imdb?

6.1.4

```
import base64

def main():
    base64_message = 
"CgkJICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAuLS0tW1tfX11dLS0tLS4KICAgIC
AgICAgICAgICA7LS0tLS0tLS0tLS0tLS58ICAgICAgIF9fX18KICAgICAgICAgICAgICB8ICAgI
CAgICAgICAgIHx8ICAgLi0tW1tfX11dLS0tLgogICAgICAgICAgICAgIHwgICAgICAgICAgICAg
fHwgIDstLS0tLS0tLS0tLS58CiAgICAgICAgICAgICAgfCAgICAgICAgICAgICB8fCAgfCAgICA
```

```
gICAgICAgfHwKICAgICAgICAgICAgICB8X19fX19fX19fX19fX3wvICB8ICAgICAgICAgICB8f
AogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIHxfX19fX19fX19fX3wvCgo="

    emsg = base64_message.encode()
    message_bytes = base64.b64decode(emsg)
    message = message_bytes.decode('ascii')

    print(message)
```

6.2.5

file1.py

```
class GreetingCard:
    def __init__(self, rt = "Dana Ev", sr = "Eyal Ch"):
        self._recipient = rt
        self._sender = sr
    def greeting_msg(self):
        print(f"recipient:{self._recipient}, sender:{self._sender}")
```

file2.py

```
from file1 import GreetingCard
class BirthdayCard(GreetingCard):
    def __init__(self, rt = "Dana Ev", sr = "Eyal Ch" , sr_age = 0):
        super().__init__(rt, sr)
        self._sender_age = sr_age
    def greeting_msg(self):
        super().greeting_msg()
        print(f"Happy birthday, age:{self._sender_age}")
```

main.py

```
import file2
def main():
    gd = file2.GreetingCard()
    bd = file2.BirthdayCard()
    gd.greeting_msg()
    bd.greeting_msg()
```

6.3.3

```
import gtts
import os

def main():
    text = "first time i'm using a package in next.py course"
    gtts.gTTS(text=text, lang='en', slow=False).save('one.mp3')
    os.system('one.mp3')
```

6.4

```
from PIL import Image, ImageDraw
first = (146, 399, …
second = (156, 141, …
 with Image.open(r"C:\Users\Alemo\Downloads\ex6p4.jpg") as im:
        draw = ImageDraw.Draw(im)
        draw.line(first, fill=(0, 255, 255), width=10)
        draw.line(second, fill=(0, 255, 255), width=10)
        im.show()
```