Task 1

```
#That solution is a bit strange - i use yielding which most advantage is not
#using memory and then I save it into list, but I have to use list -
#otherwise i wouldnt be able to use list comprehension. Tbh I dont have any other idea
def polynomial(c,x):
  def coefficient(c,x):
    for x_element in x:
      for i, c_element in enumerate(c):
        yield c_element*x_element**i

  a=coefficient(c,x)
  yielded_list=[]
  my_list=[]

  for _ in range(len(x)):
    for i in range(len(c)):
      yielded_list.append(next(a))
    my_list.append(sum([e for e in yielded_list]))
    yielded_list=[]

  return my_list

c = [1,2,3]
x = [0, 2, 4, 8]
p=polynomial(c,x)
print(p)
```

```
[1, 17, 57, 209]
```

Task 2

```
import matplotlib.pyplot as plt
x=[0, 2, 4, 8]
y=polynomial(c,x)
plt.plot(x,y)
n=100
xd=[8*x/n for x in range(n)]
yd=polynomial(c,xd)
plt.plot(xd,yd)
```

[<matplotlib.lines.Line2D at 0x7f469f115f28>]

Task 3

```
def upper_letter(letter, string_to_upper):
  alist = [e for e in myString ]
  w=[x if x!='e' else x.upper() for x in string_to_upper]
  b=''.join( w )
  return b


upperstr = 'thepurposeoflife'
upperstr = upper_letter('e', upperstr)
print(upperstr)
```

```
thEpurposEoflifE
```

Task 4

```
records = (('Sam', 19, 'CS'),
('Nicole', 21, 'Biochemistry'),
('Paul', 20, 'Fine Arts'),
('Ashley', 18, 'History'))

def showrecords(records):
  '''Unpack records stored in a tuple of tuples and print each one in a nice format'''
  for my_tuple in records:
    (name, age, course)=my_tuple
    print(f'Name: {name}, age:{age}, course:{course}')
    #print('%s and %d and %s' % (name, age, course)) #we can use example syntax too

showrecords(records)
```

```
Name: Sam, age:19, course:CS
Name: Nicole, age:21, course:Biochemistry
Name: Paul, age:20, course:Fine Arts
Name: Ashley, age:18, course:History
```

Task 5

```
def multiplier_of(n):
  def multiply(m):
    return m*n
  return multiply

# test code
multiply_with_5 = multiplier_of(5)
print(multiply_with_5(9))
# should return 45
multiply_with_45 = multiplier_of(multiply_with_5(9))
print(multiply_with_45(2))
# should return: 90
```

```
45
90
```

Task 6

```
def type_check(correct_type):
  def check(old_function):
    def function_inside(*args, **kwargs):
```

```
      if isinstance(*args, correct_type):
        return old_function(*args)
      else:
        print("Bad type")
    return function_inside
  return check

@type_check(int)
def times2(num):
  return num*2

print(times2(2))
times2('Not A Number')

@type_check(str)
def first_letter(word):
    return word[0]

print(first_letter('Hello World'))
first_letter(['Not', 'A', 'String'])
```

```
4
Bad type
H
Bad type
```

Task 7

```
import random
PLUGINS = dict()

def register(func):
    PLUGINS[func.__name__]=func

@register
def say_hello(name):
    return f"Hello {name}"

@register
def be_awesome(name):
    return f"Yo {name}, together we are the awesomest!"

def randomly_greet(name):
    greeter, greeter_func = random.choice(list(PLUGINS.items()))
    print(f"Using {greeter!r}")
    return greeter_func(name)

randomly_greet('John')
```

```
Using 'be_awesome'
'Yo John, together we are the awesomest!'
```