Question 1: Basic Python

(a) Re-implement the following code using defaultdict instead of try-except. Include any necessary import.

```
d = {}
dna = "gattaca"
for s in dna:
    try:
        d[s] += 1
    except KeyError:
        d[s] = 1
```

```
from collections import defaultdict
d = defaultdict(int)
dna = "gattaca"
for s in dna:
    d[s] += 1
```

(b) Why will this code fail? How could it be fixed?

```
d = {}
M = [[0,1,2],
     [3,4,5],
     [6,7,8]]
for L in M:
    d[L] = sum(L)
```

This code will fail because lists are unhashable and dictionaries can't have unhashable types as keys. This could be fixed by converting the list to a string type, e.g., d[str(L)] = sum(L).

(c) What is duck typing and how does it relate to polymorphism?

Duck typing is a way of programming in which an object passed into a function or method supports all method signatures and attributes expected of that object. With normal typing, suitability is determined by an object's type. In duck typing, an object's suitability is determined by the presence of certain methods and properties.

Polymorphisms can be used to achieve the same result as duck typing, by overriding the method of the superclass in a subclass.

(d) Give a single floating-point value in which both of the following expressions are False:

```
x >= 3
x <= 3
```

x = np.nan has type float and is false in both expressions.

(e) Here is the code which calculates the factorial function. Write a docstring containing three suitable doctests, one of which tests with an invalid input:

```
def fact(n):
    """Return the factorial of n

    >>> factorial(0)
    1
    >>> factorial(5)
    120
    >>> factorial(-1)
    Traceback (most recent call last):
    ...
    ValueError: Bad input
    """
```

Question 2: Advanced Python

(a) Suppose we have a string s containing an arithmetic expression in Python syntax, such as `s = "5+7"` or `s = "math.sqrt(12+(1+1))"`. Write code which will calculate the value of that expression. Assume math has already been imported.

```
def f(s): return (eval(s))
```

(b) The Fibonacci function below is very inefficient. Use memoisation to improve it: either implement memoisation by hand or import and use a decorator.

```
def memoize(f):
    cache = {}
    def mem_f(*args):
        if args in cache:
            return cache[args]
        else:
            cache[args] = f(*args)
            return cache[args]
    return mem_f
```

```
@memoize
def fibonacci(n):
    if n in (0,1):
        return n
    else:
        return (fibonacci(n-1) +
                fibonacci(n-2))
```

(c) What is introspection? Describe at least two Python facilities for introspection.

In programming, introspection refer to a few ways in which a program can inspect its own properties while running (as distinct from compile-time). Examples of introspection are:

- Finding out the type of an object
- Accessing docstrings, function names, and other function properties.
- Accessing filenames and line numbers of the source file;
- Accessing the callstack
- Accessing the configuration of the interpreter.

In Python:
- `callable(f)`: is f callable?
- `help`: access docstring

(d) Define the term higher-order function and give an example of one built-in to Python.

A higher order function is a function that takes a function as an argument, or returns ~~an argument~~ a function. An example in Python is the `map()` function.

`map(fun, iter)`. Map applies the given function to each element of the iterable.

(e) What is the time complexity of this function with respect to the length n of the input L? How would it change if we rewrote this as a comprehension?

```
def f(L):
    result = []
    for x in L:
        for y in L:
            result.append(x*y)
    return sum(result)
```

The number of instructions would be $n^2 + 2 + n^2 - 1 = 2n^2 + 1$

Therefore the complexity would be of $\Theta(n^2)$.

Question 3: Data Science

(a) "A panda DataFrame is like a dictionary with column names as keys and lists/series as values". Fill in the blanks

(b) What is the difference between np.save and np.savetxt? Which will result in a larger file, on disk?

np.save ~ Save an array to a binary file in Numpy .npy format

np.savetxt ~ Save an array to a text file

The larger file on disk would be given by the text format, as the first one is a binary file.

(c) In a scientific context, what is the motivation and purpose of this?

```
import random
random.seed(0)
```

Using seed we are able to get the same results when executing programs that are not deterministic. This ensures that we can replicate the same experiment every time.

(d) Demonstrate in code the core workflow for Scikit-learn Linear Regression, including import, training, evaluation, inspection of the parameters of the trained model, to prediction on unlabelled data. The code should use datasets X_train, y_train, X_test, y_test and an array of query points X_query. You may assume they have already been created in the right format.

```
import numpy as np
from sklearn.linear_model import LinearRegression

reg = LinearRegression().fit(X_train, y_train)
reg.coef_
pred = reg.predict(X_test)
score(y_test, pred)
reg.predict(X_query)
```

(e) Describe the use of inheritance and mixins in the Scikit-Learn API

In Scikit-Learn, each regression model is build as a subclass of RegressorMixin and each classifier is build as a subclass of ClassifierMixin. These superclass provides methods such as scores for the specific types. On top of that, the class BaseEstimator, is use for all models. ~~to provide methods such as~~

• Question 4: Tools and Applications

(a) Suppose we have an electrocardiogram (ECG) time-series signal, stored in a Numpy array. Suppose we pass this array to a Fourier transformation function. What format will the result be, and how can we interpret it to produce an estimate of a heart rate.

The ECG has to be presented in a format of N samples recorded at a sampling frequency f_s [Hz] and a length of a signal L (seconds).

The fourier transform decomposes a signal into a sum of sinusoidal signals of different frequencies. By doing that we can then see how much energy is present at each frequency. The result of applying the fourier transform to a ECG time-series signals would be an array of size $(N,)$ with complex numbers.

FTT is used to convert the ECG signal, i.e, in the time domain, to an ECG signal in the frequency domain for more accurate extraction of the peak values. Frequency-domain shows how much of the signal lies within each given frequency band over a range of frequencies

(b) "NetworkX represents graphs using a dict-of-dicts-of-dicts. Explain this statement with the help of a small example.

An example of a weighted graph G using NetworkX is:

```
G = {0: {1: {"w": 0.5}}, 2: {"w": 0.1}},
    1: {0: {"w": 0.5}, 1: {"w": 0.3}},
    2: {0: {"w": 0.1}}
    }
```

The type of this graph is a dictionary that has nodes as keys and edges as values. Each edge is given by a dictionary in which the keys are nods (where the edge is connected) and the values are attributes of the edge. The attributes of the edge is again a dictionary in which keys are the names of the attributes and values are the value assigned to each attribute.

Therefor, we can say that G is a dict-of-dicts-of-dicts.

(c) Rewrite the following undirected graph in adjacency matrix format. What is the main disadvantadge of this format relative to the NetworkX format?

$G = (V, E)$ where $V = \{0,1,2,3,4\}$ and $E = \{(0,1),(0,2),(0,3),(2,4),(3,4)\}$

The adjacency matrix representing this graph is

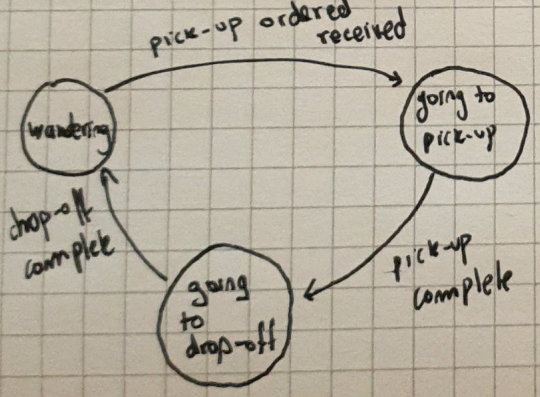
$$A_G = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

As much as this provides a simpler version to represent the graph we are missing any important attributes that the edges may have.

(d) Consider an automated taxi. It has three states: wondering, going to pick-up, going to drop-off. Its sensors can deliver these three inputs: pick-up order received, pick-up complete, drop-off complete. Draw a finite state machine suitable for controlling this taxi.

states = {wandering, going to pick-up, going to drop-off}

inputs = {pick-up ordered received, pick-up complete, drop-off complete}



(e) Write down a grammar in BNF format which can generate any of these three "sentences" below (and others) but cannot generate the last. Remember to say which symbol is the start symbol

the cat sat on the mat
cat sat on the mat
the mat sat on mat
hat the cat the mat

<sentence> ::= (<noun><verb><prep><det><noun>) | (<det><sentence>)
<verb> ::= sat
<det> ::= the
<noun> ::= cat|mat
<prep> ::= on

The non-terminal of the first rule is the start symbol