Charles Thomas Wallace Truscott Watters

declarative vs imperative knowledge

```
+ - * / // % ** ( ) != == += -= *= /= //= %= **= & | ^ ~ << >> &= |= ^= ~= <<= >>=
. * ** [ ] [i:j] [i:j:k] [i][j] [i][j][k] ; : , __ =
```

int, float, string, set, tuple, list, dictionary, class, object, function, function invocation, function return, combined type, expression

operator, operand

True, False, lambda, yield, from x import y as z
assert, global, nonlocal, pass, del
break, continue, is, is not, in, is in

```
try:

except:

finally:

else:

raise:

def function(args):
        body
        return

def main():
        body

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

abstraction, decomposition

```
for x in range(start, stop, step):

for x in a:
        for y in b:

for x in a:
        for y in x:

while(bool):
        while(bool):
```

```
def recursive_function(parameter_one, parameter_two):
        base case x:
        base case y:
        base case z:
        body
        recursive_function(parameter_one - 1, parameter_two - 1)
        return

iteration, recursion

if (bool):
        if (bool):
                if (bool):

                elif (bool):
                elif (bool):
                else:

        elif(bool):
        elif(bool):
        else:

elif(bool):
elif(bool):
else:

match(object):

        case x:
                body
        case y:
                body
        case z:
                body

branching, conditionals, control flow

class Fraction(object):

        def __init__(self, numerator, denominator):

                self.numerator = numerator
                self.denominator = denominator

        def add(self):
                pass

        def subtract(self):
                pass
        def multiply(self):
                pass

        def divide(self):
                pass

"""
Created on Sun Jan  1 03:27:11 2023

@author: Charles
"""

class Person(object):
        def __init__(self, name, date_of_birth, location):
                self.name = name
                self.date_of_birth = date_of_birth
                self.location = location

        def print_name(self):
                print("{}".format(self.name))

class Student(Person):
        def __init__(self, name, grades):
                super().__init__(name, None, "Byron Bay")
                self.grades = grades

        def print_grades(self):
                print("Grades: {}".format(self.grades))

class Employee(Person):
        def __init__(self, name, salary):
                super().__init__(name, None, "Byron Bay")
                self.salary = salary

def print_name(L):
        for n in L:
                n.print_name()
def print_all():
        John = Student("John Wayne", {"6.0001": "B", "PH526": "A"})
        Charles = Employee("Charles Truscott", 50000)
        print_name([John, Charles])

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

Encapsulation, Inheritance, Polymorphism, Generators, Decorators

Standard Library

string

```
textwrap
re
difflib

enum
collections
array
heapq
bisect
queue
struct
weakref
copy
pprintf

functools
itertools
operator
contextlib

time
datetime
calendar

decimal
fractions
random
math
statistics
numpy
pandas
matplotlib

os.path
pathlib
glob
fnmatch
linecache
tempfile
shutil
filecmp
mmap
codecs
io

pickle
shelve
dbm
sqlite3
xml.etree.ElementTree
```

```
csv

zlib
gzip
bz2
tarfile
zipfile

hmac
hashlib

subprocess
signal
threading
multiprocessing
asyncio
concurrent.futures

gettext
locale

ipaddress
socket
selectors
select
socketserver

urllib.parse
urllib.request
urllib.robotparser
base64
http.server
http.cookies
uuid
json
xmlrpc.client
xmlrpc.server
webbrowser

argparse
getopt
readline
getpass
cmd
shlex
configparser
logging
fileinput
atexit
sched
```

pydoc
doctest
unittest
trace
traceback
cgitb
pdb
profile
pstats
timeit
tabnanny
compileall
pyclbr
venv
ensurepip

site
sys
os
platform
resource
gc
sysconfig
scipy
sklearn
tensorflow

Algorithmic Complexity

Object Orientation -> data and method attributes, setters and getters, decorators, magic methods, inheritance, polymorphism, encapsulation
Requirements Analysis
Algorithms and Data Structures / Sorting and Searching / Algorithm Design / Data Structure Design