Charles Thomas Wallace Truscott Certificate in Computer Science and Programming using Python (6.0001)

MIT

declarative vs imperative knowledge

```
+ - * / // % ** ( ) += -= *= /= //= %= **=

!= == and or not < <= > >=

& | ^ ~ << >> &= |= ^= ~= <<= >>=

. * ** [ ] [:] [::] [i][j][k] ; __ : , =
```

int, float, string, set, tuple, list, dictionary, boolean, class, object, method, function, function invocation, function return, expression, combination of types

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

try, except, finally, else, error as e, e.method
```

```
def function(args):
       body
       return
```

abstraction, decomposition

```
def main():
       body

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

```
for x in range(start, stop, step):
for x in iterable: (e.g. set, tuple, list, dictionary, string)

for x in a:
       for y in b:

for x in a:
       for y in x:

while (Boolean):
       while (Boolean):

def recursive(a, b):
       base case 1:
       base case 2:
       base case x:
       body
       recursive call
       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, control flow, conditionals

Algorithmic Complexity
Standard Library
Approximation, Searching and Sorting
Object Oriented Programming
Program Definition, Requirements Analysis, System Theory
Algorithms and Data Structures Design
Thought Experiment

Algorithmic Complexity

- random access machine (sequential execution of steps one step at a time)
- step (an operation that takes a fixed amount of time)
- time constraint (a constraint on the time a program has to run)
- size of input (arbitrarily large or small, as a factor of the time spent in execution)
- dominant algebraic term (of running time of the algorithm)
- running time (actual [seconds], conceptual [algebraic equation])
    - best case, worst case, average case
    - lower bound, upper bound
- counting operations, operators, iteration, recursion, branches, variables
- O(n) n -> infinity, asymptotic notation
- theta(n) n -> any value, n -> 0 and n -> infinity, theta notation
- functionality leading dominant term in respect of variable length input

constant, linear, logarithmic, log linear, polynomial, exponential, graphing algorithmic complexity, counting steps inline

Standard Library

string
textwrap
re
difflib

enum
collections
array
heapq

```
bisect
queue
struct
weakref
copy
pprint

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

Exhaustive Enumeration, Bisection Search, Newton's Method, Bubble Sort, Permutation Sort, Selection Sort, Merge Sort

OO -> setters, getters, data and method attributes, magic methods, polymorphism, inheritance

Program Definition -> Define input and output, specify design of functions and objects
Requirements Analysis -> Define the nonfunctional and functional requirements of the program
System Theory -> Define how the software system functions, treat the software system by its characteristics

Algorithm -> A set of definitions given and steps taken in order to solve a well-formulated problem
Data Structure -> A way of organising and formatting data for an algorithm to process
Data -> A fundamental and primitive unit of information

Exhaustive Enumeration
Branch and Bound
Greedy Algorithm
Dynamic Programming
Recursive Algorithm
Divide and Conquer
Randomized Algorithm

Sum of Two Digits
Maximum Pairwise Product
Fibonacci Number Generation
Greatest Common Divisor
Least Common Multiple

Fibonacci Number Example #2
Last Digit of the Sum of Fibonacci Numbers
Last Digit of the Partial Sum of Fibonacci Numbers
Last Digit of the Sum of Squares of Fibonacci Numbers
Money Change

Maximizing the Value of the Loot
Car Fueling
Maximum Product of Two Sequences
Covering Segments by Points
Distinct Summands
Largest Concatenate

Sorted Array Multiple Search
Majority Element
Improving QUICKSORT
Number of Inversions

Points and Segments
Closest Points

Money Change Again
Primitive Calculator
Edit Distance (Strings)

Longest Common Subsequence of Two Sequences
Longest Common Subsequence of Three Sequences
Maximum Amount of Gold

3-Partition
Maximum Value of an Arithmetic Expression

Eight Queens
Hanoi Towers
Covering Segments by Points
Overlapping Segments
Largest Concatenate
Black and White Squares
Twenty-One Questions
Book Sorting
Number of Paths
Antique Calculator
Two Rocks
Three Rocks
Map Coloring
Clique Finding
Icosian Game
Guarini Puzzle
Room Assignment
Tree Construction
Subway Lines

DATA STRUCTURES

- Arrays and Lists
- Priority Queues
- Disjoint Sets
- Hash Tables
- Binary Search Trees

Algorithms on Graphs

- Graphs Decomposition
- Shortest Path
- Minimum Spanning Tree
- Shortest Paths Examples

Algorithms on Strings

- Pattern Matching
- Suffix Trees
- Suffix Arrays
- Burrows-Wheeler Transform

Advanced Algorithms and Complexity

- Flows in Networks
- Linear Programming
- NP-complete problems
- Coping with NP-completeness
- Streaming Algorithms

privileged characteristic
prototype characteristic
proxy characteristic

demonstration
derivativity of theory
dispensability of theory
dissolving and solving resolutions

reduction
reasoning
identity, entity, preserving, creating
exceptional cases
examples
argumentation
logic

imaginary cases
imaginary scenarios
thought experiment