Charles Thomas Wallace Truscott Watters

Typed from memory

declarative knowledge vs. imperative knowledge

int, float, string, set, tuple, list, dictionary, class, object, function, expression, combined type, operator, operand

+ - * / // % ** ( )

+= -= *= /= //= %= **=

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

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

. * ** [ ] [start:stop] [start:stop:step] [i][j] [i][j][k]

@ __ ; : , =

break, continue, is, is not, in, is in

assert, global, nonlocal, pass, del

True, False, lambda, yield, from x import y as z

try, except, finally, else, as, raise

```
def function(parameter_one, parameter_two):
    body
    return
```

abstraction, decomposition

```
def main():
```

```
        body

if __name__ == "__main__": main()

class Person(object):
        def __init__(self, name, age):
                self.name = name
                self.age = age
        def print_name(self):
                print("{}".format(self.name))
        def print_age(self):
                print("{}".format(self.age))

class Employee(Person):
        def __init__(self, name, age, salary):
                super().__init__(name, age)
                self.salary = salary
        def print_salary(self):
                print("{}'s salary is: {}".format(self.name, self.salary))

class Student(Person):
        def __init__(self, name, age, grades):
                super().__init__(name, age)
                self.grades = grades

        def print_grades(self):
```

```
                print("{}'s grades are: {}".format(self.name, self.grades))

In [3]: Charles = Employee("Charles Truscott", "30", "65000")

In [4]: Tai = Student("Tai Truscott", "32", {'BIO123': 'A+', 'JRNLSM1': 'A++'})

In [5]: Tai.print_grades()
Tai Truscott's grades are: {'BIO123': 'A+', 'JRNLSM1': 'A++'}

In [6]: for obj in (Charles, Tai):
   ...:     obj.print_name()
   ...:
Charles Truscott
Tai Truscott

In [7]:

for x in range(start, stop - 1, step):

for x in a:
        for y in b:

for x in a:
        for y in x:

while(bool):
```

```
        while(bool):


def recursive_function(a, b):
        base case x
        base case y
        base case z
        body
        recursive_function(a - 1, b - 1) (`Recursive Call`)
        return


iteration, recursion


if
        if
                if


                elif
                elif
                else


        elif
        elif
        else


elif
elif
```

else

branching, control flow, conditionals

Sorting and Searching

Standard Library

Algorithms and Data Structures

- Algorithm -> A set of definitions given and steps taken in order to solve a well formulated problem

- Data Structure -> A way of formatting and organising data for processing in an algorithm

- Data -> A primitive and fundamental unit of information (e.g. mass, speed, time, numerical coefficients, representation of something in the real world)

Requirements Analysis, Prototyping

- Defining input and output

- Defining Functional and Nonfunctional Requirements

- Exception handling and bounds on inputs

- Unit Testing

Computational Complexity

- step

- random access machine

- time constraint

- dominant algebraic term

- best case, worst case, average case

    - lower / upper bound

- counting steps, operators, operands, iteration, recursion, function calls

- additive and multiplicative terms

- O(n), asymptotic notation, Big O, upper bound

- theta(n), Big Theta notation, lower bound

- constant, linear, logarithmic, log linear, polynomial, exponential

Charles Thomas Wallace Truscott Watters, educated at MIT and Harvard University, 6.001x and PH526