Algorithms, Abstraction, and Functions
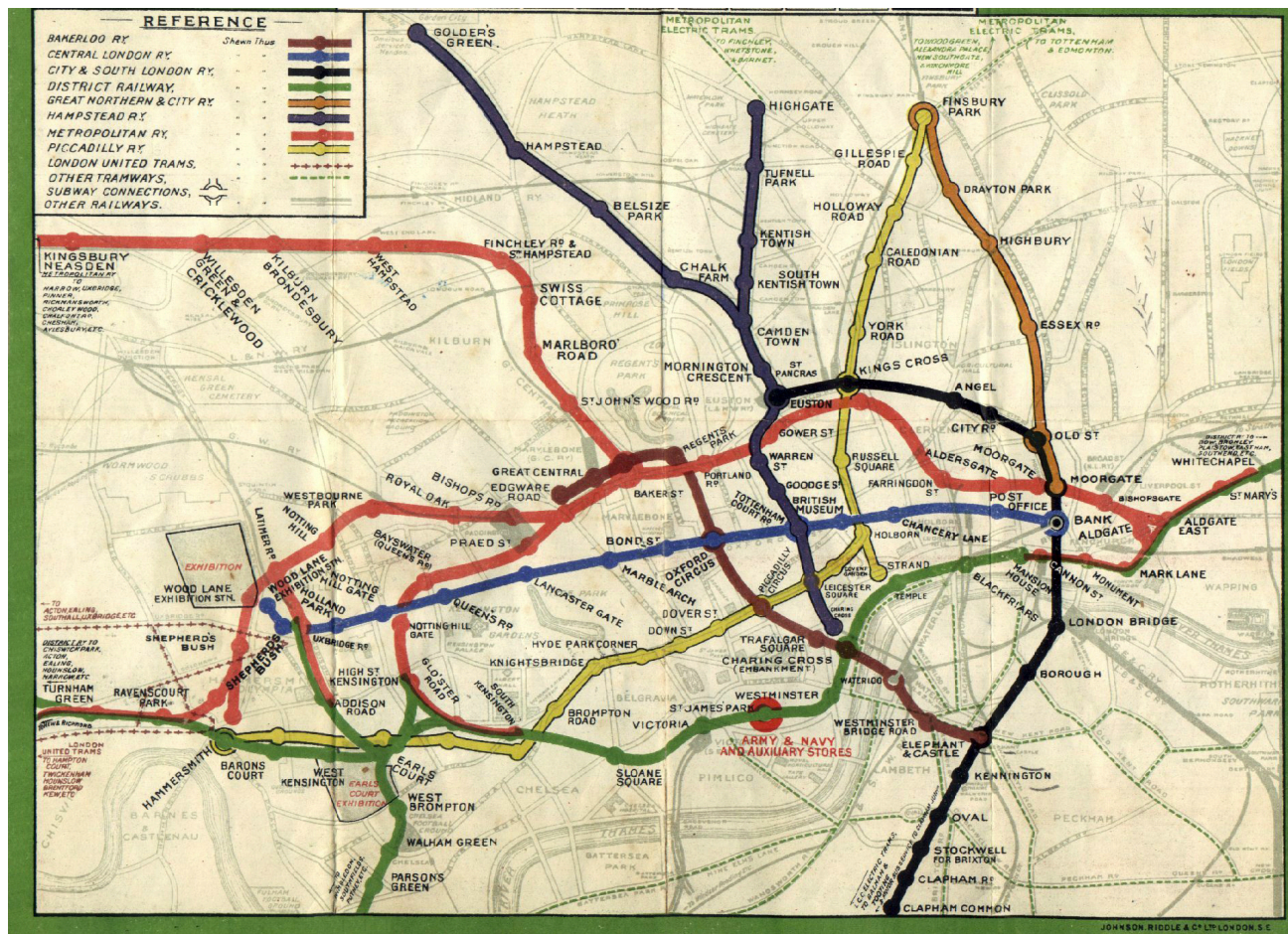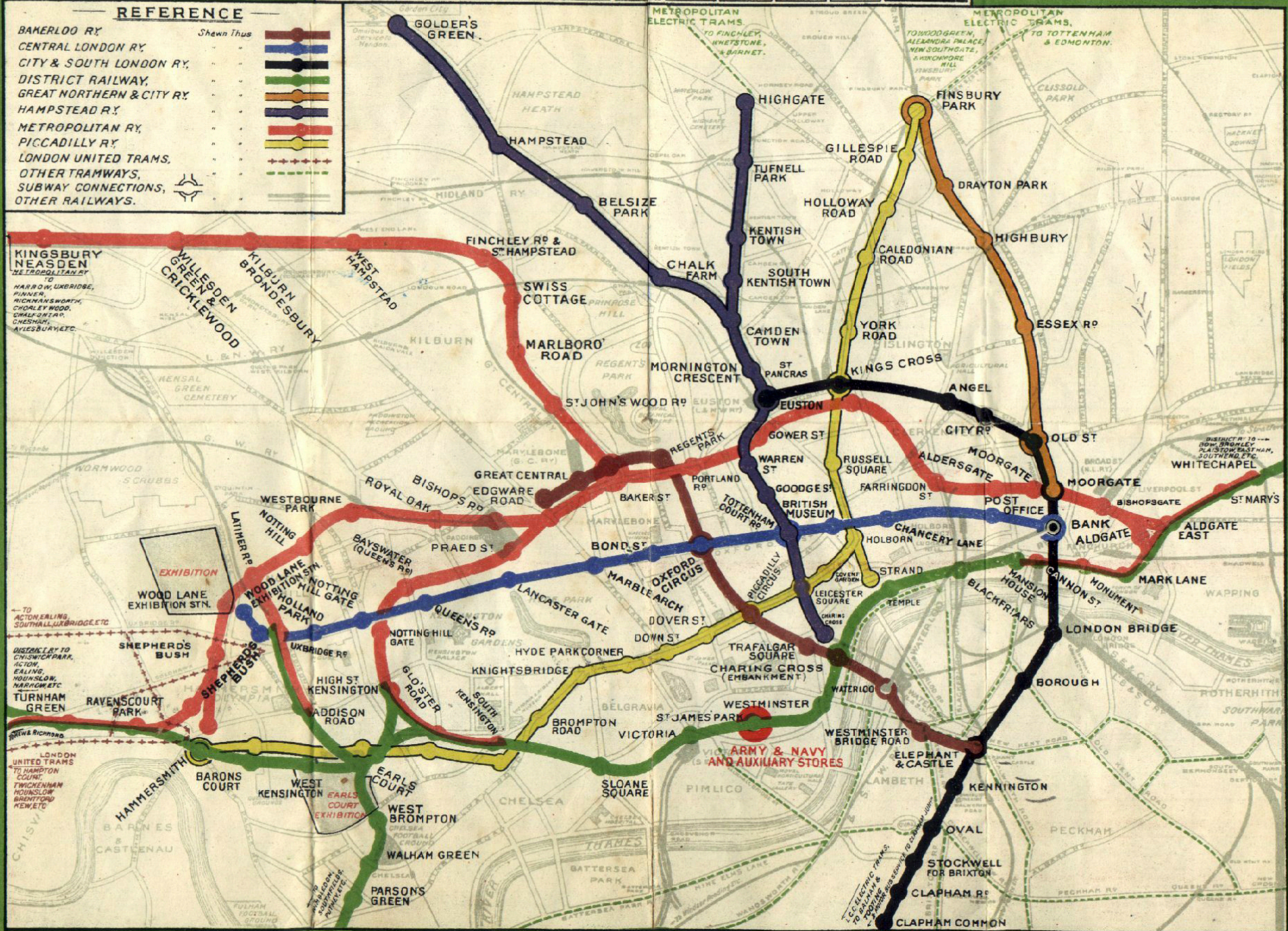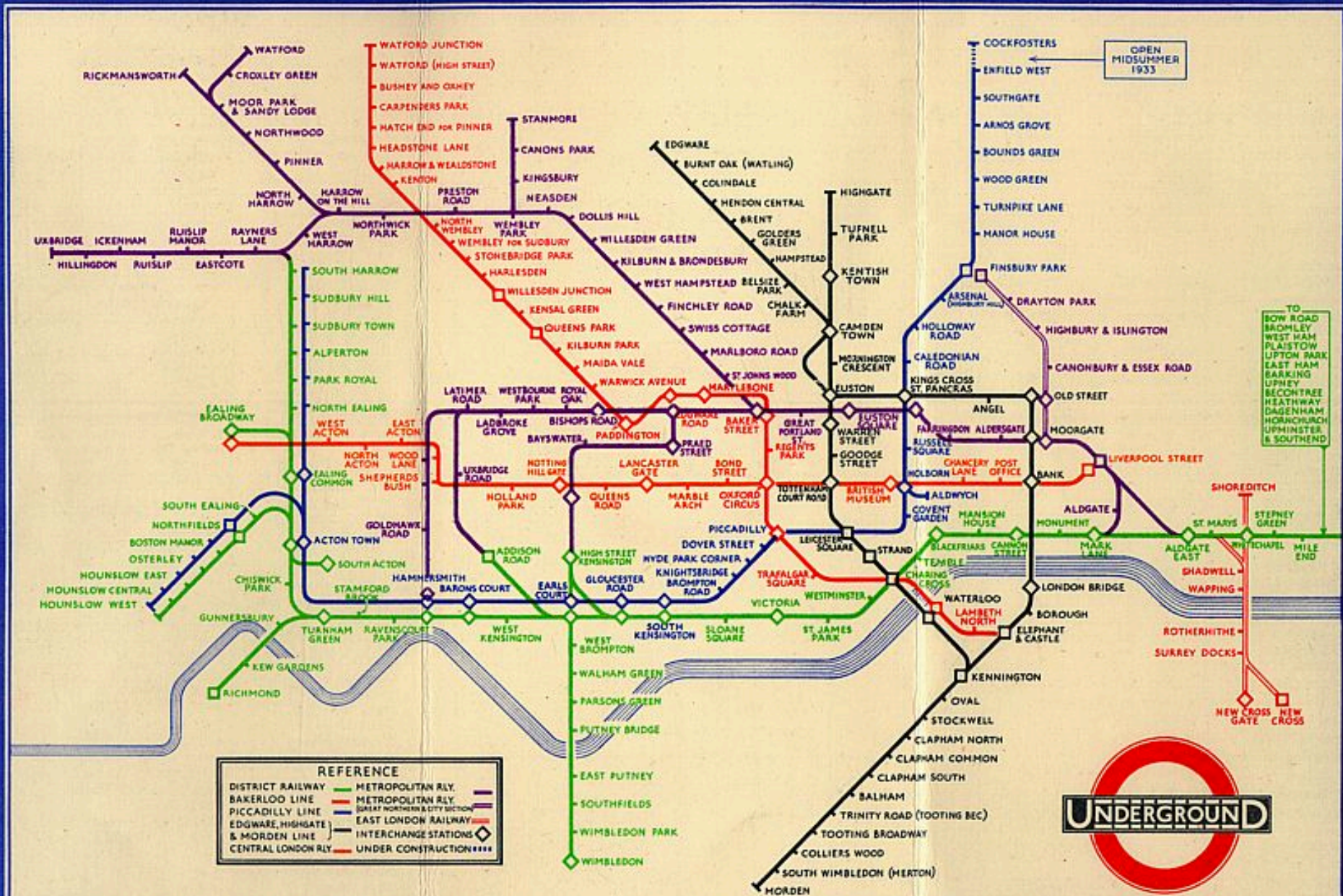
REFERENCE

BAKERLOO RY.
CENTRAL LONDON RY.
CITY & SOUTH LONDON RY.
DISTRICT RAILWAY,
GREAT NORTHERN & CITY RY.
HAMPSTEAD RY.
METROPOLITAN RY.
PICCADILLY RY.
LONDON UNITED TRAMS,
OTHER TRAMWAYS,
SUBWAY CONNECTIONS,
OTHER RAILWAYS.

Shewn Thus

METROPOLITAN ELECTRIC TRAMS. TO FINCHLEY, WHETSTONE, & BARNET.

METROPOLITAN ELECTRIC TRAMS. TO WOOD GREEN, ALEXANDRA PALACE, NEW SOUTHGATE, & WINCHMORE HILL FINSBURY PARK. TO TOTTENHAM & EDMONTON.

GOLDER'S GREEN.
HIGHGATE
HAMPSTEAD
TUFNELL PARK
BELSIZE PARK
FINCHLEY RD. & STH HAMPSTEAD
HOLLOWAY ROAD
KENTISH TOWN
CHALK FARM
SOUTH KENTISH TOWN
SWISS COTTAGE
WEST HAMPSTEAD
GILLESPIE ROAD
DRAYTON PARK
HIGHBURY
CALEDONIAN ROAD
FINSBURY PARK

KINGSBURY NEASDEN
METROPOLITAN RY. TO HARROW, UXBRIDGE, PINNER, RICKMANSWORTH, CHALFONT RD., CHESHAM, AYLESBURY, ETC.

WILLESDEN GREEN & CRICKLEWOOD
KILBURN BRONDESBURY
MARLBORO' ROAD
ST JOHN'S WOOD RD.
CAMDEN TOWN
MORNINGTON CRESCENT
ST PANCRAS
KINGS CROSS
YORK ROAD
ESSEX RD.
ANGEL
OLD ST.
CITY RD.
MOORGATE
ALDERSGATE
WHITECHAPEL

GREAT CENTRAL
EDGWARE ROAD
GOWER ST.
WARREN ST.
PORTLAND RD.
BAKER ST.
TOTTENHAM COURT RD.
GOODGE ST.
BRITISH MUSEUM
RUSSELL SQUARE
FARRINGDON ST.
POST OFFICE
MOORGATE
BISHOPSGATE
ST MARYS

ROYAL OAK
BISHOPS RD.
WESTBOURNE PARK
NOTTING HILL
BAYSWATER (QUEEN'S RD.)
PRAED ST.
BOND ST.
OXFORD CIRCUS
CHANCERY LANE
HOLBORN
BANK ALDGATE
ALDGATE EAST

LATIMER RD.
WOOD LANE EXHIBITION STN.
EXHIBITION
WOOD LANE EXHIBITION STN.
NOTTING HILL GATE
HOLLAND PARK
QUEENS RD.
MARBLE ARCH
PICCADILLY CIRCUS
LEICESTER SQUARE
COVENT GARDEN
STRAND
TEMPLE
MARK LANE
MANSION HOUSE
CANNON ST.
MONUMENT
WAPPING

SHEPHERD'S BUSH
UXBRIDGE RD.
SHEPHERDS BUSH
NOTTING HILL GATE
LANCASTER GATE
DOVER ST.
DOWN ST.
HYDE PARK CORNER
CHARING CROSS
TRAFALGAR SQUARE
WATERLOO
BLACKFRIARS
LONDON BRIDGE

TURNHAM GREEN
RAVENSCOURT PARK
HIGH ST. KENSINGTON
GLO'STER ROAD
ADDISON ROAD
KNIGHTSBRIDGE
BROMPTON ROAD
ST JAMES PARK
WESTMINSTER
CHARING CROSS (EMBANKMENT)
WESTMINSTER BRIDGE ROAD
BOROUGH

HAMMERSMITH
BARONS COURT
WEST KENSINGTON
EARLS COURT
EARLS COURT EXHIBITION
SOUTH KENSINGTON
VICTORIA
SLOANE SQUARE
ARMY & NAVY AND AUXILIARY STORES
ELEPHANT & CASTLE
KENNINGTON

LONDON UNITED TRAMS TO HAMPTON COURT, TWICKENHAM, HOUNSLOW, BRENTFORD, KEW, ETC.

WEST BROMPTON
WALHAM GREEN
PARSONS GREEN
OVAL
STOCKWELL FOR BRIXTON
CLAPHAM RD.
CLAPHAM COMMON

L.C.C. ELECTRIC TRAMS, TO CLAPHAM & TOOTING.

JOHNSON, RIDDLE & CO. LTD. LONDON, S.E.

Abstraction

"The essence of abstractions is preserving information that is relevant in a given context, and forgetting information that is irrelevant in that context."

John V. Guttag,
Introduction to Computation and Programming Using Python

Abstraction

*"The essence of abstractions is **preserving** information that is relevant in a given context, and **forgetting** information that is irrelevant in that context."*

John V. Guttag,
Introduction to Computation and Programming Using Python

- Think of a task or problem where:
  - you used to have to think about each step of the problem
  - but now it's automatic and you don't need to think about each step anymore.

Supercalifragilisticexpialidocious!
Even though the sound of it
Is something quite atrocious
If you say it loud enough
You'll always sound precocious
Supercalifragilisticexpialidocious!

Um diddle diddle diddle um diddle ay
Um diddle diddle diddle um diddle ay!

Because I was afraid to speak
When I was just a lad
My father gave me nose a tweak
And told me I was bad
But then one day I learned a word
That saved me achin' nose
The biggest word I ever heard
And this is how it goes: Oh!

Supercalifragilisticexpialidocious!
Even though the sound of it
Is something quite atrocious
If you say it loud enough
You'll always sound precocious
Supercalifragilisticexpialidocious!

Um diddle diddle diddle um diddle ay
Um diddle diddle diddle um diddle ay!

He traveled all around the world
And everywhere he went
He'd use his word and all would say
"There goes a clever gent"
When Dukes and maharajas
Pass the time of day with me
I say me special word and then
They ask me out for tea

Supercalifragilisticexpialidocious!
Even though the sound of it
Is something quite atrocious
If you say it loud enough
You'll always sound precocious
Supercalifragilisticexpialidocious!

Supercalifragilisticexpialidocious!
Even though the sound of it
Is something quite atrocious
If you say it loud enough
You'll always sound precocious
Supercalifragilisticexpialidocious!

Um diddle diddle diddle um diddle ay
Um diddle diddle diddle um diddle ay!

Because I was afraid to speak
When I was just a lad
My father gave me nose a tweak
And told me I was bad
But then one day I learned a word
That saved me achin' nose
The biggest word I ever heard
And this is how it goes: Oh!

Supercalifragilisticexpialidocious!
Even though the sound of it
Is something quite atrocious
If you say it loud enough
You'll always sound precocious
Supercalifragilisticexpialidocious!

Um diddle diddle diddle um diddle ay
Um diddle diddle diddle um diddle ay!

He traveled all around the world
And everywhere he went
He'd use his word and all would say
"There goes a clever gent"
When Dukes and maharajas
Pass the time of day with me
I say me special word and then
They ask me out for tea

Supercalifragilisticexpialidocious!
Even though the sound of it
Is something quite atrocious
If you say it loud enough
You'll always sound precocious
Supercalifragilisticexpialidocious!

Abstraction in Programming: Functions

- A *function* is used to assign a name to a mini-program within a larger program.

- Any time you want to run the mini-program, you can use the name of the function instead of repeating the code for the mini-program.

- To use a function, we must *define* it first.

Defining a function

Gives your function a name so it can be run later

Syntax:

```
def name_of_function():
    statement      # Notice how these lines are
    statement      # indented.  A group of indented
    statement      # lines is called a block.
    ...            # This is how Python knows which
                   # lines of code go with the
                   # function.
```

Pick a name for your function that describes what it does!
(Just like you pick variable names that describe what the variable holds.)

Defining a function
Gives your function a name so it can be run later

Syntax:

```python
def print_chorus():
  print("Supercalifragilisticexpialidocious!")
  print("Even though the sound of it")
  print("Is something quite atrocious")
  print("If you say it loud enough")
  print("You'll always sound precocious")
  print("Supercalifragilisticexpialidocious!")
```

Functions

- To use a function, we must *define* it first.
- After defining a function, to run the code inside, you *call* the function.

Calling a function

Runs the code inside the function definition

Syntax:

name_of_function()

After defining a function, you can call it any number of times you want.

Each time it is called Python acts as if you had typed in all of the lines of the function definition.

- You are in charge of dessert for Thanksgiving dinner. You decide to make two pumpkin pies and an apple pie.
- Write a program that **defines** three functions:
  - `make_apple()` should print a description of how to make an apple pie.
  - `make_pumpkin()` should print a description of how to make a pumpkin pie.
  - `cook_dessert()` should **call** `make_apple()` and `make_pumpkin()` appropriately to make the pies.

The `main()` function

- Python programs usually include a `main()` function that is the first function that runs when the program begins.
  - This function is in charge of calling any other functions.
- This is not (technically) required in Python, but is a good habit.
  - Required in other languages like C++ and Java.
  - Required for CS 141! ☺

The `main()` function

- From this point on, always ***define*** a `main()` function in your programs.

- Always ***call*** the `main()` function as the last line of your program (your `.py` file).

```python
def print_chorus():
    print("Supercali…")
    (etc)


def print_um_diddle():
    print("Um diddle diddle…")
    (etc)


def print_verse1():
    print("Because I was afraid to speak…")
    (etc)


# A function for the "main" program.
def main():
    print_chorus()          # Print the chorus
    print_um_diddle()       # Print the um diddles
    print_verse1()          # Print the 1st verse
    print_chorus()          # Print the chorus again
    print_um_diddle()       # Print the um diddles again
    print_verse2()          # Print the 2nd verse
    print_chorus()          # Print the chorus the last time


main()                      # Start the program
```

- Add code to the `main()` function so that the user is asked how many people are coming to dinner.
  - If 0-4 people are coming, you should make 1 pumpkin and 1 apple pie.
  - If 5-8 people are coming, you should make one extra pumpkin pie (2 pumpkin/1 apple).
  - If more than 8 people are coming, you should make another extra of each pie (4 total, 2 each).
- Use if/elif/else to call `make_pumpkin()` and `make_apple()` the appropriate number of times.
- **Challenge**: Write this code using three if statements rather than if/elif/else. Does this make the code easier or harder to understand?