A set C is countable if it is empty or there is a surjective function from $\mathbb{N}$ to C.

If A, B are countable, so is $A \cup B$.

If A is countable and $B \subseteq A$, then B is coutable.

If A, B are countable, so is $A \times B$.

If $A \neq \varnothing$ and countable and there is a surjective function from A to B then B is countable.

Proof. There is a surjective function $g : \mathbb{N} \to A$. Then $h : \mathbb{N} \to B$ is a surjective function where $h(i) = f(g(i))$ for all $i \in \mathbb{N}$.

quod erat dem ■

☺

Theorem 1

$\mathbb{Q}^+$ is countable;

Proof. Let $f : \mathbb{Z}^+ \times \mathbb{Z}^+ \to \mathbb{Q}^+$ be defined by $f(a, b) = \frac{a}{b}$.

Note that $(1, 2), (2, 4), (3, 6)$ all map to $\frac{1}{2}$ under f.

Since f is surjective, thus $\mathbb{Q}^+$ is countable.

quod erat dem ■

Theorem 2

$\{0, 1\}^*$ = the set of all finite inary sequences is countable.
a lexicographically order: $\lambda, 0, 1, 00, 01, 10, 11, 000, \ldots$.
$g : \mathbb{N} \times \mathbb{N} \to \{0, 1\}^*$, $g(i, j) =$ "j^{th} lexicographically smallest string of length i if $0 \leq j \leq 2^i$; otherwise λ"

There are 2^i binary string of length c.

g	0	1	2	3	4	
0	λ	λ	λ	λ	λ	$\cdots$
1	0	1	λ	λ	λ	$\cdots$
2	00	01	10	11	λ	$\cdots$
3	000					

☺

Theorem 3

The set of all finite strings of ASCII characters is countable.

Corollary 1 – Corollary of Theorem 3

The set of all syntactically correct Python programs is countable.

Theorem 4

A countable union of countable sets is countable

Proof. Suppose C is a countable collection of countable sets.

Then there is a surjective function $f : \mathbb{N} \to C$ [$f(i)$ is a set in C] and

for each $S \in G$, there is a surjective function on $g_s : \mathbb{N} \to S$

(now we want to prove $\bigcup\{S \mid S \in C\} = \bigcup\{f(i) \mid i \in \mathbb{N}\}$)

Consider the function $h : \mathbb{N} \times \mathbb{N} \to \bigcup\{S \mid S \in C\}$ such that $h(i, j) = g_{f(i)}(j)$ for the j^{th} element of $f(i)$.

Let $X \in \bigcup\{S \mid S \in C\}$ be arbitrary;

Then there exists $S \in C$ such that $x \in S$, and there exists $i \in \mathbb{N}$ such that $f(i) = S$.

Since $x \in S$ and g_s is surjective, there exist $j \in \mathbb{N}$ such that $g_s(j) = x$.He

nce $x = g_{f(i)}(j) = h(i, j)$, so h is surjective.

Thus $\bigcup\{S \mid S \in C\} = \bigcup\{f(i) \mid i \in \mathbb{N}\}$ is countable.

quod erat dem ■

Theorem 5

The set $\{0, 1\}^\omega$ of all infinite binary sequences is uncountable MAT 8.1.4.

Proof. Suppose $\{0, 1\}^\omega$ is countable. Then there exists a surjective function $f : \mathbb{N} \to \{0, 1\}^\omega$.

$B(0) = 100011 \cdots$

$B(1) = 011101 \cdots$

$B(2) = 100000 \cdots$

$\vdots$

$B(i) = \{B(i)_j\}_{j \geq 0}$

Let $D =$ the sequence of bits on the diagonal. That is, $D_i = B(i)_i$ for all $i \in \mathbb{N}$.

Let $C =$ the sequence of bits obtained form D by complementing evrry bit. So, $C_i = 1 - B(i)_i$

$D = 110 \cdots$

$C = 001 \ldots$

$C \in \{0, 1\}^\omega$ and B is surjective so $\exists j \in \mathbb{N}$ such that $B(j) = C$. Then $B(j)_j = C_j = 1 - B(j)_j$. This is a contradiction.

quod erat dem ■

Another example of a diagonalization proof.

There is a compile (program) C that determines whether a given ASCII string P is a syntactically correct Python program that makes a single ASCII string as input i.e. $P \in C$.

$G(P)$ outputs True if $P \in C$, False if $P \notin C$.

Want a python program H that takes as input two ASCII strings, P and X such that $H(P, X)$ outputs True if $P \in C$ and $P(x)$ halts False if $P \notin C$ or $P(X)$ runs forever. ☺

Theorem 6 – Halting Problem

No such python program H exists.

Proof. Suppose there has such a python program H.

def t1(P: str, x: str):

⋮

If justreturn is the string

justreturn = "def j(s: str): return s'' $\in C$, then H(justreturn, 'hello') returns True

if goforever is the string

goforever = "def g(t: str): while True: pass" $\in C$, then H(goforever, 'hello') returns False

...

Consider the syntatically correct Python function D:

```
def D(x): if H(x,x): while True: pass else: return True
```

let $d \in C$ be the string,

```
d = def function D(x): if H(x,x): while True: pass else: return True
```

What happens when Drunson inputed from the code of D:

If $H(d, d)$ = false then $D(d)$ returns True

If $H(d, d)$ = true then $D(d)$ goes into a infinite loop

From the definition of H,

If $D(d)$ returns then $H(d, d)$=True

If $D(d)$ goe sinto an infinite loop then $H(d, d)$ = False

This is a contradiction.

quod erat dem ■