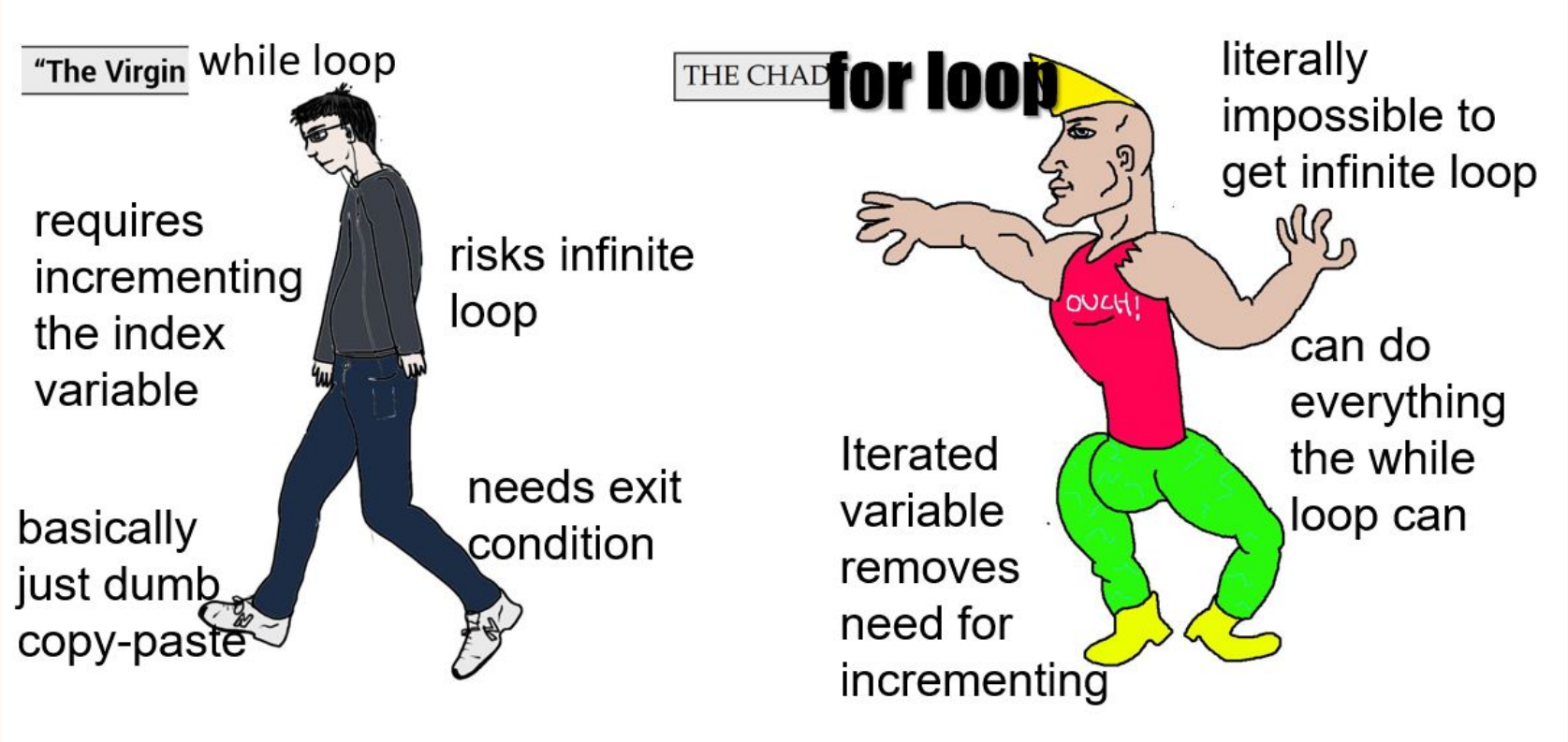
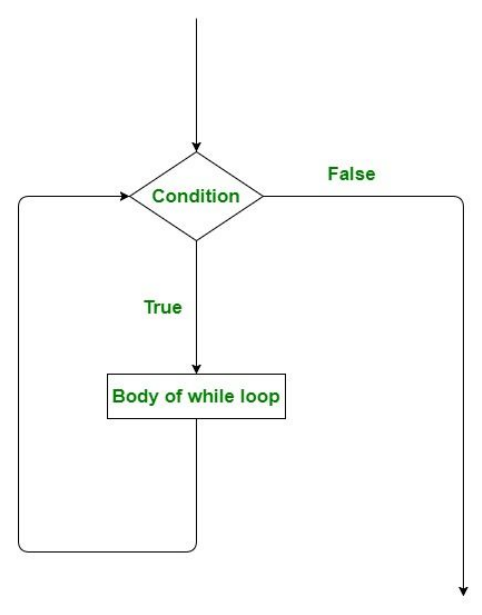
`for` loops


"The Virgin while loop
requires incrementing the index variable
risks infinite loop
needs exit condition
basically just dumb copy-paste
THE CHAD for loop
literally impossible to get infinite loop
OUCH!
can do everything the while loop can
Iterated variable removes need for incrementing

While Loop

```
while <boolean condition>:
  <instructions>
```



For Loop

```
for <item> in <sequence>:
  <instruction>
```

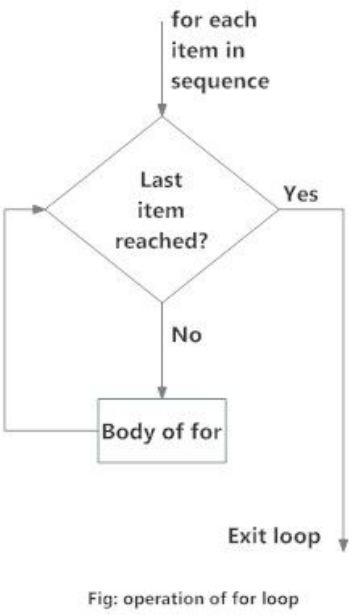


Fig: operation of for loop

for loop

- Repeats same chunk of code **for** each item in a sequence
  - A sequence, or an “iterable”, can be a list, tuple, string etc.
  - Sequence length specifies total number of iterations
  - The <item> becomes an “iterator” variable that can be used inside the loop

```
for <item> in <sequence>:
  <instruction>
```

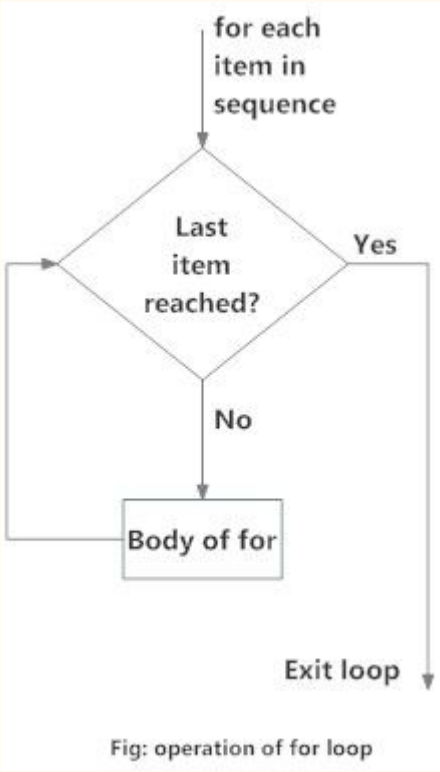


Fig: operation of for loop

`for` loops using iterables

```
for i in [1, 2, 3, 4, 5]:
  print(f"Current i is {i}")
```

```
someStr = "Hello"
for i in someStr:
  print(f"Letter: {i}")
```

```
Letter: H
Letter: e
Letter: l
Letter: l
Letter: o
```

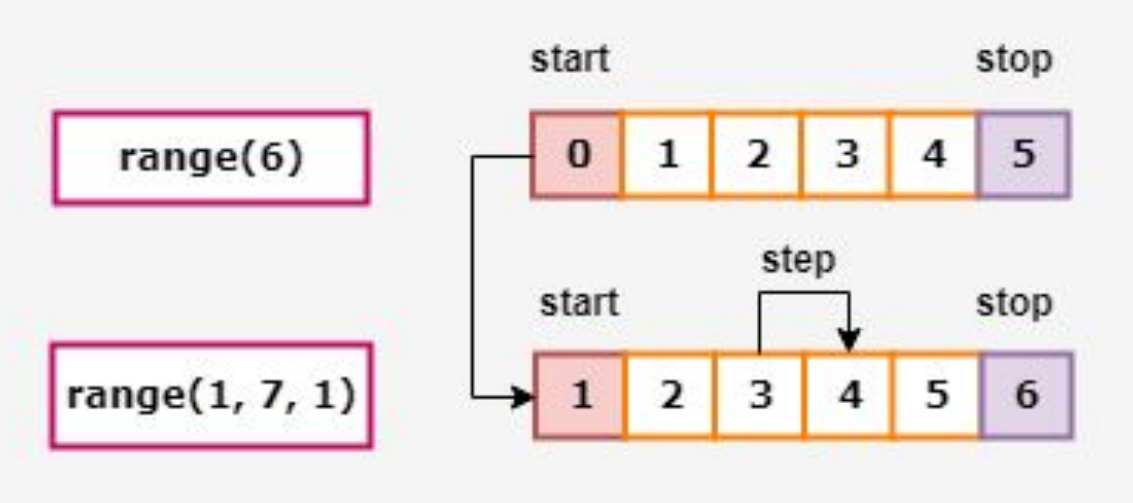
`for` loops using dictionaries

```
myDict = {"a":"x", "b":"y", "c":"z"}

for key in myDict:
  print(f"{ key } -- { myDict[key] }")
```

`range()`

- Generates a sequence of numbers
- `range(n)` generates a sequence from 0 to (`n-1`) with a step-size of 1
- `range(a, n, s)` generates a sequence from starting value `a` to (`n-1`) with a step-size of `s`.



`range()` example

```
for i in range(5):
  print(i)
```

```
0
1
2
3
4
```

```
for i in range(1,6):
  print(i)
```

```
1
2
3
4
5
```

`range()` interactive examples

```
for i in range(1, 10, 2):
  print(i)
```

```
1
3
5
7
9
```

```
for i in range(7):
  print(i)
```

```
0
1
2
3
4
5
6
```

```
for i in range(0, 7, 1):
  print(i)
```

```
0
1
2
3
4
5
6
```

Using `range()` to count

```
for i in range(5):
  print("Hello World")
```

```
Hello World
Hello World
Hello World
Hello World
Hello World
```

Using `range()` for indexing

```
NATO_phonetic_alphabet = ['Alpha', 'Bravo', 'Charlie', 'Delta', 'Echo', 'Foxtrot']
print(len(NATO_phonetic_alphabet))
range(len(NATO_phonetic_alphabet))
list(range(len(NATO_phonetic_alphabet)))
```

```
6
[0, 1, 2, 3, 4, 5]
```

Nested loops

```
for letter in ["A", "B", "C"]:
    for digit in [1, 2, 3]:
        print(letter + str(digit))
```

Summary

```
while <boolean condition>:
  <instructions>
```

```
for <item> in <sequence>:
  <instruction>
```

	`while` loops	`for` loops
Loop is executed ...	… whenever a condition is satisfied	… over the items of a predetermined sequence or iterable (list/tuple/string etc.)
Use when ...	… you don't know how times to run the loop (ie. iterations) but you do know when to stop the loop (ie. the condition)	… you know exactly how many times to run the loop

Further learning

- If you want to learn more about coding, we're making a resource list for self-guided learning
  - Links to websites, courses and cheat-sheets we've found useful
  - Will be posted on the MBSI Coding Workshop Facebook Group Page

- If you get stuck, ask for help!
  - MBSI Coding Workshops Facebook Group

Breakout Time!

FEEDBACK FORM:

https://forms.gle/1GnsHhYUav7D281F8