Lecture 7: Iteration

Comp 102

Forman Christian University

Recap

Recap

```
if <condition> :
    < code >
    < code >
    ...
```

- **Two choices**, we care about **only one**

Recap

```
if <condition> :
      < code >
      < code >
      . . .
```

```
if <condition>:
      < code >
      . . .
else:
      <code>
      . . .
```

- **Two choices**, we care about **both**

Recap

```
if <condition> :
    < code >
    < code >
    ...
```

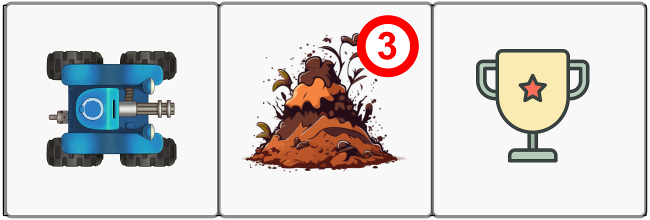
```
if <condition>:
    < code >

    ...
else:
    <code>
    ...
```

```
if <condition>:
    < code >
    ...
elif <condition>:
    <code>
    ...
elif <condition>:
    <code>
    ...
else:
    <code>
    ...
```

- Multiple choices

Loops

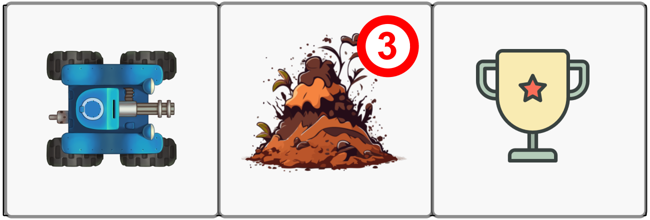
- **Commands Available:**
  - `forward()`
  - `remove()`
  - `repeat n:`



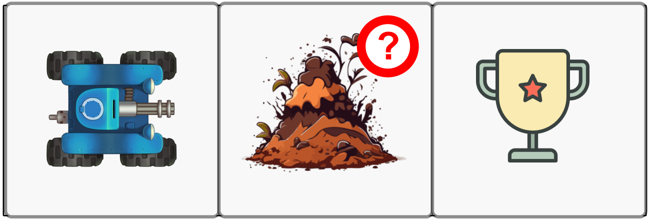
- **Commands Available:**
  - `forward()`
  - `remove()`
  - `repeat n:`

- **Solution:**

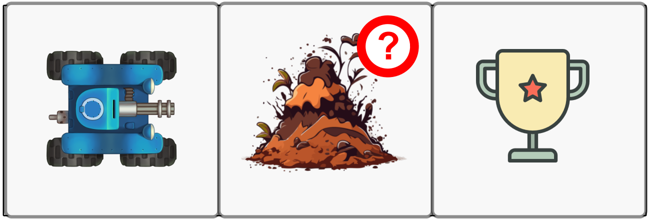
```
repeat 3:
   remove()
forward()
forward()
```



- **Commands Available:**
  - `forward()`
  - `remove()`
  - `repeat n:`

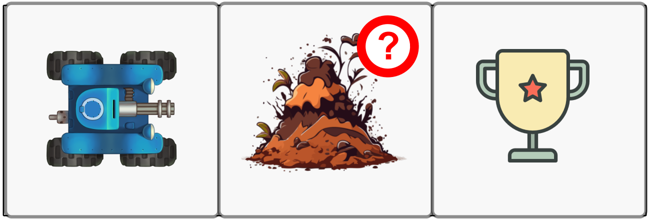


- **Commands Available:**
  - `forward()`
  - `remove()`
  - `repeat n:`
- **Solution Possible?**

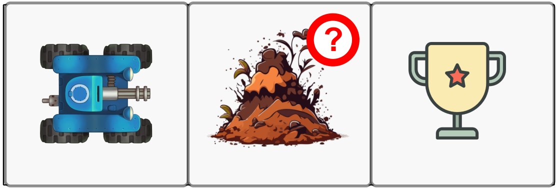


- **Commands Available:**
  - `forward()`
  - `remove()`
  - `repeat n:`

- **Solution Possible?**
  NO



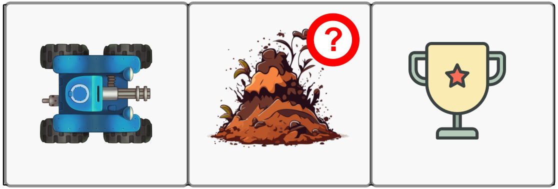
- **Commands Available:**
  - `forward()`
  - `remove()`
  - `repeat n:`
  - `while <condition>:`
  - `has_dirt`



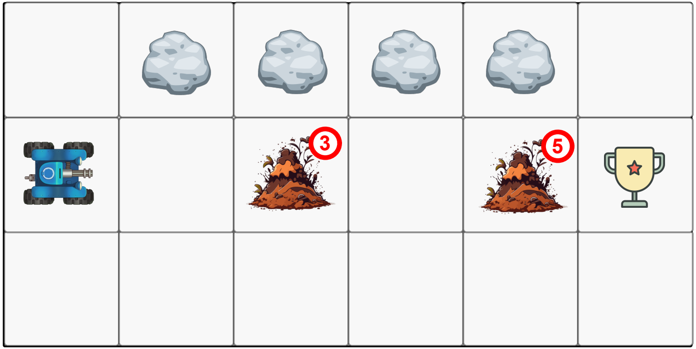
- **Commands Available:**
  - `forward()`
  - `remove()`
  - `repeat n:`
  - `while <condition>:`
  - `has_dirt`

- **Solution:**

```
while has_dirt:
    remove()
forward()
forward()
```

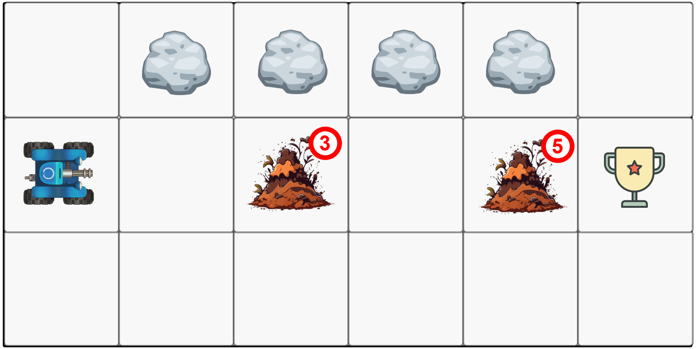


- **Commands Available:**
  - forward()
  - remove()
  - repeat n:
  - has_dirt
  - if <condition>:
  - while <condition>:

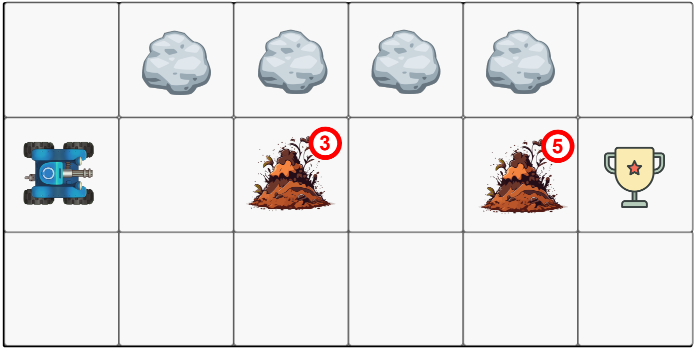


- **Solution:**

```
forward()
repeat 3:
    remove()
forward()
forward()
repeat 5:
    remove()
forward()
forward()
```

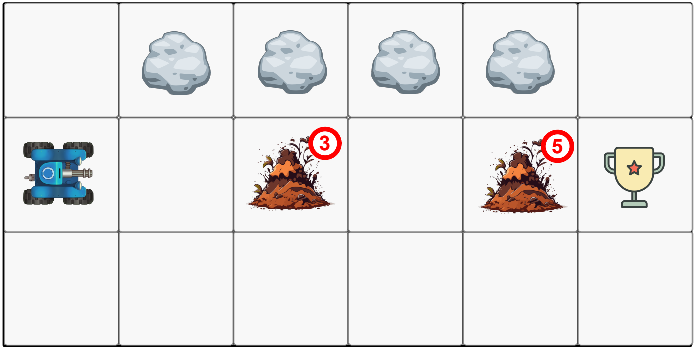


- **Commands Available:**
  - `forward()`×1
  - `remove()`×1
  - `repeat n:`×1
  - `has_dirt`×2
  - `if <condition>:`×1
  - `while <condition>:`×1



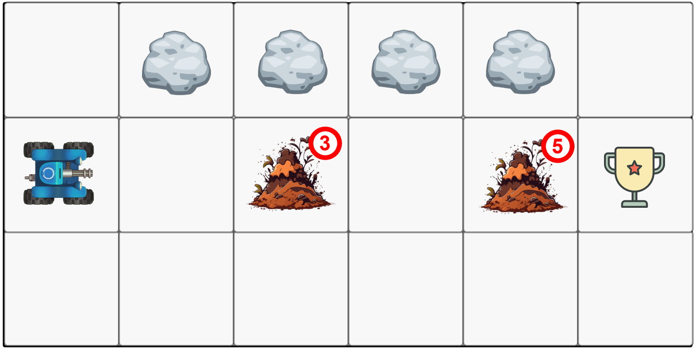
- **Solution:**

```
repeat 5:
   if has_dirt:
      while has_dirt:
         remove()
   forward()
```



- **Solution:**

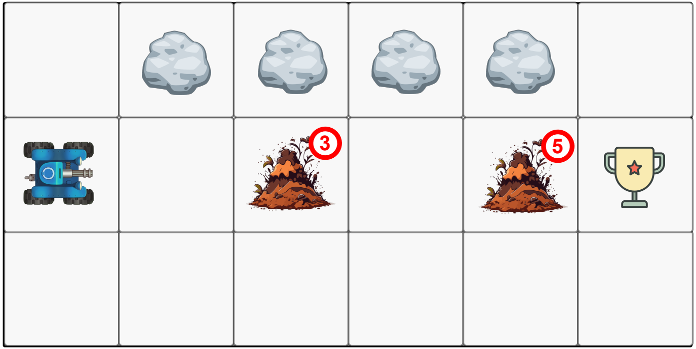
```
repeat 5:
    if has_dirt:
        while has_dirt:
            remove()
    forward()
```



Improve ?

- **Solution:**

```
repeat 5:
    while has_dirt:
        remove()
    forward()
```



While Loop in Python

```
if <condition>:
    <code>
    ...
```

While Loop in Python

while

~~if~~ <condition>:

    <code>
    ...

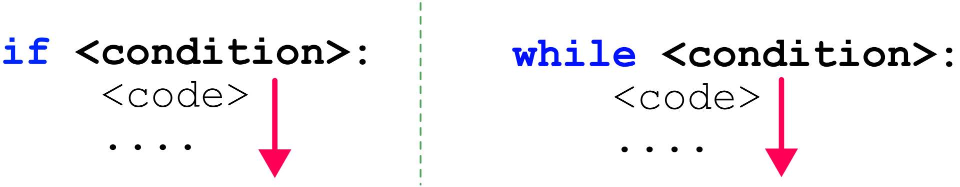
if vs while

```
if <condition>:
    <code>
    ....
```

```
while <condition>:
    <code>
    ....
```

if vs while

Both go in when condition is True

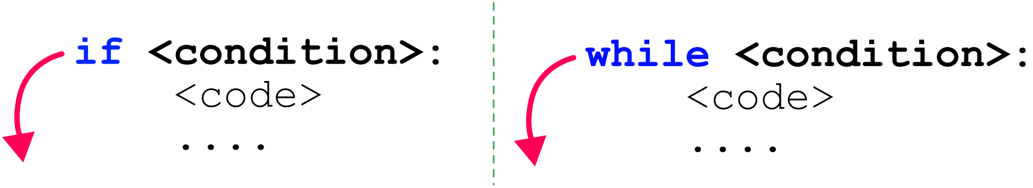


```
if <condition>:
    <code>
    ....
```

```
while <condition>:
    <code>
    ....
```

if vs while

Both go out when condition is False



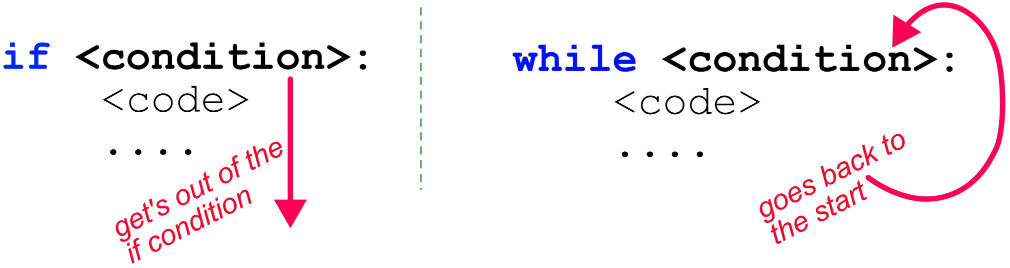
```
if <condition>:
    <code>
    ....
```

```
while <condition>:
    <code>
    ....
```

if vs while

runs only Once

runs multiple times



```
if <condition>:
    <code>
  ....
```

get's out of the if condition

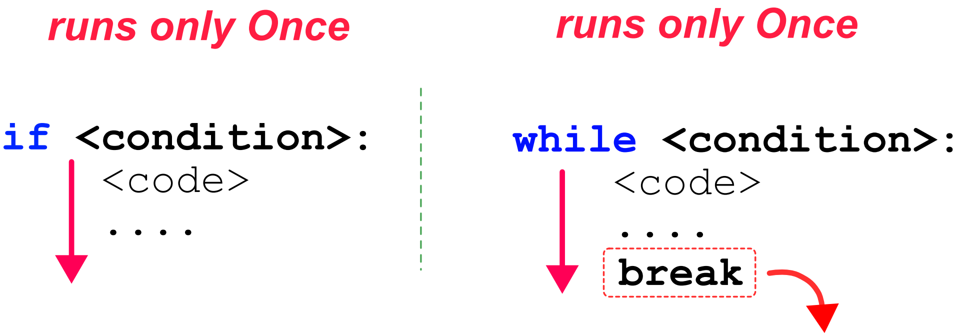
```
while <condition>:
    <code>
  ....
```

goes back to the start

Big Idea

while loop is a supercharged **if statement**

The **break** keyword causes the loop to terminate immediately



runs only Once

```
if <condition>:
    <code>
    ....
```

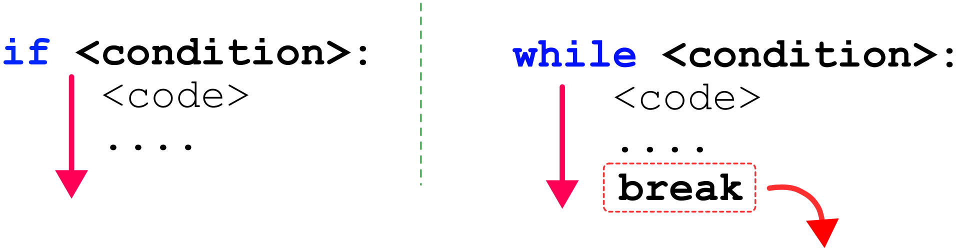
runs only Once

```
while <condition>:
    <code>
    ....
    break
```

The **break** keyword causes the loop to terminate immediately

runs only Once

runs only Once



```
if <condition>:
    <code>
    ....
```

```
while <condition>:
    <code>
    ....
    break
```

Now they both behave exactly the same!

while loop Example

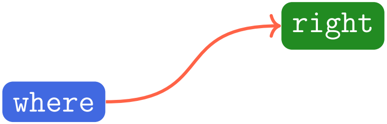
```
You are in the Lost Forest.
************
************
☺
************
************
Go left or right?
```

Program:

```python
where = input("You’re in the Lost Forest. Go left or right? ")
while where == "right":
    where = input("You’re in the Lost Forest. Go left or right? ")
print("You got out of the Lost Forest!")
```

while loop Example

```
You are in the Lost Forest.
************
************
☺
************
************
Go left or right?
```

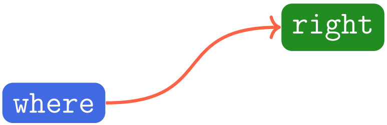


Program:

```python
where = input("You're in the Lost Forest. Go left or right? ")
while where == "right":
    where = input("You're in the Lost Forest. Go left or right? ")
print("You got out of the Lost Forest!")
```

while loop Example

```
You are in the Lost Forest.
************
************
☺
************
************
Go left or right?
```



where

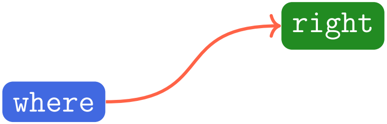
right

Program:

```
where = input("You're in the Lost Forest. Go left or right? ")
while where == "right":
    where = input("You're in the Lost Forest. Go left or right? ")
print("You got out of the Lost Forest!")
```

while loop Example

```
You are in the Lost Forest.
************
************
☺
************
************
Go left or right?
```



where

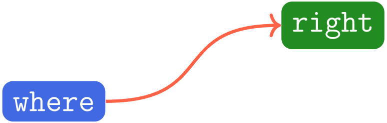
right

Program:

```
where = input("You're in the Lost Forest. Go left or right? ")
while where == "right":
    where = input("You're in the Lost Forest. Go left or right? ")
print("You got out of the Lost Forest!")
```

while loop Example

```
You are in the Lost Forest.
************
************
☺
************
************
Go left or right?
```



where

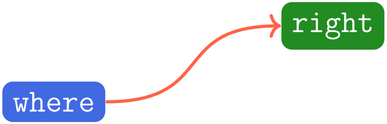
right

Program:

```
where = input("You're in the Lost Forest. Go left or right? ")
while where == "right":
    where = input("You're in the Lost Forest. Go left or right? ")
print("You got out of the Lost Forest!")
```

while loop Example

```
You are in the Lost Forest.
************
************
☺
************
************
Go left or right?
```

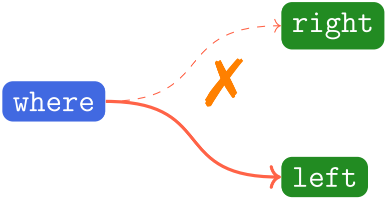


Program:

```
where = input("You’re in the Lost Forest. Go left or right? ")
while where == "right":
    where = input("You’re in the Lost Forest. Go left or right? ")
print("You got out of the Lost Forest!")
```

while loop Example

```
You are in the Lost Forest.
************
************
☺
************
************
Go left or right?
```



where

right

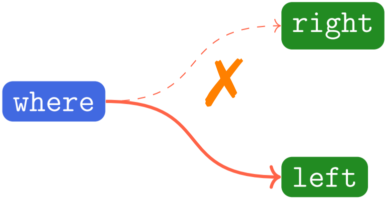
left

Program:

```
where = input("You’re in the Lost Forest. Go left or right? ")
while where == "right":
    where = input("You’re in the Lost Forest. Go left or right? ")
print("You got out of the Lost Forest!")
```

while loop Example

```
You are in the Lost Forest.
************
************
☺
************
************
Go left or right?
```

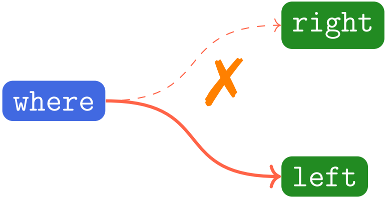


Program:

```
where = input("You’re in the Lost Forest. Go left or right? ")
while where == "right":
    where = input("You’re in the Lost Forest. Go left or right? ")
print("You got out of the Lost Forest!")
```

while loop Example

```
You are in the Lost Forest.
************
************
☺
************
************
Go left or right?
```



Program:

```
where = input("You're in the Lost Forest. Go left or right? ")
while where == "right":
    where = input("You're in the Lost Forest. Go left or right? ")
print("You got out of the Lost Forest!")
```

You Try!

What is printed when you type ‘RIGHT’?

(note the case)

```
where = input("Go left or right? ")
while where == "right":
    where = input("Go left or right? ")
print("You got out!")
```

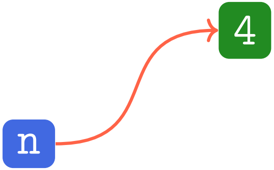
while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```

Shell

while loop Example

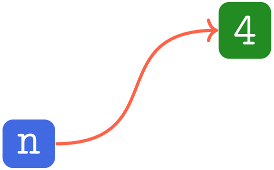
```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```



Shell

while loop Example

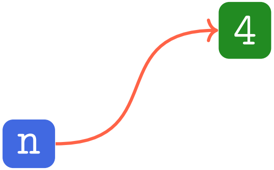
```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```



Shell

while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```

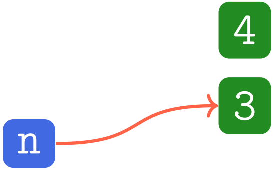


Shell

```
x
```

while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```

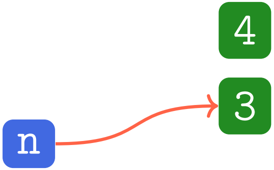


Shell

```
x
```

while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```

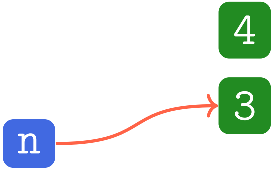


Shell

```
x
```

while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```

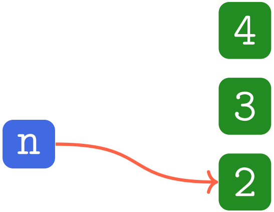


Shell

```
x
x
```

while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```

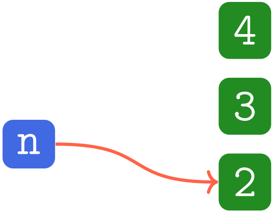


Shell

```
x
x
```

while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```

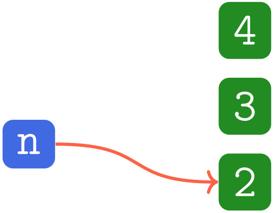


Shell

```
x
x
```

while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```

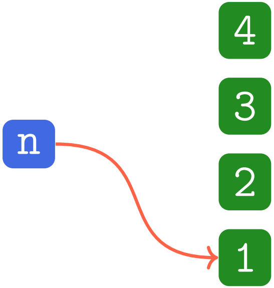


Shell

```
x
x
x
```

while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```

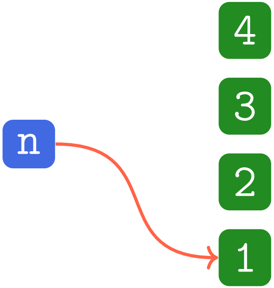


Shell

```
x
x
x
```

while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```



Shell

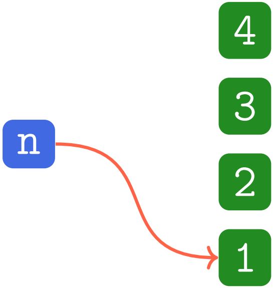
```
x
x
x
```

while loop Example

```python
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```

Shell

```
x
x
x
x
```

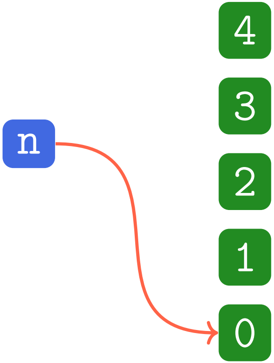


while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```

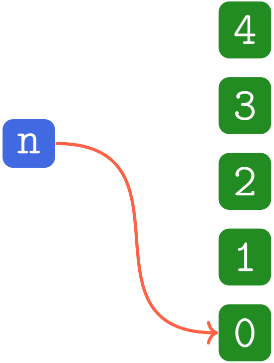
Shell

```
x
x
x
x
```



while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```

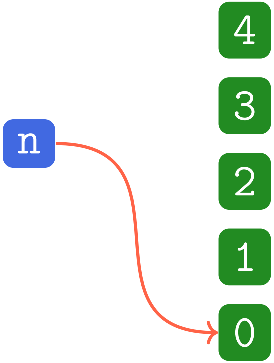


Shell

```
x
x
x
x
```

while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    n = n-1
```



Shell

```
x
x
x
x
```

while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print(’x’)
    ~~n = n-1~~
```

What happens without this last line? Try it!

while loop Example

```
n = int(input("Enter a non-negative integer: "))
while n > 0:
    print('x')
    ~~n = n-1~~
```

What happens without this last line? Try it!

To terminate:

- Hit `Ctrl + c`

You Try!

Run this code and stop the infinite loop in your IDE

```
while True:
    print("noooooo")
```

Big Idea

while loops can **repeat** code inside **indefinitely**!

Sometimes they need your intervention to end the program.

You Try!

Expand this code to show a sad face when the user entered the while loop more than 2 times.

Hint: use a variable as a counter

```
where = input("Go left or right? ")
while where == "right":
    where = input("Go left or right? ")
print("You got out!")
```

Iterate through **numbers in a sequence**

```
n = 0
while n < 5:
    print(n)
    n = n+1
```

Iterate through numbers in a sequence

```
n = 0
while n < 5:
    print(n)
    n = n+1
```

Set loop variable outside while loop

Iterate through numbers in a sequence

```
n = 0
while n < 5:
    print(n)
    n = n+1
```

Set loop variable outside while loop

Test loop variable in condition

Iterate through numbers in a sequence

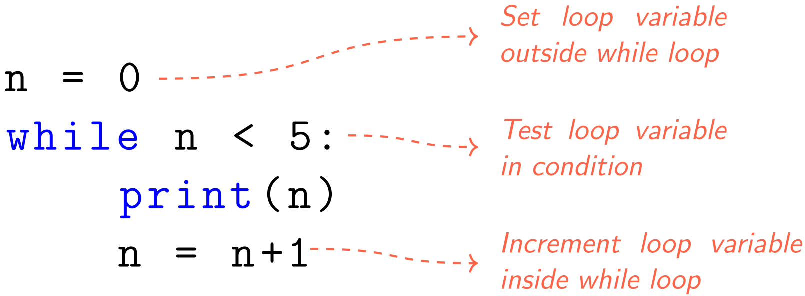
```
n = 0
while n < 5:
    print(n)
    n = n+1
```

Set loop variable outside while loop

Test loop variable in condition

Increment loop variable inside while loop

Iterate through **numbers in a sequence**



```
n = 0
while n < 5:
    print(n)
    n = n+1
```

Set loop variable outside while loop

Test loop variable in condition

Increment loop variable inside while loop

Almost all `while` loop programs have a similar structure

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

Initialize the sum to 0

```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```

Initialize the sum to 0

Set loop variable outside while loop

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```

Initialize the sum to 0

Set loop variable outside while loop

Test loop variable in condition

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```

Initialize the sum to 0

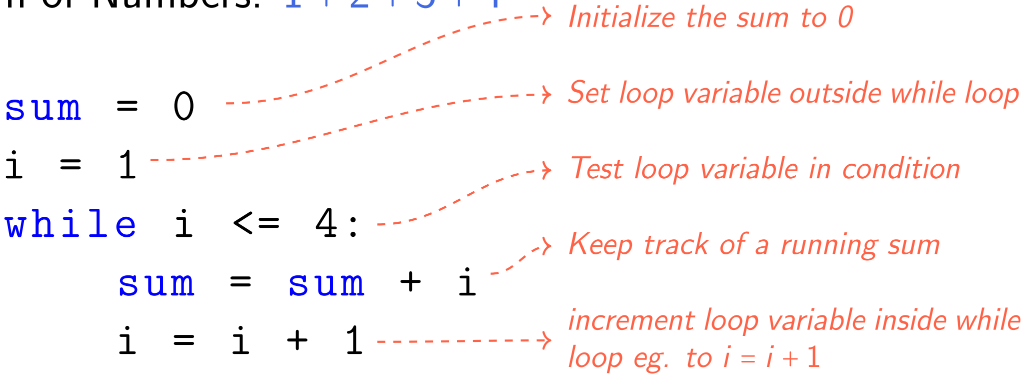
Set loop variable outside while loop

Test loop variable in condition

Keep track of a running sum

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$



```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```

Initialize the sum to 0

Set loop variable outside while loop

Test loop variable in condition

Keep track of a running sum

increment loop variable inside while loop eg. to $i = i + 1$

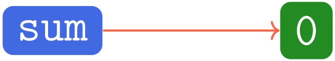
Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```

Iteration Examples

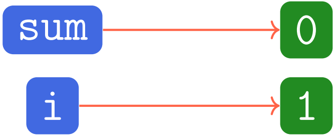
Sum of Numbers: $1 + 2 + 3 + 4$



```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```

Iteration Examples

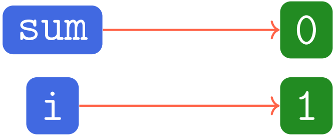
Sum of Numbers: $1 + 2 + 3 + 4$



```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

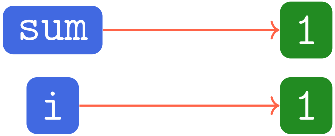


```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

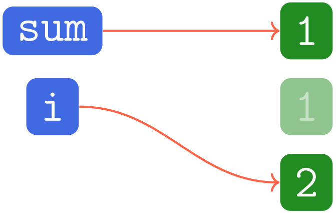
```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```



Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

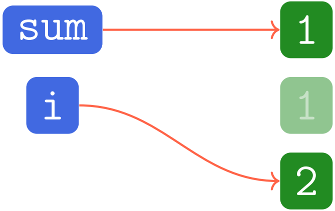
```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```



Iteration Examples

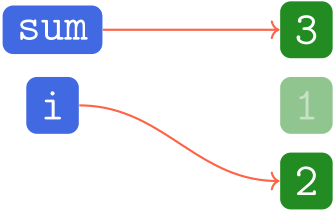
Sum of Numbers: $1 + 2 + 3 + 4$

```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```



Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

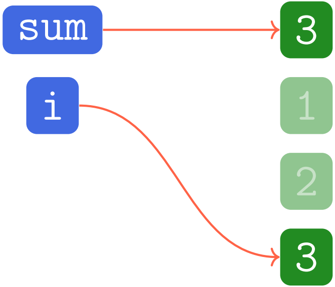


```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

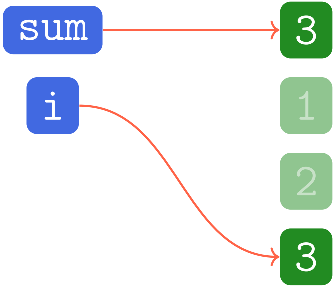
```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```



Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```



sum

3

i

1

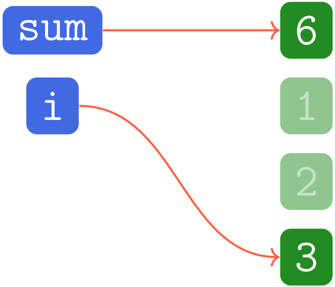
2

3

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```



sum

6

i

1

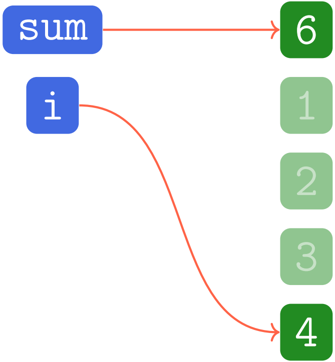
2

3

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

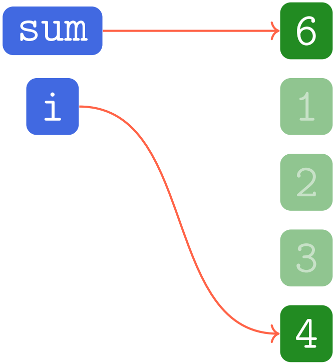
```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```



Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```



sum

6

i

1

2

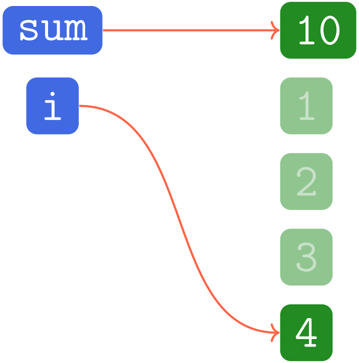
3

4

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```



sum

10

i

1

2

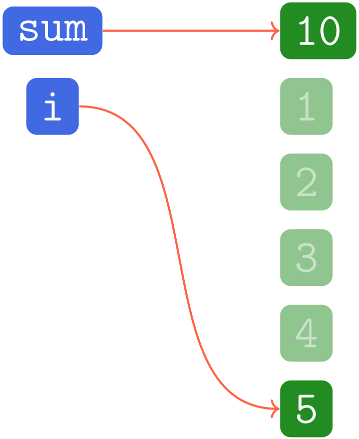
3

4

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```



sum

10

i

1

2

3

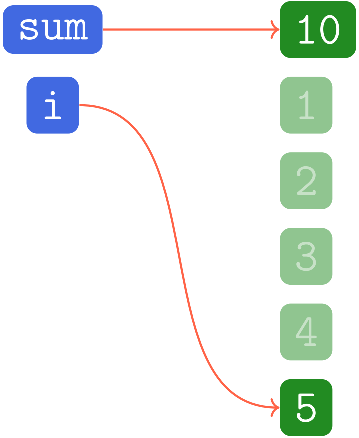
4

5

Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

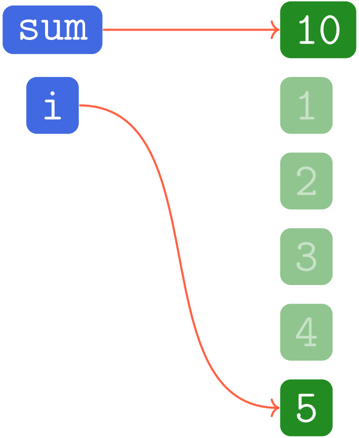
```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```



Iteration Examples

Sum of Numbers: $1 + 2 + 3 + 4$

```
sum = 0
i = 1
while i <= 4:
    sum = sum + i
    i = i + 1
```



Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

```
factorial = 1
i = 1
while i <= 4:
    factorial *=i
    i = i + 1
```

Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

Initialize the factorial to 1

```
factorial = 1
i = 1
while i <= 4:
    factorial *=i
    i = i + 1
```

Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

```
factorial = 1
i = 1
while i <= 4:
    factorial *=i
    i = i + 1
```

Initialize the factorial to 1

Set loop variable outside while loop

Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

```
factorial = 1
i = 1
while i <= 4:
    factorial *=i
    i = i + 1
```

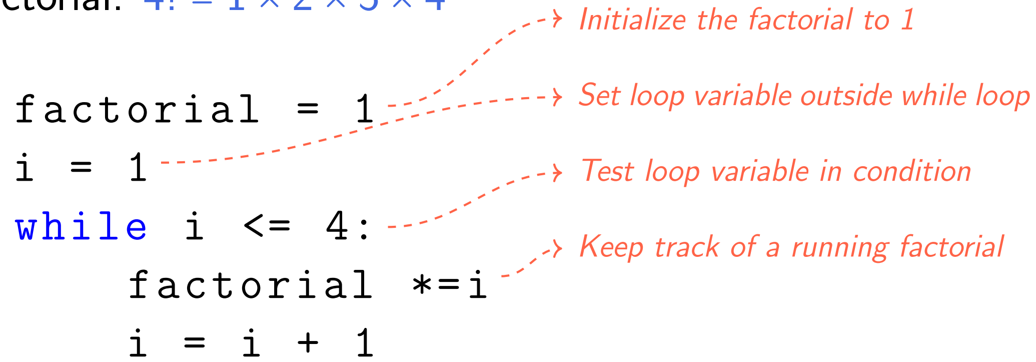
Initialize the factorial to 1

Set loop variable outside while loop

Test loop variable in condition

Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$



```
factorial = 1
i = 1
while i <= 4:
    factorial *=i
    i = i + 1
```

Initialize the factorial to 1

Set loop variable outside while loop

Test loop variable in condition

Keep track of a running factorial

Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

```
factorial = 1
i = 1
while i <= 4:
    factorial *=i
    i = i + 1
```

Initialize the factorial to 1

Set loop variable outside while loop

Test loop variable in condition

Keep track of a running factorial

increment loop variable inside while loop eg. to $i = i + 1$

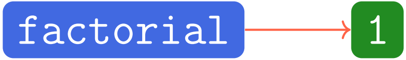
Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```

Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$



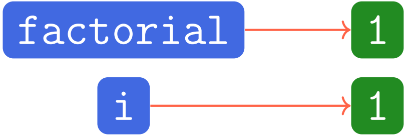
factorial

1

```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```

Iteration Examples

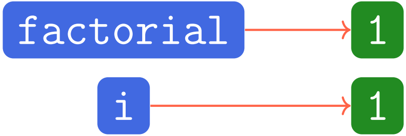
Factorial: $4! = 1 \times 2 \times 3 \times 4$



```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```

Iteration Examples

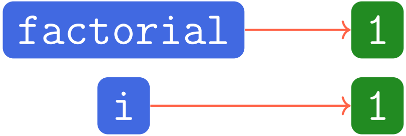
Factorial: $4! = 1 \times 2 \times 3 \times 4$



```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```

Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$



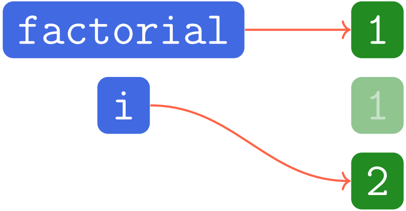
factorial → 1

i → 1

```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```

Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

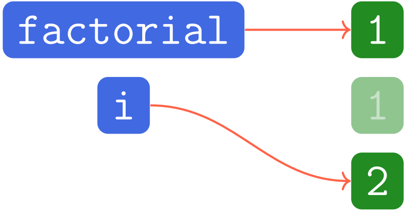


```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```

Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```

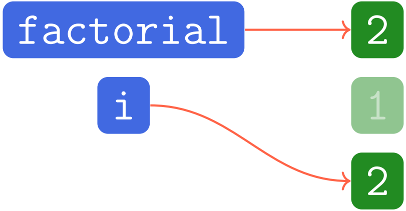


factorial 1

i 1 2

Iteration Examples

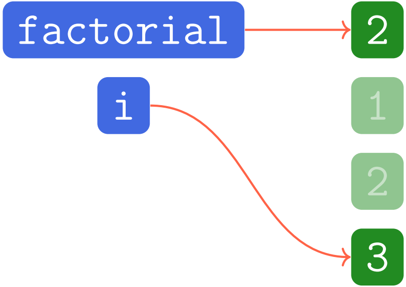
Factorial: $4! = 1 \times 2 \times 3 \times 4$



```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```

Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

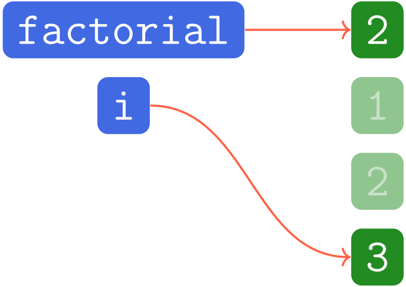


```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```

Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```

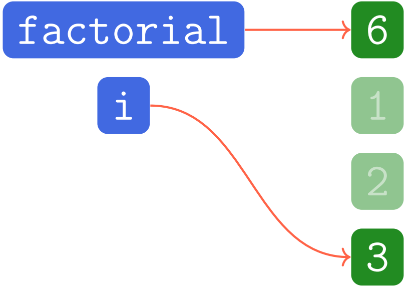


factorial
2
i
1
2
3

Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

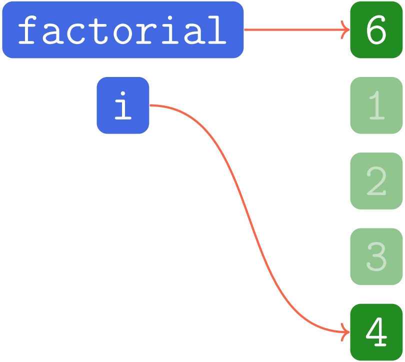
```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```



Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```



factorial

6

i

1

2

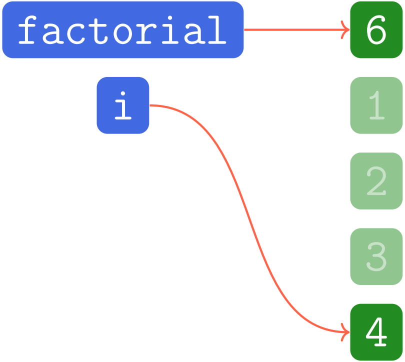
3

4

Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

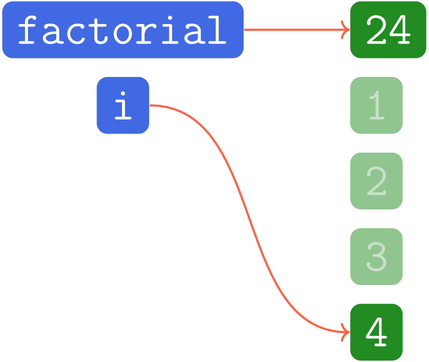
```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```



Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

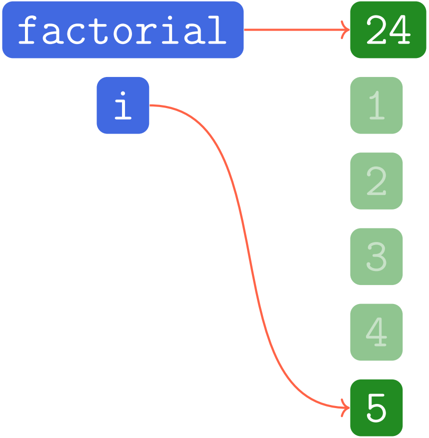
```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```



Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

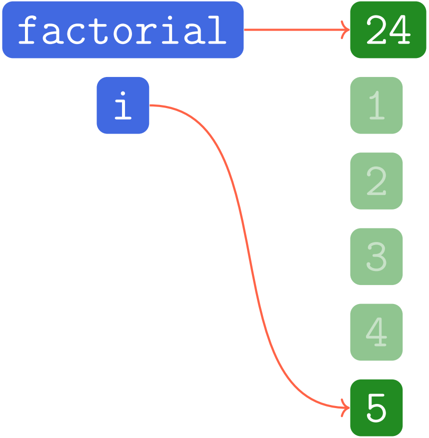
```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```



Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

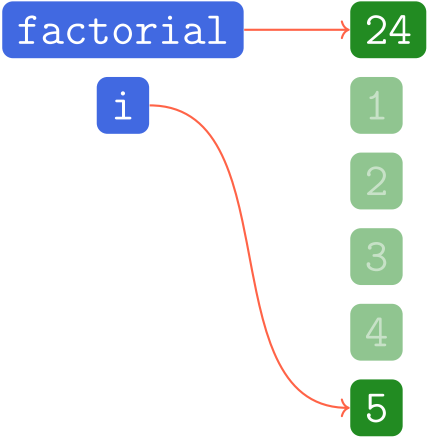
```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```



Iteration Examples

Factorial: $4! = 1 \times 2 \times 3 \times 4$

```
factorial = 1
i = 1
while i <= 4:
    factorial *= i
    i = i + 1
```



factorial

24

i

1

2

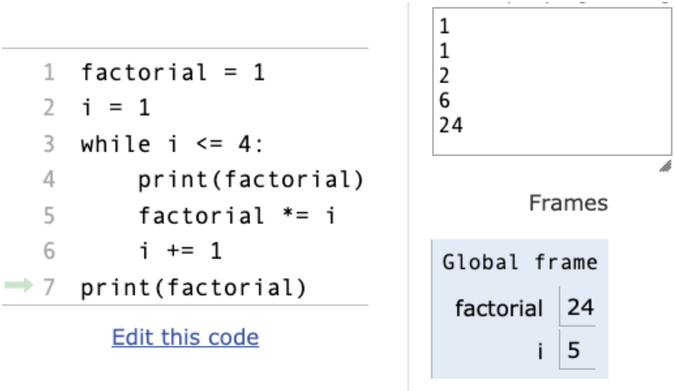
3

4

5

PythonTutor.com

Factorial: $4! = 1 \times 2 \times 3 \times 4$



```python
factorial = 1
i = 1
while i <= 4:
    print(factorial)
    factorial *= i
    i += 1
print(factorial)
```

```
1
1
2
6
24
```

factorial	24
i	5

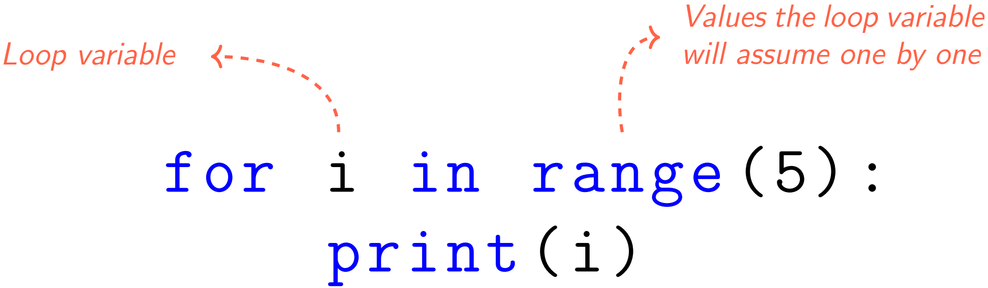
Useful Tip

Python Tutor can help you visualize how your code works

If you're stuck, try running your code there!

`for` Loops

for Loop Syntax

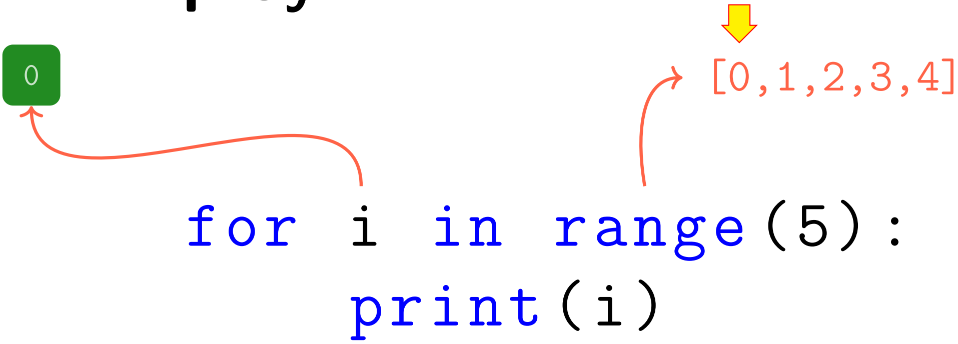


Loop variable

Values the loop variable will assume one by one

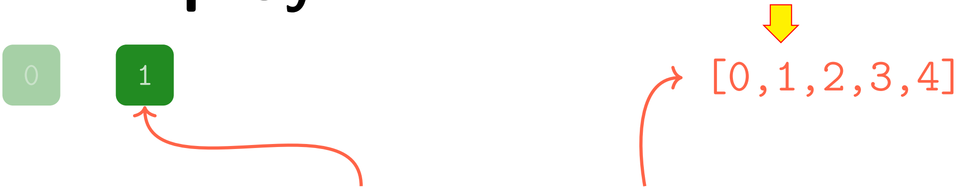
```
for i in range(5):
    print(i)
```

for Loop Syntax



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

for Loop Syntax

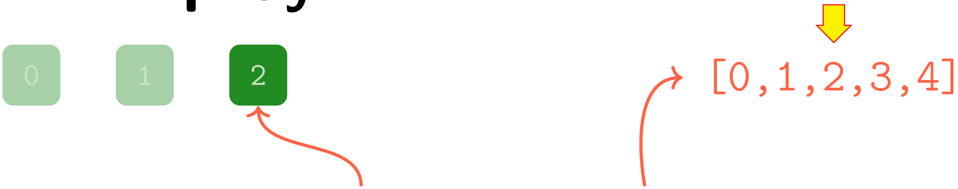


0 1

[0,1,2,3,4]

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

for Loop Syntax

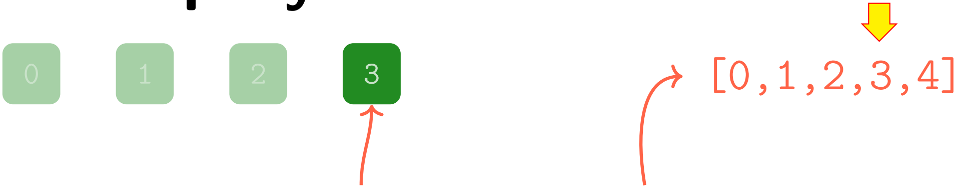


0 1 2

[0,1,2,3,4]

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

for Loop Syntax

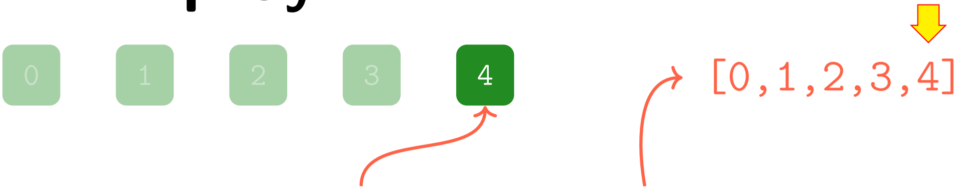


0 1 2 3

[0,1,2,3,4]

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

for Loop Syntax



0 1 2 3 4

[0,1,2,3,4]

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

range(start, stop, step) Function

- Generates a **sequence** of ints, following a pattern
- A lot like what we saw for **slicing strings**

range(start, stop, step) Function

- Generates a **sequence** of ints, following a pattern
- A lot like what we saw for **slicing strings**
- Often omit start and step, e.g:
    - `for i in range(4):`
        - ★ `start` defaults to 0
        - ★ `step` defaults to 1
    - `for i in range(3,5):`
        - ★ `step` defaults to 1

You Try!

What do these print?

1.
   ```
   for i in range(1,4,1):
       print(i)
   ```

2.
   ```
   for j in range(1,4,2):
       print(j*2)
   ```

3.
   ```
   for me in range(4,0,-1):
       print("$"*me)
   ```

Loop over Strings

What will this print?

```python
s = "hello"
for i in range(5):
    print(s[i])
```

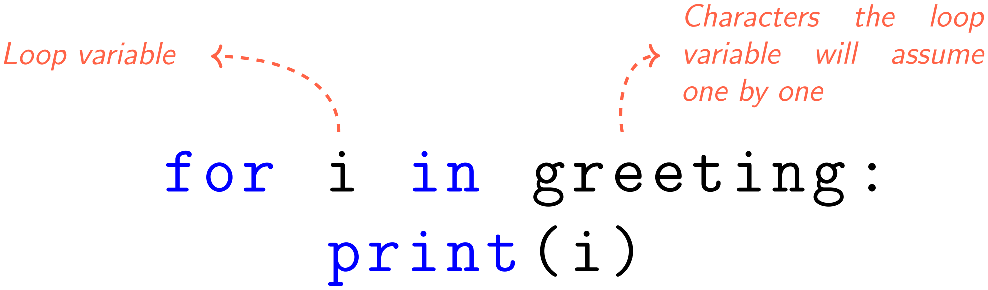
You Try!

Count the number of `‘a’` characters in any given string e.g. `‘casablanca’`.

for Loop on Strings

```
greeting = ’hello’
for i in greeting:
    print(i)
```

for Loop on Strings

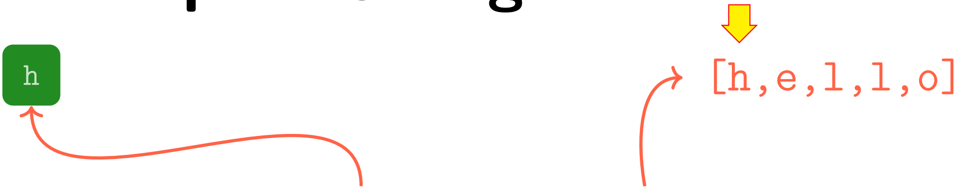


Loop variable

Characters the loop variable will assume one by one

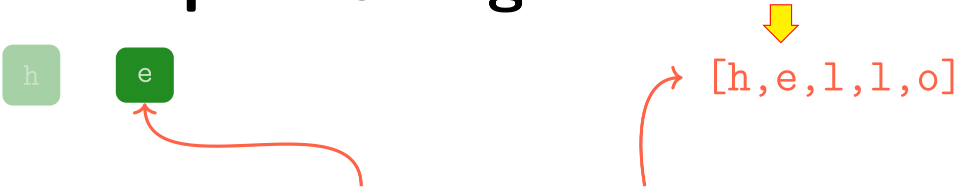
```
for i in greeting:
    print(i)
```

for Loop on Strings



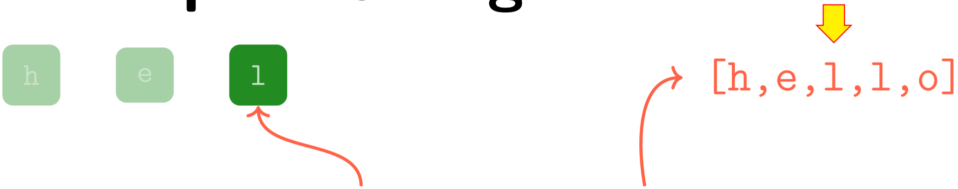
```
for i in greeting:
    print(i)
```

for Loop on Strings



```
for i in greeting:
    print(i)
```

for Loop on Strings

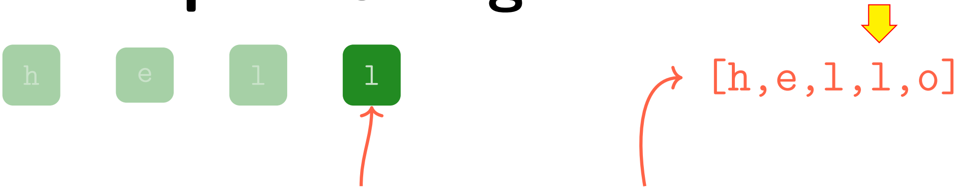


h e l

[h,e,l,l,o]

```
for i in greeting:
    print(i)
```

for Loop on Strings

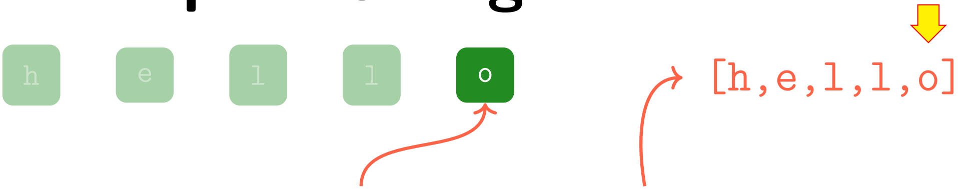


h e l l

[h,e,l,l,o]

```
for i in greeting:
    print(i)
```

for Loop on Strings



h e l l o

[h,e,l,l,o]

```
for i in greeting:
    print(i)
```

in has two meanings in Python.

- When used with for loops:
  - in can be used to iterate over a sequence.
- When used with strings:
  - in can be used to check if a character is in a string.
  - in can be used to check if one string is in another string.
- It can be used to check if a character is in a string.

The in Operator

a.k.a the **membership operator**.

Try the following code in Python shell:

- `‘a’ in ‘banana’`
- `‘cat’ in ‘caterpillar’`
- `‘q’ in ‘comp102’`
- `‘o’ in ‘aeiou’`

Loop over Strings

Code to check for letter i or u in a string. All 3 do the same thing:

```
s = "demo loops - fruit loops"
for index in range(len(s)):
    if s[index] == ’i’ or s[index] == ’u’:
        print("There is an i or u")
```

Loop over Strings

Code to check for letter i or u in a string. All 3 do the same thing:

```
s = "demo loops - fruit loops"
for index in range(len(s)):
    if s[index] == ’i’ or s[index] == ’u’:
        print("There is an i or u")
```

```
for char in s:
    if char == ’i’ or char == ’u’:
        print("There is an i or u")
```

Loop over Strings

Code to check for letter i or u in a string. All 3 do the same thing:

```
s = "demo loops - fruit loops"
for index in range(len(s)):
    if s[index] == ’i’ or s[index] == ’u’:
        print("There is an i or u")
```

```
for char in s:
    if char == ’i’ or char == ’u’:
        print("There is an i or u")
```

```
for char in s:
    if char in ’iu’:
        print("There is an i or u")
```

You Try!

- How many 2's in a string number? (895244254)
- Sum of all 2's in a string number? (895244254)
- How many vowels in a sentence?

You Try!

- Assume you are given a string of lowercase letters in variable s. Count how many unique letters there are in the string. For example if:

  `s = "abca"`

  Then your code prints 3.

Questions?