Loops and Algorithms

NSC 1002 Workshop 3

10 Nov 2020

Geoff Vallis

Loops

- Using computer programs to solve problems is great!
- The problems we encounter often involve repetitive tasks that are too complex or time-consuming to do by hand
- Loops are a **control structure** (like conditional **if** statements) that allow a program to repeat a set of instructions

For Loops

- A `for` loop iterates *for* a set number of steps (called an iterable).
- A for loop iterates over some kind of list: e.g.:

```
day_list=['Mon', 'Tues', 'Wednes', 'Thurs', 'Fri']
for d in day_list:
    print(f'Today is {d}day')
```

For Loops

- A `for` loop iterates *for* a set number of steps (called an iterable).
- A for loop iterates over some kind of list: e.g.:

```
day_list=['Mon', 'Tues', 'Wednes', 'Thurs', 'Fri']
for d in day_list:
    print(f'Today is {d}day')
```

```
Today is Monday
Today is Tuesday
Today is Wednesday
Today is Thursday
Today is Friday
```

For Loops

A numerical iterator can also be used:

```
N = 5
for i in range(N):
    print(i**2)
```

For Loops

A numerical iterator can also be used:

```
N = 5
for i in range(N):
    print(i**2)
```

```
0
1
4
9
16
```

NB: What does `range(N)` do? 0...N,

For Loops

A numerical iterator can also be used:

```
N = 5
for i in range(N):
    print(i**2)
```

```
0
1
4
9
16
```

NB: What does `range(N)` do? 0...N, 1...N,

For Loops

A numerical iterator can also be used:

```
N = 5
for i in range(N):
    print(i**2)
```

```
0
1
4
9
16
```

NB: What does `range(N)` do? 0...N, 1...N, 0...N-1?

For loops: moon.py

Let's start by placing the following into a new Python program, `moon.py`.

```
def main():
    """Compute weight on moon based on gaining 1 kg / year."""
    earth_weight = 52.0
    for year in range(0, 25):
        current_weight = earth_weight + 1
        moon_weight = earth_weight * 0.165
        print(f'In year {year} your weight is {moon_weigth}')
```

For loops: moon.py

Now we have what we set out to do:

```
def main():
    """Compute weight on moon based on gaining 1 kg / year."""
    earth_weight = 52.0
    for year in range(25):
        earth_weight += 1
        moon_weight = earth_weight * 0.165
        print(f'In year {year} your weight is {moon_weight}')
    pass
pass
more stuff...
```

While Loops

A `while` loop can be used when iteration is needed until a specific condition is met, no matter how many iterations it takes.

```
def main():
    """Create a list of squares."""
    list_of_sqrs=[ ]
    i=0
    while len(list_of_sqrs) < 5:
        list_of_sqrs.append(i**2)
        i += 1
        print(list_of_sqrs)
```

While Loops

A `while` loop can be used when iteration is needed until a specific condition is met, no matter how many iterations it takes.

```
def main():
    """Create a list of squares."""
    list_of_sqrs=[ ]
    i=0
    while len(list_of_sqrs) < 5:
        list_of_sqrs.append(i**2)
        i += 1
        print(list_of_sqrs)
```

```
[0, 1, 4, 9, 16]
```

Be careful with while loops as they might never stop!

Statement Nesting

Loops and conditionals can be "nested". This means multiple statements are grouped together.

```
def main():
    """Create a list of squares."""
    list_of_sqrs=[ ]
    i=0
    while len(list_of_sqrs) < 5:
        if i % 2 == 0:
            list_of_sqrs.append(i**2)
        i += 1
    print(list_of_sqrs)
```

Statement Nesting

Loops and conditionals can be "nested". This means multiple statements are grouped together.

```
def main():
    """Create a list of squares."""
    list_of_sqrs=[ ]
    i=0
    while len(list_of_sqrs) < 5:
        if i % 2 == 0:
            list_of_sqrs.append(i**2)
        i += 1
    print(list_of_sqrs)
```

```
[0, 4, 16, 36, 64]
```

Statement Nesting

```
def main():
    """Example of nested loops."""
    mult_list=[ ]
    N = 2
    M = 3
    for j in range(N):
        for i in range(M):
            mult_list.append(3 * i + 4 * j)
    print(mult_list)
```

Statement Nesting

```
def main():
    """Example of nested loops."""
    mult_list=[ ]
    N = 2
    M = 3
    for j in range(N):
        for i in range(M):
            mult_list.append(3 * i + 4 * j)
    print(mult_list)
```

```
[0, 3, 6, 4, 7, 10]
```

Statement Nesting

```
def main():
    """Example of nested loops."""
    mult_list=[ ]
    N = 2
    M = 3
    for j in range(N):
        for i in range(M):
            mult_list.append(3 * i + 4 * j)
    print(mult_list)
```

```
[0, 3, 6, 4, 7, 10]
```

```
3*0+4*0 = 0, 3*1+4*0 = 3, 3*2+4*0 = 6
3*0+4*1 = 4, 3*1+4*1 = 7, 3*2+4*1 = 10
```

Statement Nesting (Problems)

```
def main():
    """Example of how nested loops can go wrong."""
    list_of_sqrs = [ ]
    i = 0
    while len(list_of_sqrs) < 5:
        if i % 2 == 0:
            list_of_sqrs.append(i**2)
            i += 1
    print(list_of_sqrs)
```

Statement Nesting (Problems)

```
def main():
    """Example of how nested loops can go wrong."""
    list_of_sqrs = [ ]
    i = 0
    while len(list_of_sqrs) < 5:
        if i % 2 == 0:
            list_of_sqrs.append(i**2)
            i += 1
    print(list_of_sqrs)
```

This will loop infinitely, once `i == 1`, it gets no larger, neither does `list_of_sqrs`, thus `len(list_of_sqrs)` remains at `1`.

Statement Nesting (Problems)

```
def main():
    """Another example of how nested loops can go wrong."""
    N=5
    for i in range(N):
        if i % 2 == 0:
            print(i**2, 'is even')
            if i % 2 != 0:
                print(i**2, 'is odd')
```

Statement Nesting (Problems)

```
def main():
    """Another example of how nested loops can go wrong."""
    N=5
    for i in range(N):
        if i % 2 == 0:
            print(i**2, 'is even')
            if i % 2 != 0:
                print(i**2, 'is odd')
```

This is problematic...the statement `print(i**2, 'is odd')` will only be executed if `i` is both odd AND even!

Statement Nesting (Problems)

```
def main():
    """Example of correct nested loop."""
    N=5
    for i in range(N):
        if i % 2 == 0:
            print(i**2, 'is even')
        elif i % 2 != 0:
            print(i**2, 'is odd')
```

Statement Nesting (Problems)

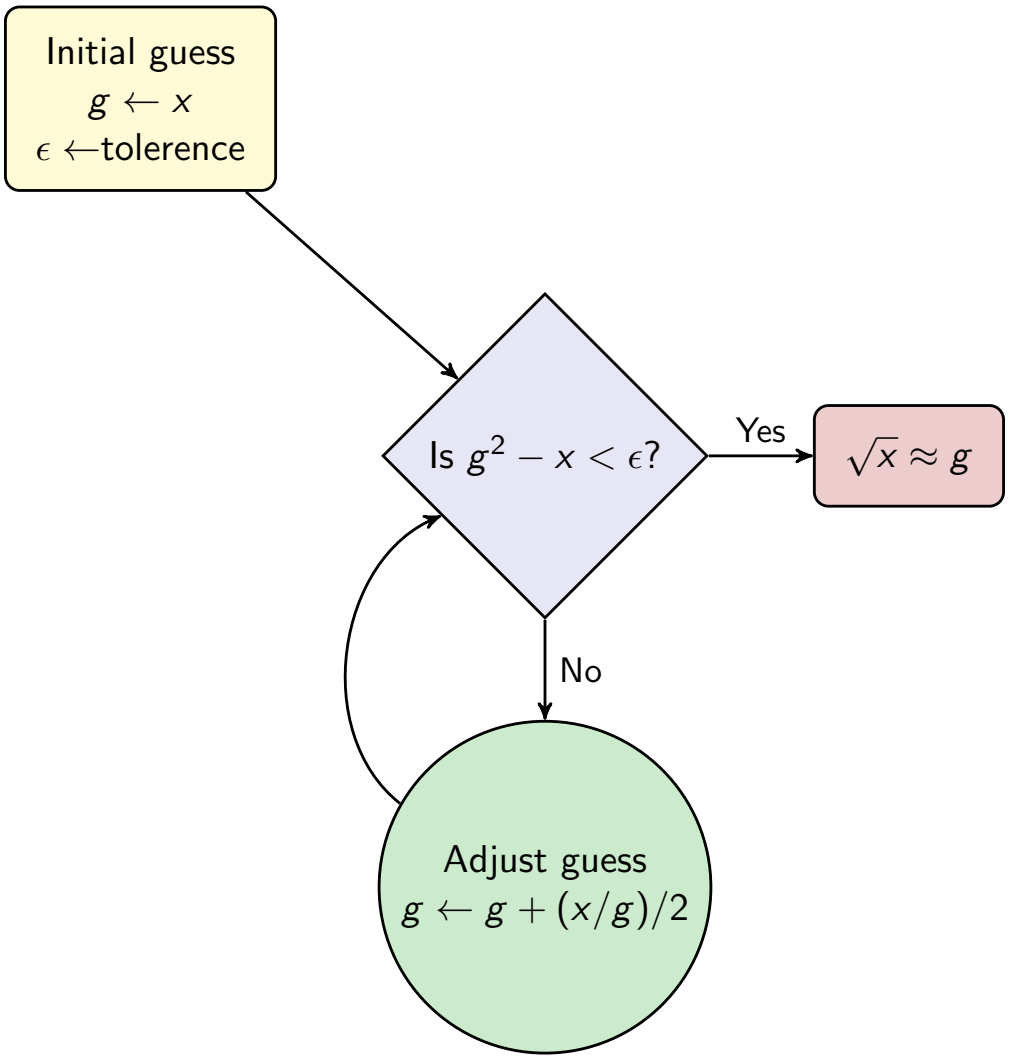
```
def main():
    """Example of correct nested loop."""
    N=5
    for i in range(N):
        if i % 2 == 0:
            print(i**2, 'is even')
        elif i % 2 != 0:
            print(i**2, 'is odd')
```

This is better...the statement `print(i**2, 'is odd')` will only be executed if `i` is odd.

Algorithms

- Algorithms are a set of instructions or rules for a computation
- For example: root finding, minimisation/maximisation, finding mean and variance, differential equation solving, ...
- Today you will be working with a root finding algorithm attributed to Heron of Alexandria, next time some other root finding algorithms
- These will all accomplish the same goal: the approximate square root of a number is found, but will vary in design, accuracy, and speed.

Heron of Alexandria's Algorithm



Questions?

Questions?

Now you have some time to work through worksheet examples