

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Looping with for

The INFDEV Team @ HR

Hogeschool Rotterdam
Rotterdam, Netherlands

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Introduction

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Lecture topics

- the (lack of) limitations of `while` loops
- `for` statements and their semantics
- `for` as a *limited* form of `while`

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

while loops

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Potential issues

- While loops specify unbounded iteration
- This means that the number of iterations is not necessarily easy to specify
- For example
 - Virtual machines
 - User-driven loops
 - Servers
 - Operating systems
 - ...

Unbounded loop example

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

```
n,m = input("Let's have two numbers")  
cnt = 1  
while n > m:  
    cnt = cnt + 1  
    n = n / m  
print("Result is %d" % cnt)
```

What does this code do?

How many steps does it take?

Unbounded loop example

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

```
quit = False
while not quit:
    action = raw_input("Should I quit?")
    if (action == "Yes") | (action == "yes"):
        quit = True
    else:
        print("You are not a quitter.")
```

What does this code do?

How many steps does it take?

Unbounded loop example

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

```
y = 10.0
vy = 0.0
dt = 0.05
while (abs(vy) > 0.9) | (y > 0.2):
    new_y = y + vy * dt
    if new_y <= 0.1:
        vy = -vy * 0.7
    else:
        vy -= 9.8 * dt
    y = new_y
    ... draw a ball at position (10,y) ...
```

What does this code do?

[google.com](https://www.google.com)

How many steps does it take?

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Potential issues

- while loops are very powerful
- with great power comes...

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Potential issues

- while loops are very powerful
- with great power comes...
- ...greater chance of bugs

Potential issues

- Subtle changes might affect behaviour deeply
- For example, a change in value of 0.1 makes the loop non-terminating
- The culprit may be hidden in a lot of places
 - Floating point errors
 - Logical repetition: state always changes, within a circular trajectory
 - ...

Unbounded loop example

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

```
y = 10.0
vy = 0.0
dt = 0.05
while (abs(vy) > 0.9) | (y > 0.1):
    new_y = y + vy * dt
    if new_y <= 0.1:
        vy = -vy * 0.7
    else:
        vy -= 9.8 * dt
        y = new_y
    ... draw a ball at position (10,y) ...
```

Does this loop terminate? (This is not the same code as in Slide 8!)

Unbounded loop example

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

```
y = 10.0
vy = 0.0
dt = 0.05
while (abs(vy) > 0.9) | (y > 0.1):
    new_y = y + vy * dt
    if new_y <= 0.1:
        vy = -vy * 0.7
    else:
        vy -= 9.8 * dt
    y = new_y
    ... draw a ball at position (10,y) ...
```

Does this loop terminate? (This is not the same code as in Slide 8!)

No. The condition has changed to $y > 0.1$.

Unbounded loop example

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

```
y = 10.0
vy = 0.0
dt = 0.1
while (abs(vy) > 0.9) | (y > 0.2):
    new_y = y + vy * dt
    if new_y <= 0.1:
        vy = -vy * 0.8
    else:
        vy -= 9.8 * dt
        y = new_y
    ... draw a ball at position (10,y) ...
```

Does this loop terminate? (This is not the same code as in Slide 8!)

Unbounded loop example

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

```
y = 10.0
vy = 0.0
dt = 0.1
while (abs(vy) > 0.9) | (y > 0.2):
    new_y = y + vy * dt
    if new_y <= 0.1:
        vy = -vy * 0.8
    else:
        vy -= 9.8 * dt
    y = new_y
    ... draw a ball at position (10,y) ...
```

Does this loop terminate? (This is not the same code as in Slide 8!)

No. $dt = 0.1$ and $vy = -vy * 0.8$.

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Correctly encoding intentions

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Why is while not enough

- The expressive power of `while` is not always needed
- Sometimes we want something simpler, and less dangerous
- For example, consider:
 - For each *hostile alien*
 - Do *attack it*

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Why is `while` not enough

- A loop such as:
 - For each *hostile alien*
 - Do *attack it*
- Is predictable
- Performs a fixed number of steps (one per hostile alien)
- Will certainly terminate

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Why is while not enough

- In general, we wish to always correctly encode our intention of repeating code N times
- The code must precisely fit our intentions, like a tailored italian suit
 - Code should not be too complicated
 - Code should not be too simple

Correctly encoding intentions

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Code that is too complicated?

- A while loop where we need to perform N steps
- There are many subtle ways to break the code

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Code that is too complicated?

- Classes, objects, and inheritance everywhere
- To know which code is actually run to say Hello world! you need to read twelve files

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Code that is too complicated?

- Events, lambda's, higher-order combinators everywhere
- To know what the program does you need two doctorates (CompSci and Maths)
 - Plus internal access to the sliced brain of the original programmer

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Code that is too simple?

- No handling of error cases
- Ignoring hard circumstances

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Code that is too simple?

- No handling of error cases
- Ignoring hard circumstances
- Not implementing all features correctly
 - Showing progress off
 - Building impressive but pointless demo's

Looping with
`for`

The INFDEV
Team @ HR

Introduction

`while` loops

Correctly
encoding
intentions

Iterating with
`for`

Advanced
loops

Conclusion

Iterating with `for`

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Code that is too simple?

- Python, and many other modern languages, offer explicit constructs for bounded repetition
 - We specify precisely the number of steps that need to be performed
 - The language takes care of performing the right number of steps
 - The construct is much harder to break^a than a `while`-loop
- These constructs are called for-loops

^aRunning forever

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Syntax of for

- Number of repetitions (a range iterator)
- That stores the index of the current repetition (a variable)
- Body of the loop that is repeated at every iteration (a block of code)

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

```
for VARIABLE in range(END):  
    BODY
```

- VARIABLE is any valid variable name that becomes useable within the BODY; will range from 0 to END-1
- END is any positive number; the body will be repeated END-1 times
- BODY is a series of statements

Semantics of for

- The general form is `for VAR in RANGE: BODY` (f_{VRB})
- If VAR is still within RANGE, then we jump to the beginning of BODY and then increment the variable, otherwise we jump to the end of the whole for

$$\begin{cases} (PC, S) \xrightarrow{f_{VRB}} (firstLine(B), S) & \text{when } S[V] \in R \\ (PC, S) \xrightarrow{f_{VRB}} (skipAfter(B), S) & \text{when } S[V] \notin R \end{cases}$$

- At the end of the loop assume that we have two invisible instructions
 - $V = V + 1$
 - jump back to begin loop

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Index of the current repetition

- The BODY of the for loop is always the same
- Depending on the current step, we may perform different processing
- For this, we need to know how far we have come in the loop
 - The iteration VARIABLE tells us this

Index of the current repetition

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

```
graph = ""
for i in range(11):
    for j in range((i-5) * (i-5)):
        graph += '='
    graph += "\n"
print(graph)
```

What does this do?

How do the different steps perform different actions based on the value of the iteration variable(s)?

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Index of the current repetition

- Different processing per different steps makes the loop perform a more complex operation.
- Complex is not the same as complicated.
- To avoid needless complication, the different steps must still do related things

Unrelated actions in a loop

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

```
for i in range(6):  
    if i == 0:  
        #...run a game of tic-tac-toe...  
    elif i == 1:  
        #...draw a smiley...  
    elif i == 2:  
        #...run a turtle program...  
    elif i == 3:  
        #...convert degrees to fahrenheit...  
    elif i == 4:  
        #...draw a square...  
    elif i == 5:  
        #...draw a triangle...
```

What is the relationship between the iterations?

Is a for loop really needed?

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Body of the loop

- The code of the body is a block of code
- A block of code is any statement or series of statements
- Among these statements, we can use as many if's, for's, and while's

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Body of the loop

- There is no *obfuscated code* prize available
- Nesting too many complex constructs might make code **needlessly complicated**
- Remember that a for-loop adds a large number of possible execution paths, just like a while-loop

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Advanced loops

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Ranges of iteration

- We do not always want to iterate through values between 0 and a given number
- Even though we still need to perform a fixed number of steps
- So a for-loop still has advantages over a while-loop

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

We might want to...

- ...decrement instead of increment, that is “go backwards”
- ...iterate between a range of values that does not start with zero
- ...take steps of more than one between iterations

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

The range function

- Actually takes three parameters: `range(start, end, step)`
- With one parameter we only specify end, while `start = 0` and `step = 1`
- With two parameters we specify start and end, while `step = 1`
- With all parameters we specify start, end, and step

Specific starting point

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

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


Multiple steps

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

```
for i in range(0, 20, 5):  
    print(i)
```

Backwards range

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

```
for i in range(10, 0, -1):  
    print("oooooooooooo\r" + str(i)),  
    sleep(0.3)  
print("\rBOOOM!!!!!!")
```

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Nesting for-loops

- The BODY of a for-loop contains arbitrary code
- This arbitrary code may also contain loops
- Loops within loops have a “multiplicative” behaviour
 - A loop of M step within a loop of N steps performs $N \cdot M$ steps

Multiplicative behaviour

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

```
cnt = 0
for i in range(10):
    for j in range(5):
        cnt += 1
print(cnt)
```

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Nesting for-loops

- Each loop adds its own iteration variable
- The iteration variables, together, are an N-dimensional point
- A single loop performs a “linear” computation, two loops perform a “square” computation, three perform a “cubic” computation, etc.

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Nesting for-loops

- Multiple for-loops perform a predetermined number of computations
- This means that we can always translate multiple for-loops into a single one^a

^aThis will usually break readability, so it is not advised: it is just a reasoning exercise.

```
for i in range(0,10):  
    for j in range(0,5):  
        print(i,j)
```

can be simulated with

```
for x in range(0,50):  
    i = x / 5  
    j = x % 5  
    print(i, j)
```

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Conclusion

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

Conclusion

- while-loops can encode any form of iteration.
- When the number of iterations is known beforehand, while is too powerful
- To use the right level of abstraction (which is less sensitive to bugs), we use for-loops instead
- This allows us to instruct the language to perform exactly the required number of steps, usually with less code

Looping with
for

The INFDEV
Team @ HR

Introduction

while loops

Correctly
encoding
intentions

Iterating with
for

Advanced
loops

Conclusion

The best of luck, and thanks for the
attention!