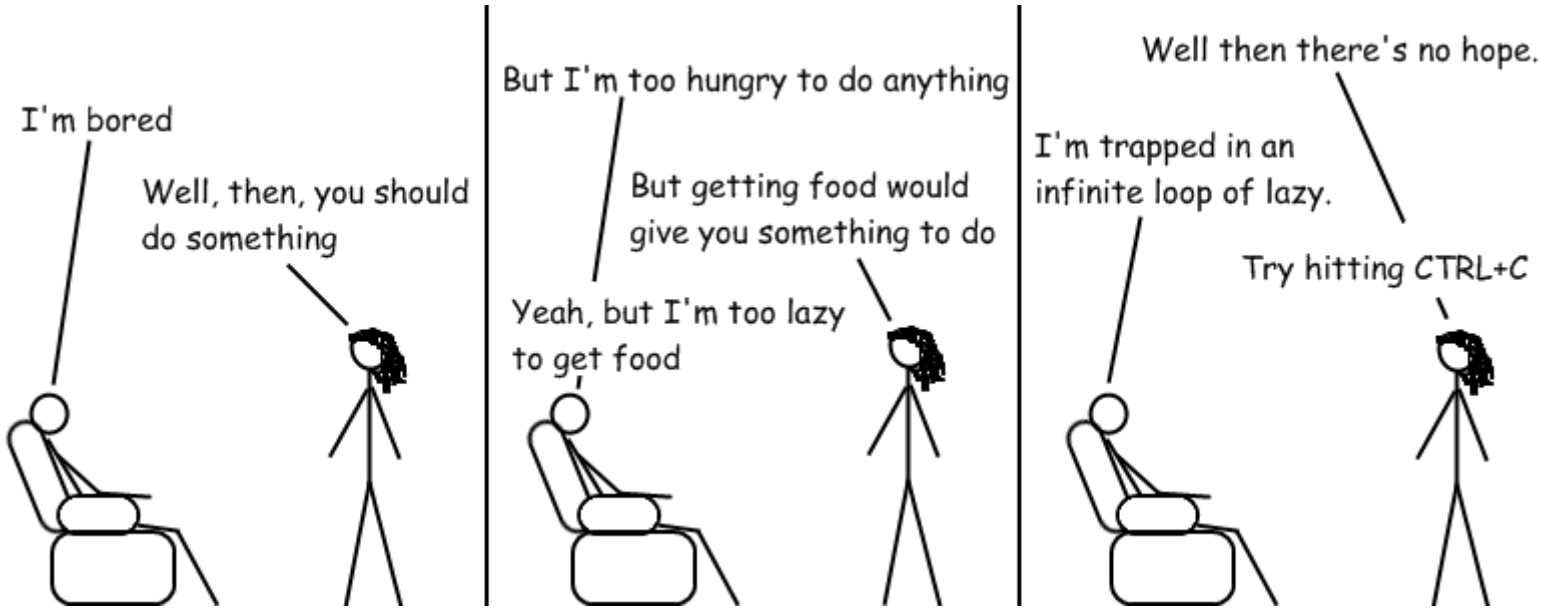
PRINT: Python bootcamp 2020

Statements: loops and conditionals

Johanna Hartke (jhartke@eso.org)



Nonglish.com

Resources: "Think Python: How to think like a computer scientist" by Allen Downey

First and foremost:

- Indentation is key!
- Makes code more readable
- In python: marks code blocks
- Good text editors take care of this
- Please, please, please do not mix spaces and tabs

The modulus operator %

- Works on integers
- Yields the remainder of division

Example

In [6]:
```
quotient = 5//3 # true division
remainder = 5%3
print('Quotient: ', quotient)
print('Remainder: ', remainder)
```

```
Quotient:  1
Remainder:  2
```

Boolean expressions

- Expressions that are either `True` or `False`
- `True` and `False` are not strings, they have type `bool`
- `None` is evaluated as `False`
- Nonzero numbers are evaluated as `True`

Examples:

In [7]:
```
10 == 10
```

Out[7]:
```
True
```

In [9]:
```
10 == 20
```

Out[9]:
```
False
```

In [11]:
```
type(False)
```

Out[11]:
```
bool
```

Comparison operators

Similar to other languages:

```
x != y        # x is not equal to y
x > y         # x is greater than y
x < y         # x is less than y
x >= y        # x is greater than or equal to y
x <= y        # x is less than or equal to y
```

Logical operators

1) and

In [14]:
```
n = 6
n > 0 and n < 10
```

Out[14]:
```
True
```

2) or

In [18]:
```
n%2 == 0 or n%3 == 0
```

Out[18]:
```
True
```

3) `not` negates a boolean expression

In [19]:
```
not (n > 10)
```

Out[19]:
```
True
```

Conditional execution

```
if statement == True:
    do something
```

Note the indentation.

Example:

In [4]:
```
x = 42
if x > 0:
    print('x is positive')
if x < 0:
    pass # do nothing
```

```
x is positive
```

Alternative execution

- two possibilities
- `if` followed by `else`

In [6]:

```
x = 42
if x%2 == 0:
    print('x is even')
else:
    print('x is odd')
```

```
x is even
```

Chained conditionals

- more than two possibilities
- start with `if`
- continue with `elif` (not `else if`)
- can end with `else`
- conditions are checked in order
- even if more than one condition is true, only the first rue branch executes

In [11]:

```
x = -42
if x > 0:
    print('x is positive')
elif x == 0:
    print('x is zero')
else:
    print('x is negative')
```

```
x is negative
```

Nested conditionals

- combine conditionals
- might become complicated to read
    - many tabs...

In [13]:

```
x = -42
if x > 0:
    print('x is positive')
else:
    if x == 0:
        print('x is zero')
    else:
        print('x is negative')
```

```
x is negative
```

Nested conditionals

- use logical operators to simplify nested condtionals

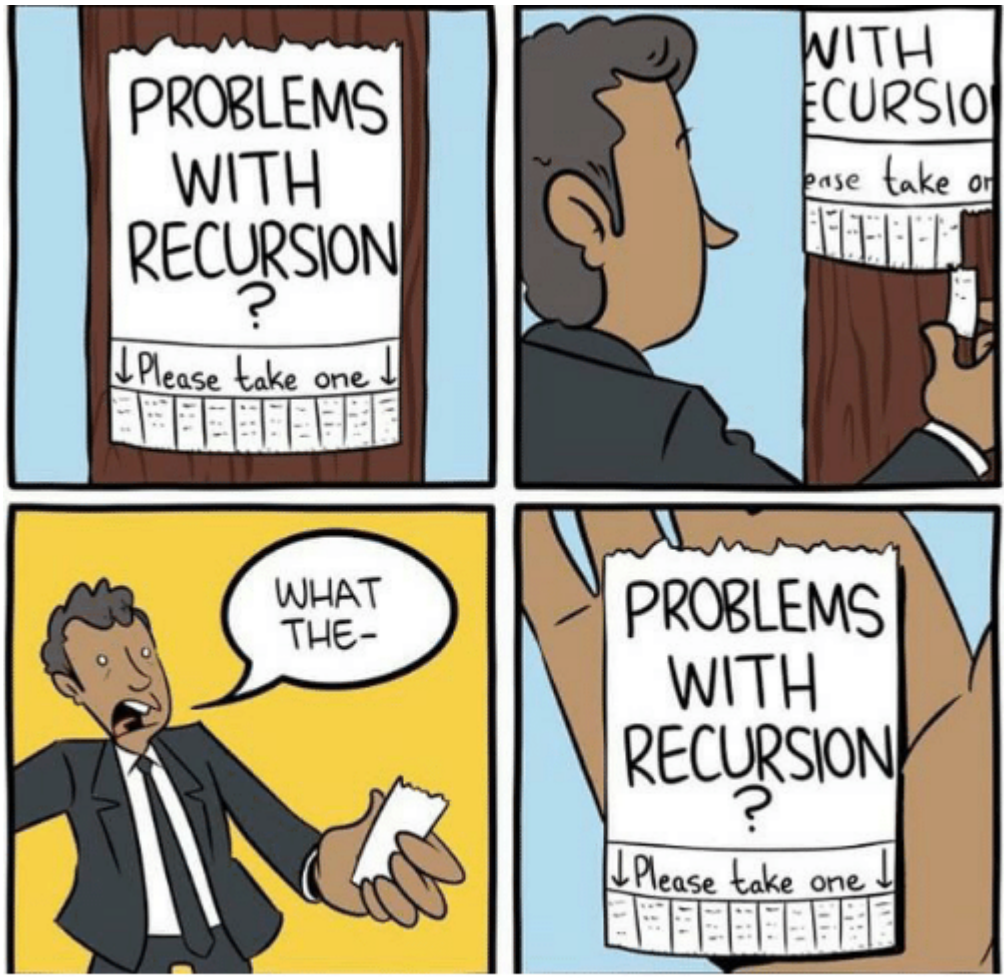
In [15]:
```
x = 42
if 10 < x:
    if x < 100:
        print('x is a positive double-digit number.')
```

```
x is a positive double-digit number.
```

In [16]:
```
if 10 < x and x < 100:
    print('x is a positive double-digit number.')
```

```
x is a positive double-digit number.
```

Recursion



Recursion time.

Recursion

- A function that calls itself is **recursive**.
- The process is called **recursion**.

Example: A function that prints a string n times.

In [18]:
```
def print_string(s, n):
    if n <= 0:
        return
    print(s)
    print_string(s, n-1)
```

In [19]:
```
word = 'test'
amount = 10
print_string(word, amount)
```

```
test
test
test
test
test
test
test
test
test
test
```

Infinite Recursion

- **Not a good idea**
- Python will prevent you from having an infinite recursion run forever

```
RuntimeError: Maximum recursion depth exceeded
```

Exercise 1: The final countdown

Write a recursive programme that prints a count-down from n. As soon as zero is reached, the program should print **Ka-Boom!**.

Exercise 2:

Write a programme that can compute factorials of a given integer using recursion:

$n! = n \cdot (n-1)!$

Iteration

> *"Repeating identicial or similar tasks without making errors is something that computers do well and people do poorly."*

Until now: used recursion to perform repetition.

While statements

```
while condition == True:
    do something (body)
```

Execution Flow

1. Evaluate the condition (`True` or `False`)
2. If `False`: exit the `while` statement and continue with next statement
3. If `True`: execute the body and go back to step 1.

The body of the loop

- Change the value of one or more variable
- Eventually, the condition has to become `False`
- Else: **Infinite Loop**

The iterative final countdown

In [33]:
```
def countdown(n):
    while n>0:
        print(n)
        n = n - 1
    print('Ka-Boom!')
```

In [35]:
```
countdown(10)
```

```
10
9
8
7
6
5
4
3
2
1
Ka-Boom!
```

Exercise 3:

Earlier, we wrote the recursive function `print_string` :

```
def print_string(s, n):
    if n <= 0:
        return
    print(s)
    print_string(s, n-1)
```

Rewrite the function using iteration instead of recursion.

`For` loops

Iterate over the items of any sequence

- Strings
- Lists
- ...

In [56]:
```
for letter in 'python':
    print('Current letter: ', letter)
```

```
Current letter:  p
Current letter:  y
Current letter:  t
Current letter:  h
Current letter:  o
Current letter:  n
```

In [57]:
```
fruits = ['banana', 'apple', 'mango']

for fruit in fruits:
    print('Current fruit: ', fruit)
```

```
Current fruit:  banana
Current fruit:  apple
Current fruit:  mango
```

In [69]:
```
# alternatively, as a while loop
index = 0
while index < len(fruits):
    fruit = fruits[index]
    print('Current fruit: ', fruit)
    index = index + 1
```

```
Current fruit:  banana
Current fruit:  apple
Current fruit:  mango
```

Looping and couting

In [62]:
```
word = 'banana'
count = 0
for letter in word:
    if letter == 'a':
        count += 1
print(count)
```

```
3
```

Exercise 4

Use the code snipped above, and rewrite it as a function named count. The function should accept the string, and the letter that is supposed to be counted.

I want to `break` free!

- `break` terminates the current loop
- operation is resumed at the next statement

In [36]:
```
for letter in 'Queen':
    if letter == 'e':
        break
    print('Current letter: ', letter)
```

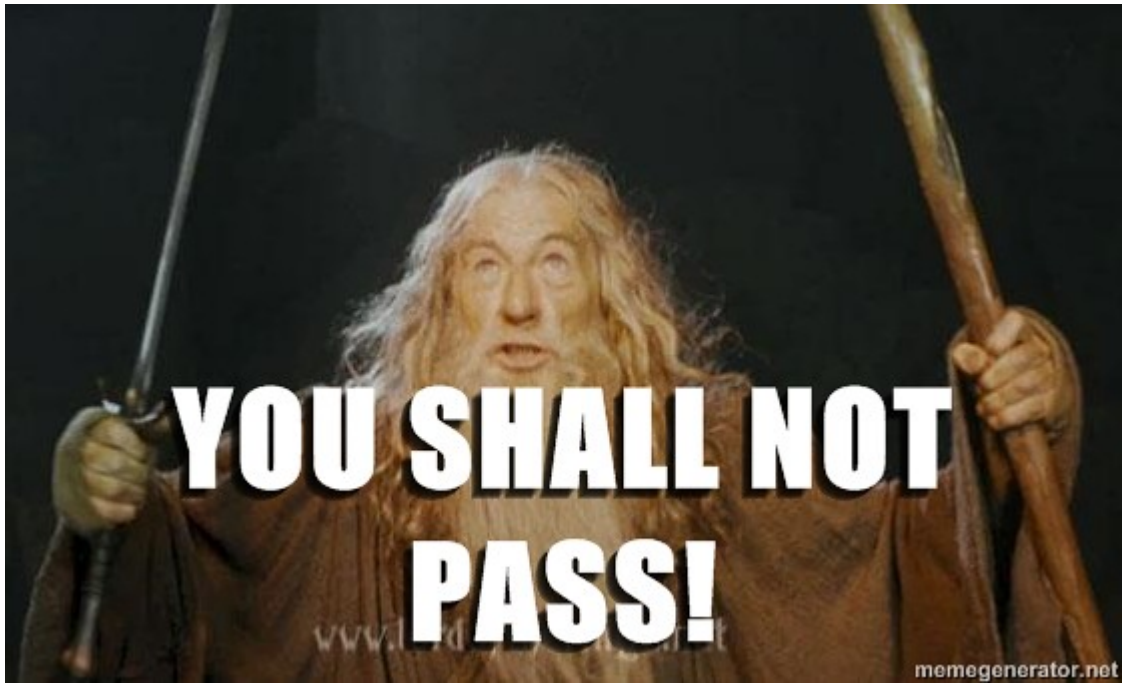
```
Current letter:  Q
Current letter:  u
```

In [40]:

```
number = 25
while number > 0:
    print('Current number:', number)
    number = number - 1
    if number == 15:
        break
```

```
Current number: 25
Current number: 24
Current number: 23
Current number: 22
Current number: 21
Current number: 20
Current number: 19
Current number: 18
Current number: 17
Current number: 16
```

You shall not `pass`



- *null operation*: do nothing
- useful as a placeholder when your code is still in development

In [44]:
```
for letter in 'Gandalf':
    if letter == 'a':
        pass
    else:
        print('Current letter: ', letter)
```

```
Current letter:  G
Current letter:  n
Current letter:  d
Current letter:  l
Current letter:  f
```

`Continue`

- After `continue`, return to the beginning of the loop
- Rejects all the remaining statements in the current iteration of the loop

In [70]:
```
for letter in 'Gandalf':
    if letter == 'a':
        continue
    print('Current letter: ', letter)
```

```
Current letter:  G
Current letter:  n
Current letter:  d
Current letter:  l
Current letter:  f
```

List comprehensions

Short way to create lists. Instead of

```
squares = []
for i in range(10):
    squares.append(i**2)
```

we can simply write

```
squares = [i**2 for i in range(10)]
```

Exercise 5

Write a programme that counts the number of characters for the Hogwarts houses in the following list of strings:

```
strings = ['Gryffindor', 'Ravenclaw', 'Hufflepuff', 'Slytherin']
```

If the number of characters is even, the programme should print the name of the house.

Final Exercise:

```
% phd.m
%
% author: Cecilia
% date: 09/08/05

load THESIS_TOPIC

while (funding==true)
    data = run_experiment(THESIS_TOPIC);
    GOOD_ENOUGH = query(advisor);
    if (data > GOOD_ENOUGH)
        graduate();
        break
    else
        THESIS_TOPIC = new();
        years_in_gradschool += 1;
    end
end
```

Rewrite Cecilia's code in python3 and hope it doesn't become obsolete before you finish your thesis.