Computing GC Content using functions and conditionals

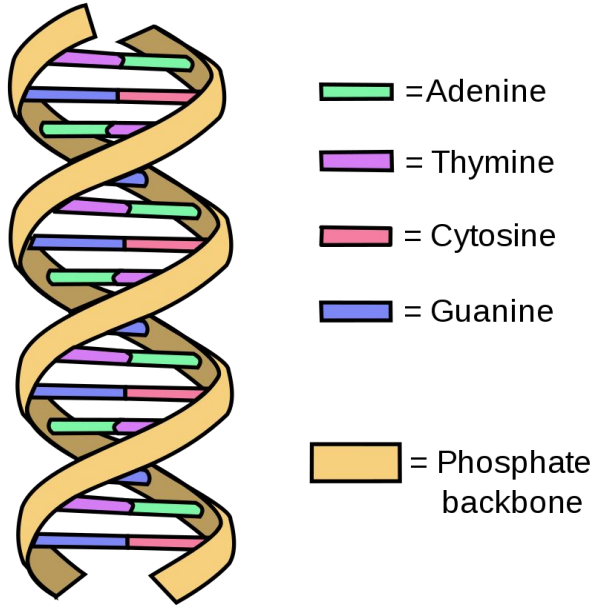
BILD 62

By the end of this lecture you will be able to:

- Recognize function syntax & write a simple function
- Recognize Booleans & write conditional logic statements
- Test conditional statements in Python

DNA Refresher

- Nucleic acids contain all of the information to build our cells!
- In deoxyribonucleic acid (DNA) there are four different: **adenine (A), cytosine (C), guanine (G), and thymine (T).**
- The sequence of a nucleic acid polymer is defined by the order of these bases, which we can represent with a string of A's, C's, G's, and T's.
- **Base pairs:** A bonds to T, and C bonds to G



DNA

Representing DNA on a computer

```
5′ - ATTCGTCA - 3′
3′ - TAAGCAGT - 5′
```

Forward strand

Reverse strand

same # of G or C, so we can work with either strand

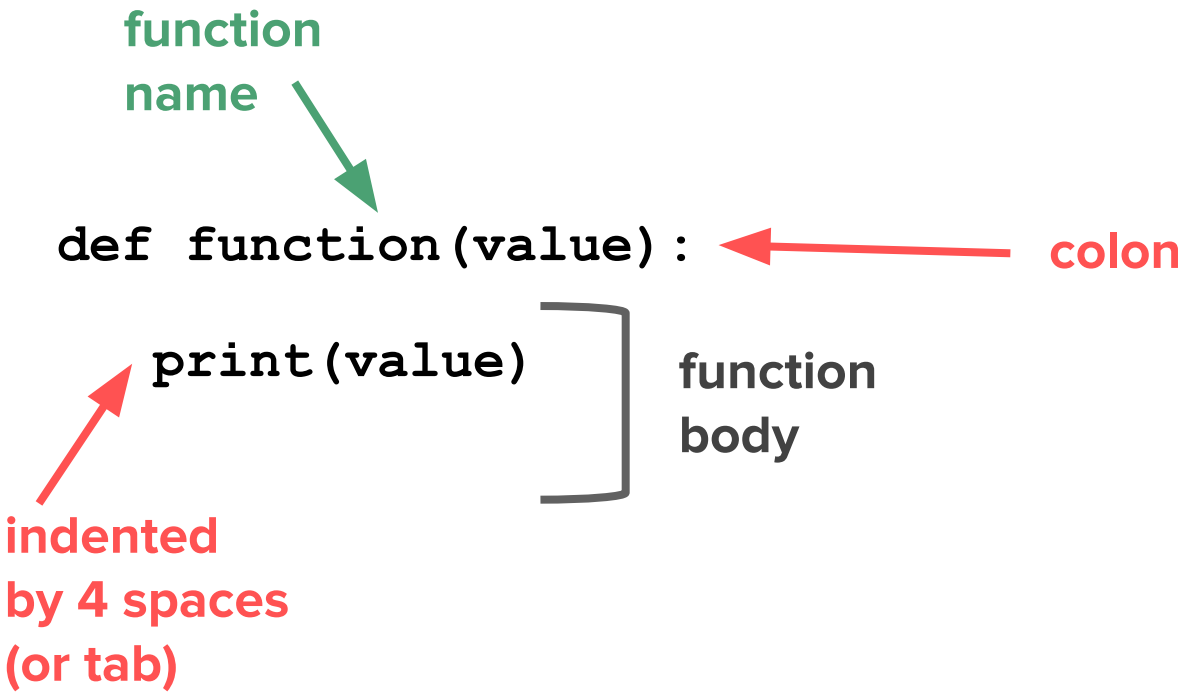
One way to characterize & distinguish different sequences of DNA is by their **GC content**. Can we write a **program** that does this?

Functions are pieces of code that are designed to do ***one task***

Functions take in inputs, process those inputs, and *possibly* return an output.

Python has *built-in* functions, but we can also write our own!

`function` syntax



`function` syntax

input arguments (these can be variables or default arguments)

```
def function(b):
    a = b**2
    return a

a = function(6)
```

`return` to retrieve a variable outside of a function *(what happens in the function stays in the function)*
ALSO ENDS THE FUNCTION!

call to function giving it the argument and saving the returned variable as a

`function` syntax

```
def function(b):
    c = b**2
    a = c * 2
    return a
a = function(6)
print(c)
????
```

`return` to retrieve a variable outside of a function ***(what happens in the function stays in the function)***

By the end of this lecture you will be able to:

- Recognize function syntax & write a simple function
- **Recognize Booleans & write conditional logic statements**
- **Test conditional statements in Python**

Operators in Python

Operators are special symbols that carry out arithmetic or logical computation.

Type of operator	**Examples**
assignment	`a = 6`
arithmetic (math)	`2 * 3`
logic (boolean)	`True and False`
comparison	`a != 6`
identity	`a is 6`
membership	`'a' in 'cat'`

Basic conditional operators in Python

Symbol	Operation	Usage	Outcome
==	Is equal to	`10==5*2`	`True`
!=	Is not equal to	`10 != 5*2`	`False`
>	Is greater than	`10 > 2`	`True`
<	Is less than	`10 < 2`	`False`
>=	Greater than *or* equal to	`10 >= 10`	`True`
<=	Less than *or* equal to	`10 <= 10`	`True`

Boolean variables store `True` (1) or `False` (0) and are the basis of all computer operations.

Syndey Padua:
https://sydneypadua.com/2dgoggles/happy-200th-birthday-george-boole/

`if` statements syntax

```
if condition:

    print('condition met')

    print('nice work.')

print('not in the block')
```

you need a colon here!

indented
by 4 spaces
(or tab)

block

`if/else` statement syntax

```
if condition:
        print('condition met')
        print('nice work.')
else:
        print('condition not met')
```

you need a colon here!

One more conditional: `elif`

- Short for “else if”
- Enables you to check for additional conditions ➜ *necessary if there is more than two outcomes*

```
condition_1 = False
condition_2 = True

if condition_1:
    print('Condition 1 is true.')
elif condition_2:
    print('Condition 2 is true.')
else:
    print('Both Condition 1 and 2 are false.')
```

Resources

Stepik Introduction to Python book, Chapter 2

Software Carpentries Conditionals

Whirlwind Tour of Python: Control Flow

Merely Useful Functions

Python Tutorial: Functions