

# Computing GC Content using functions and conditionals

---

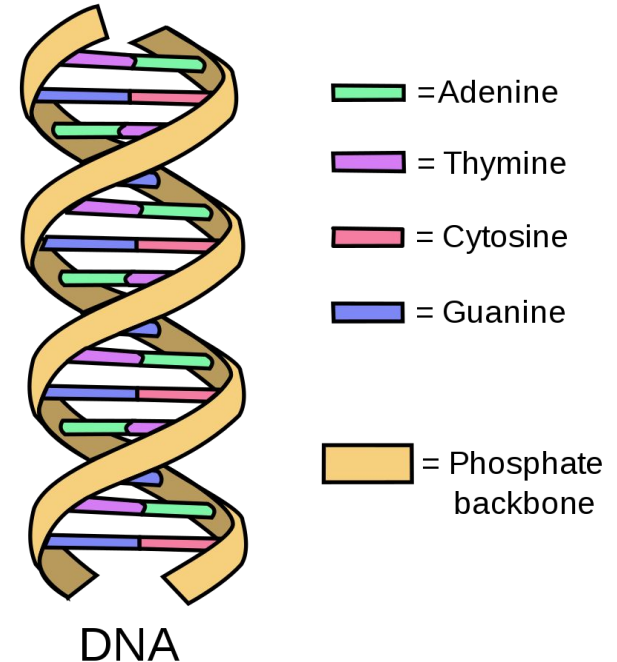
BILD 62

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



# Representing DNA on a computer

5' - ATTCGTCA - 3'

Forward strand

3' - TAAGCAGT - 5'

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  
name

```
def function(value):
```

colon

```
    print(value)
```

function  
body

indented  
by 4 spaces  
(or tab)

The diagram illustrates the syntax of a Python function definition. It shows the code `def function(value):` on the first line and  `print(value)` on the second line. A green arrow points from the text 'function name' to the word 'function' in the first line. A red arrow points from the text 'colon' to the colon at the end of the first line. A red arrow points from the text 'indented by 4 spaces (or tab)' to the indentation of the second line. A large right-facing curly bracket on the right side of the second line groups the indented code, with the text 'function body' next to it.

# function syntax

**input arguments** (these can be variables or default arguments)

```
def function(b):
```

```
    a = b**2
```

```
    return a
```

**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

```
a = function(6)
```

# 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	<code>a = 6</code>
arithmetic (math)	<code>2 * 3</code>
logic (boolean)	<code>True and False</code>
comparison	<code>a != 6</code>
identity	<code>a is 6</code>
membership	<code>'a' in 'cat'</code>

# Basic conditional operators in Python

Symbol	Operation	Usage	Outcome
<code>==</code>	Is equal to	<code>10==5*2</code>	True
<code>!=</code>	Is not equal to	<code>10 != 5*2</code>	False
<code>&gt;</code>	Is greater than	<code>10 &gt; 2</code>	True
<code>&lt;</code>	Is less than	<code>10 &lt; 2</code>	False
<code>&gt;=</code>	Greater than <i>or</i> equal to	<code>10 &gt;= 10</code>	True
<code>&lt;=</code>	Less than <i>or</i> equal to	<code>10 &lt;= 10</code>	True

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

Sydney Padua:

<https://sydneypadua.com/2dgoggles/happy-200th-birthday-george-boole/>





# if statements syntax

`if` condition:  you need a colon here!

indented  
by 4 spaces  
(or tab)

```
    print('condition met')  
    print('nice work.')  
print('not in the block')
```

 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

[Intro. to Comp. Sci. & OOP: Python · Cogniterra](#)

[Plotting and Programming in Python: For Loops](#)

[Plotting and Programming in Python: Conditionals](#)

[Whirlwind Tour of Python: Control Flow](#)

[Merely Useful Functions](#)

[Python Tutorial: Functions](#)