Python for scientists
Lesson 3
Data comparison and conditional statements

def complementary(seq):
    nt_comp = {
        'A': 'T',
        'T': 'A',
        'C': 'G',
        'G': 'C',
    }
    compseq = ''
    for nt in seq:
        compseq += nt_comp[nt]
    return compseq

Alvaro Sebastián Yagüe
www.sixthresearcher.com
@SixthResearcher
Data comparison: boolean expressions
Data comparison: boolean expressions

A boolean expression is a comparison between two data values and returns True or False. True and False values are from another data type: boolean type.

```python
>>> a = 'apples'
>>> b = 'pears'
>>> a == b
False
>>> c = 'apples'
>>> a == c
True
>>> b == c
False
>>> a = 123
>>> b = 456
>>> c = 579
>>> a == b
False
>>> a == c
False
>>> c == a + b
True
```

“True”

“False”
Data comparison: boolean expressions

A boolean expression is a comparison between two data values and returns True or False. True and False values are from another data type: boolean type.

```python
>>> easter_countries = ['Poland', 'Ukraine']
>>> western_countries = ['France', 'Spain']
>>> easter_countries == western_countries
False
>>> copy_easter_countries = easter_countries.copy()
>>> copy_easter_countries
['Poland', 'Ukraine']
>>> easter_countries == copy_easter_countries
True
>>> a = 1
>>> b = 2
>>> a = b  # Be careful '==' is not the same than '=='
>>> a == b
True
>>> a
2
```

‘==‘ and ‘==‘ are not the same thing
Data comparison: operators
The comparison operators provided by Python are:

```
>>> a = 524
>>> b = 525
>>> a == b
False
>>> a != b
True
>>> a < b
True
>>> a > b
False
>>> a <= b
True
>>> a >= b
False
>>> a == b-1
True
```

- `x == y`  \( x \) is equal than \( y \)
- `x != y`  \( x \) is different than \( y \)
- `x < y`  \( x \) is lower than \( y \)
- `x > y`  \( x \) is greater than \( y \)
- `x <= y`  \( x \) is lower or equal than \( y \)
- `x >= y`  \( x \) is greater or equal than \( y \)
The comparison operators provided by Python are:

```python
>>> a <= b-1
True
>>> a >= b-1
True
>>> c = 'potatoes'
>>> d = 'pears'
>>> a == c
False
>>> a != c
True
>>> c != d
True
>>> c = 'pears'
>>> c != d
False
>>> c == d
True
```

### Operators:

- `x == y`  
  - x is equal than y
- `x != y`  
  - x is different than y
- `x < y`  
  - x is lower than y
- `x > y`  
  - x is greater than y
- `x <= y`  
  - x is lower or equal than y
- `x >= y`  
  - x is greater or equal than y
Data comparison: logical operators

The logical operators allow to combine multiple comparisons (boolean expressions):

“and”  “or”  “not”

```python
>>> 1+1==2 and 2+2==4  # Both comparisons are true
    True
>>> 1+1==2 and 2+2==5  # One comparison is true and one false
    False
>>> 1+1==2 or 2+2==5   # One comparison is true and one false
    True
>>> 1+1==3 or 2+2==4   # One comparison is false and one true
    True
>>> 1+1==3 or 2+2==5   # Both comparisons are false
    False
>>> not 1+1==2         # 'not' reverses true to false
    False
>>> not 1+1==3         # or false to true
    True
>>> 1+1==2 and not 2+2==5 # One comparison is true and one false but reversed with 'not'
    True
>>> not 1+1==3 or not 2+2==5 # Both comparisons are false but...
    True
```
Data comparison: logical operators

The logical operators allow to combine multiple comparisons (boolean expressions):

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A and B == 1</th>
<th>A or B == 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data comparison: logical operators

The logical operators allow to combine multiple comparisons (boolean expressions):

```
  "and"            "or"

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>A and B == 1</th>
<th>A or B == 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```
Data comparison:
conditional statements
Data comparison: conditional statements

A **conditional statement** decides about the execution or not of a block of code based in some data comparison:

```python
>>> if 1+1==2:
    # If the data comparison is true prints the message
    print("Clever...")

Clever...

>>> if 1+2!=5:
    # If the data comparison is true prints the message
    print("Clever...")

Clever...

>>> if 1+1==5:
    # If the data comparison is false does nothing
    print("Clever...")
else:
    # But prints this other message
    print("Come back to school...")

Come back to school...
```
Data comparison: conditional statements

A **conditional statement** decides about the execution or not of a block of code based in some data comparison:

```python
>>> if 1+1==2:
    # If the data comparison is true prints the message
    print("Clever...")

Clever...
```

```python
>>> if 1+2!=5:
    # If the data comparison is true prints the message
    print("Clever...")

Clever...
```

```python
>>> if 1+1==5:
    # If the data comparison is false does nothing
    print("Clever...")

>>> if 1+1==5:
    # If the data comparison is false doesn't print this message
    print("Clever...")
else:
    # But prints this other message
    print("Come back to school...")

Come back to school...
```
Data comparison: conditional statements

```python
>>> temperature=10
>>> if temperature<0:
    print("It's frozen")
elif temperature>=0 and temperature<20:
    print("It's cold")
elif temperature>=20 and temperature<25:
    print("It's warm")
else:
    print("It's hot")

It's cold

>>> city='Poznan'
>>> if city=='Madrid' or city=='Barcelona':
    print(city,"is in Spain")
elif city=='London' or city=='Manchester':
    print(city,"is in UK")
elif city=='Warsaw' or city=='Poznan':
    print(city,"is in Poland")
else:
    print(city,"is elsewhere")

Poznan is in Poland
```
Data comparison: looking for list values

```python
>>> polish_cities=['Warsaw','Krakow','Poznan','Wroclaw','Lodz','Lublin']
>>> spanish_cities=['Madrid','Barcelona','Zaragoza','Sevilla','Valencia']
>>> british_cities=['London','Manchester','Liverpool','Glasgow','Edinburg']
>>> city='Zaragoza'
>>> if city in polish_cities:
    print(city,"is in Poland")
elif city in spanish_cities:
    print(city,"is in Spain")
elif city in british_cities:
    print(city,"is in UK")
else:
    print(city,"is unknown")

Zaragoza is in Spain

>>> world_cities={'Poland':polish_cities,'Spain':spanish_cities,'UK':british_cities}
>>> world_cities
>>> city='Zielona Gora'
>>> if city in world_cities['Poland']:
    print(city,"is in Poland")
elif city in world_cities['Spain']:
    print(city,"is in Spain")
elif city in world_cities['UK']:
    print(city,"is in UK")
else:
    print(city,"is unknown")

Zielona Gora is unknown
```
Data comparison: nested conditionals

```python
>>> spanish_cities=['Madrid', 'Barcelona', 'Zaragoza', 'Sevilla', 'Valencia']
>>> italian_cities=['Rome', 'Milan', 'Firenze', 'Venice', 'Napoly']
>>> polish_cities=['Warsaw', 'Krakow', 'Poznan', 'Wroclaw', 'Lodz', 'Lublin']
>>> german_cities=['Berlin', 'Hamburg', 'Munich', 'Cologne', 'Frankfurt']
>>> city='Poznan'
>>> temperature=17
>>> if (city in spanish_cities or city in italian_cities) and temperature>=30) or ((city in polish_cities or city in german_cities) and temperature>=25):
    print("It's hot")
 elif ((city in spanish_cities or city in italian_cities) and (temperature<30 and temperature>=20)) or ((city in polish_cities or city in german_cities) and (temperature<25 and temperature>=15)):
    print("It's warm")
 else:
    print("It's cold")

It's warm
>>> city='Zaragoza'
>>> if city in spanish_cities or city in italian_cities:
    if temperature>=30:
        print("It's hot")
    elif temperature<30 and temperature>=20:
        print("It's warm")
    else:
        print("It's cold")
 elif city in polish_cities or city in german_cities:
    if temperature>=25:
        print("It's hot")
    elif temperature<25 and temperature>=15:
        print("It's warm")
    else:
        print("It's cold")

It's cold

This is a big mess

Using nested conditionals we can improve code readability
Exercise:
What’s the time?
Exercise: What’s the time?

Write a Python code to tell the hour in English or Polish formal way.
Exercise: What’s the time?

A little help...

```python
>>> hour = 10
>>> minutes = 40
>>> if minutes>30 and minutes<60:
    minutes2=60-minutes
    hour2=hour+1
    print("It's",minutes2,"minutes to",hour2)
```

It's 20 minutes to 11

else if...

else if...

else ...
Python for scientists

Next lesson...

User data input and ‘while’ loops

def complementary(seq):
    nt_comp = {
        'A': 'T',
        'C': 'G',
        'G': 'C',
        'T': 'A',
    }
    compseq = ''
    for nt in seq:
        compseq += nt_comp[nt]
    return compseq

Alvaro Sebastián Yagüe

www.sixthresearcher.com

@SixthResearcher