Cheatography

Python Basics Cheat Sheet

by MacaSalva via cheatography.com/93894/cs/20707/

Comments in python

# For a single line comment.

`'''` For a multiple line comment

`"""`

`"""`
For a multiple line comment

Useful commands

`"\n"`	Jumps to a different line.
`"\t"`	Gives some spaces.
`#%%`	Creates a chunk of code.

Python Variables

`string`	Holds text based values
`int`	Interger numbers
`float`	Decimal numbers
`boolean`	True or False

There are also list, tuples and dictionaries.

To see what are the types of the variables we are working with: `type(_)`
Guidelines:

- Use descriptive names.
- Be consistent.
- Keep length in check.

You can't use spaces not dots (.) when defining names

Values

`input("__")`	Returns a string
`type( )`	Checks the type
`isinstance(x, float)`	Checks what you ask
Converting values	
`float(x)`	Converts x to float
`int(x)`	Converts x to interger
`str(x)`	Converts x to string

Values (cont)

`bool(x)`	Converts x to boolean
Creating values	
`range (start, stop[, step])`	You can create a range or usea a given

Mathematical operations

`+`	Addition
`-`	Substraction
`*`	Multiplication
`/`	Division
`%`	Modulus
`**`	Exponent
`//`	Floor division

There also exist a library called `math` *to make other kind of mathematical operations*

We can use this with numbers and booleans

String methods

`len(s)`	Calculate the lenght
`+`	Add two strings
`*`	Repeat a string
`s.find(x)`	Finds the first position of x in the string
`s.count(x)`	Counts the number of times x is in the string
`s.upper()`	All in upper case
`s.lower()`	All in lower case
`s.title()`	First letter upper case
`s.replace(x,y)`	Replace x by y

String methods (cont)

`s.strip()`	Strips spaces at the end
`s.split()`	Splits in whatever you say

Conditions

`"__" in [ ]` Is something in.	`"__" not in [ ]` Is something not in.
> Larger than	< Lower than
>= Larger or equal than	< = Lower or equal than
== Equal to something	!= Not equal to something
`and`	`or`
`not`	

Functions

```
def function_name ( arg1,arg2):
command1
command2
command3
return variable
```

You just have to use the name of the function with the number of variables that you have arrange.

```
def function_name ( *args):
command1
command2
command3
return variable
```

When you don't know the number of arguments that you want to use, you just use the * with the args.

```
def function_name ( **kwargs):
command1
command2
command3
return variable
```

With dictionaries

Lambda functions

It returns the value with a single expression, the can not obtain commands or more than one expression

Python Basics Cheat Sheet

by MacaSalva via cheatography.com/93894/cs/20707/

Functions (cont)

```
g=lambda x:x+2
```

Map functions

```
map()
```

Takes a function and an iterable like a list as arguments and applies the function to each item in the iterable.

When using functions you can use the command you want for the result, but is better using return instead of print because you can assign the result to a variable

Booleans

Displayed as:

`True`	`False`

Different than R

Conditionals

```
if __a __ :
       do this
elif __:
       do this
else:
       do this
```

The form of arrange is important, while you are under a condition you have to write inside it

While Loops:

```
while condition:
     statement1
     statement2
     statement3
```

In this kind of loops, while the condition is still true, it continues happening.

Helping in While Loops:

`continue`	Skips the current iteration and go to the next one
`break`	Halts the execution of a loop entirely

Both of them are useful to avoid infinite loops

For Loops

```
for i in list/range:
     statement1
     statement2
     statement3
```

A for loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly.

Library random

`randi-nt(a, b)`	creates values from the range you ask

For using the library you have to import it:

```
import random
```

Then, for using it, you need to say that you are going to look for the function in the library. If you want to avoid it you can import just one function from the library.

Helping in For Loops:

`continue`	Skips the current iteration and go to the next one
`break`	Halts the execution of the loop entirely

These commands continue and break are useful to avoid infinite loops