Python Cheat Sheet

by PaolaP (pao361) via cheatography.com/140352/cs/29763/

Rules

Python relies on proper indentation.
For example:

```
age = 18
if age >=18:
  print("Be sure to vote")
else:
  print("Sorry, too young")
```

Naming Rules

A variable name: MUST begin with a letter or underscore(_)

CANNOT contain spaces, punctuation or special characters others than the underscore

CANNOT begin with a number

CANNOT be the same as a reserved keyword in Python such as print, True, else, etc

A variable name is case sensitive

built-in functions

print()	this outputs something to the screen
input()	ask for input from the program user
str()	converts a variable to a string data type
int()	convert a variable to an int data type
float()	convert a variable to a float(decimal) data type
round()	rounds a number

Comparison Operators

==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Basic Math Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	division remainder
**	Exponent

Data Types

str	string(characteres typically words, sentences)
int	integer(0,5,133)
float	decimal number(1.23,623.664)
list	a collection of variables (mango, banana, oranges)
bool	boolean value (True, False)

Special Characters

\n	new line
\t	tab

LOCAL/GLOBAL Variables

LOCAL	Variable created within a function and only can be used by the function that defines them

LOCAL/GLOBAL Variables (cont)

GLOBAL	Variable defined outside of a function and can be accessed by any function without passing them to the function. Read-only and cannot be modified

Boolean Operators

not x	x **and** y	x **or** y

try and except

try:
  code statements
except: *#for all exceptions*
  code statements

try:
  code statements
except ValueError: *#Specific error type*
  code statements

Concatenate using "+" or "f"

combining strings
myName = "Paola"
print("Hello " + myName)
print(f"Hello {myName}")
string and a numeric value
age= 22
print("Your age: " + age)
print(f"Your age: {age}")

Capital and lowercase letters

```
hello = "hello world"
print(hello.upper())
  # will print HELLO WORLD
print(hello.lower())
  # will print hello world
print(hello.capitalize())
```

Capital and lowercase letters (cont)

```
  # will print Hello world
```

Control loops

break	breaks out of your loop causing the program to move to the next line after the loop
continue	while skip this round of the loop and go into the next loop iteration

Statements

If Statement
if *expression*:
 statements
elif *expression*:
 statements
else:
 statements
While Loop
while *expression*:
 statements
For Loop
for *var* in *collection*:
 statements
Counting For Loop
for *i* in range(*start*, *end* [, *step*]):
 statements
(*start* is included; *end* is not)

if statements

```
if myAge < 18:
   print("Too young") #If TRUE
prints this
elif my Age <21:
   print("Go ahead") #If TRUE
prints this
else:
   print("Bye!") #if FALSE
prints this
```

While loops

```
#while loops run as long as, or
while, a certain condition is
true
while True:
     #do something
else:
     #do something
#Example:
current_number = 1 #set the
first value
 #check the value of current_n-
umber and see if it is less than
or equal to 5
while current_number <=5:
   print(current_number) #print
out the value of the variable
   current_number += 1 #add one
to the variable
```

The loop will run again until the current_value variable becomes 6 and then it will stop. Use break and continue to control loop

for loops

```
colors = ['red', 'green',
'blue']
#colors is a list data type
for color in colors:
    #name each individual item
color within the colors list so
that you can output the
individual variable
    print(color)
```

write() method example

**Opening in append mode will add the new data to the end of the file"
with open ("*filename.txt*, "a") as File:
 File.write("Hello\n")

Read methods

read()	read the entire file and return its contents as a string
readlines()	read the entire file and return its contents as a list
readline()	read the next line in the file and returns its content as a string

read() adn readlines() work best for smaller files. readline() for larger files.

Function Definition

Function named blocks of code that are designed to do a specific task
def *name*(*arg1*, *arg2*, ...):
code statements
return *expr*

return: stores the variable
It can be with arguments or without it

Functions Example

Function definition with NO arguments/parameters
def helloWorld():
 print("Hello, world!")
Function definition WITH arguments/parameters
def helloUser(*firstName*):
 print("Hello", *firstName*)
Calling a function
helloWorld()

LISTS/TUPLE

List [ ]	Collection of items in a particular order. List indexes start at 0
Tuple ( )	It is a list but Unable to be changed

Lists functions Example

fruits =	['apple', 'banana' , 'orange']
print(fruits)	Output an entire list
print(fruits[2])	Output an element in a list: orange
fruits[0] = 'grapes'	Modifying an element in a list: apple by grapes
fruits.append('pear')	Adding an element to the end of a list
fruits.insert(0, 'mango')	adding a list element in a specific position
fruits.remove('banana')	removing a list element
fruits.pop(0)	removing a specific list element
fruits.pop()	removing the last list element
del fruits	removing an entire list
fruits.clear()	emptying a list
findApple = (fruits.count("apple"))	count for specific item
fruits. reverse()	reverse the order of list
fruits.sort()	sort the list. fruits.sort(key=str.lower) to make sure everything is in lowercase

Lists functions Example (cont)

sorted_fruits = sorted(fruits)	If you want the list to remain the same positions, you can use the sorted to create a copy of the sorted list without impacting the original list

Types of files

Text files	each line ends with a new line character (\n) or a carriage return character (\r) on Windows systems
Binary files	Are intended to be read by other programs, not humans. common types are: program files, image files, audio files, video files, database files and compressed files.

File fuctions

open(*filename*, mode)	**mode** is an optional argument that specifies how you want to open the file. r = read, a = append, w = write, b = binary.
filename.**close()**	close an open file object
print(*filename*.**read()**)	output the content of the file

File fuctions (cont)

with open(*filename*) as *newfilename*:	automatically close a file if an exception happens. Also, it allows to assign a name to the file object in the same line of code and ends with a colon: creating a code block
write() method	use write mode when you are creating a new file, not when you are working with an existing file of data, Open the file in append mode ("a") if you wish to add to an existing file.

A file path must be included if the file is not in the same directory as the Python program