Python Cheat Sheet

[1. Capturing the user input](#_otewno41hr5s)

[2. Strings](#_9avprxjsb24u)

[3. Lists](#_rhwip58rzwjz)

[Exercises](#_qs6n1jiqdee9)

### 

### 1. Capturing the user input

number = input(“Enter a number: “)

Prints “Enter a number :” on the screen and captures what the user typed into the variable called number. This variable is an integer

print (“your number is: “ + str(number))

Prints “your number is: 8” (if number is 8). str(number) is required to transform the number into a string

word = raw\_input(“Enter a word or letter: “)

Prints “Enter a word or letter” on the screen and captures what the user typed into the variable called word. This variable is a string

print (“your word is: “ + word)

Prints “your word is: boat” (if word is “boat”).

### 2. Strings

A string is a list of characters. For example

word = “Hello world”

word\_number = “100”

All the operations from the ”list” section apply to string as well

len(word)

Will return the number of characters of the word (in this example 10)

word.upper()

Will return the world in capital letters, in this case “HELLO WORLD”

word.lower()

Will return the world in capital letters, in this case “hello world”

word + word\_number

Will concatenate the 2 strings and print ”Hello world 100”

int(word\_number)

Will transform a string representing a number into a variable of type int which can be used by int operation. For example you can do 3 + int(word\_number)will return 100 whereas 3 + word\_number will return an error because it can not add an int and a str.

### 3. Lists

A list can contain anything (numbers, strings, boolean…). For example

colors = [“red”, “blue”, “yellow”]

numbers = [1, 2, 3, 4]

letters = [“a”, “b”, “c”]

len(colors)

Returns the number of elements in the list (3 in this case)

colors[0]

Returns the first element of a list (“red” in this case)

colors[0:2]

Returns the elements between position 0 and 2 (2 excluded) ([“red”, “blue”]in this case)

colors[:1]

Returns the elements after position 1([“yellow”]]in this case)

colors.append(“black”)

Adds an element at the end of the list

colors.sort()

Sort the list

colors.reverse()

Reverse the list

if “red” in colors: print “I found red in the list of colors”

Verify if an element in in a list

for element in colors: print element

Will go over each element of the list and will perform the operation listed. (in this case it will print all the colors of the list, one by one)

**', '.join**(letters)

Will create a string concatenating elements of letters separated by “,”. In this example, it will generate the string “a, b, c”. ‘:’.join(letters) will generate “a:b:c”.

### Exercises

* Start the python file with the comment #!/usr/bin/python
* Exercise 1
  + Ask user to enter a word and then print the number of characters in that word.

Enter a word: spaghetti

spaghetti has 9 characters

* Exercise 2
  + Ask a user to enter a word
  + Ask a user to enter a letter
  + Print if the word contains this letter

Enter a word: spaghetti

Enter a letter: t

spaghetti contains the letter t

Enter a word: spaghetti

Enter a letter: o

spaghetti does not contain the letter o

* Exercise 3
  + Ask the user a word and then print each letter on each line.

Enter a word: hello

h

e

l

l

o

* Exercise 4
  + Ask a user to enter a word
  + Ask a user to enter a letter
  + Print how many times this letter appear in the word

Enter a word: spaghetti

Enter a letter: s

spaghetti does contains 1 letter s

Enter a word: spaghetti

Enter a letter: t

spaghetti does contains 2 letter t

Enter a word: spaghetti

Enter a letter: v

spaghetti does not contain the letter v

* Exercise 5
  + Ask a user to enter a word
  + Ask a user to enter a letter
    - If letter is a space, then print “ok quitting the program.” and finish the the program
    - Otherwise, print how many times this letter appear in the word (like in exercise 4) and ask again to enter a letter

Enter a word: spaghetti

Enter a letter: s

spaghetti does contains 1 letter s

Enter a letter: t

spaghetti does contains 2 letter t

Enter a letter:

OK quitting the program.

* Exercise 6:
  + Ask a user to enter a word
  + Print the list of vowels and list of consonant of that word
  + Tip: use .append to add to a list

Enter a word: spaghetti

Vowels: aei

Consonants: spghtt

* Exercise 7a:
  + Given a list [“\_”, “\_”, “T”, “\_”, “T”], print it in a nice format as

Word to Guess: \_ \_ T \_ T \_

* Start from the program below to change the code of the function pretty\_print to make it look like the example.

#!/usr/bin/python

# This function nicely prints the list of character words

def pretty\_print(word):

print word

word\_to\_guess = [‘\_’, ‘\_’, ‘T’, ‘\_’, ‘T’]

pretty\_print(word\_to\_guess)

* Tip: use the “.join” operation from the list section of the cheat sheet
* Exercise 7b:
  + Now add some colors

Word to Guess: \_ \_ T \_ T \_

Tip: You can use the following function

# This function print in blue the string passed as argument.

def blue(word):

blue\_begin = u'\u001b[0;34m'

blue\_end = u'\u001b[0m'

return blue\_begin + word + blue\_end

* Exercise 7c
  + more colors

Word to Guess: \_ \_ T \_ T \_

Tip: You can use the following function

# This function print in red the string passed as argument.

def red(word):

red\_begin = u'\u001b[0;31m'

red\_end = u'\u001b[0m'

return red\_begin + word + red\_end

* Exercise 8 (turtle)
  + In this exercise we will draw the handman step by step.
  + You can use this documentation to find the function to call: <https://docs.python.org/2/library/turtle.html>
  + Create a file called pendu\_turtle.py which contains the following code

#!/usr/bin/python

Def step1():

# Draw an horizontal line

Def step2():

# Add a vertical

step1()

step2()

step3()

Hangman Game:

* Add “pretty” printing of the word to guess (reuse exercise 7)
* Count the number of tries and print it at each round
* Define a maximum number of lives and stop the game if user reached this number of lives
* Support words with repeated letters “emma”:
  + You need to replace the user of index by a for loop
* Randomly choose a word from a list:
  + Create a function with the code of the hangman we have so far

# Run the hangman algorithm for this game

def hangman(word):

...

* + Outside the function:
    - define a list of words
    - choose a word from this list
    - call the hangman function for this word
* Randomly choose a word from a list which is stored in a document
* Add level to the game:
  + Level 1 is for words with 3 letters
  + Level 2 is for words with 4 letters
  + …..
* Draw the hangman with turtle
  + Copy the file pendu\_turtle.py
  + Add to the top of your file
  + Import pendu\_tutle.py