Exercises Python Fundamentals

# Exercise 1.1 – Python prompt

Experiment with the python prompt

1. Open the python prompt.
   * by opening IDLE
   * or by typing python in the command window
   * or in any other IDE. For example PyCharm.
2. Execute several simple numeric calculations.
3. Use the print function to print Hello World.

# Exercise 1.2 – Hello

Create and execute a Python module

1. Open IDLE or an other IDE of your choice
2. Create a new Python file called first.py
3. Save this file in a newly created directory for this course
4. Use the print function to **print** Hello World
5. **Save** the file as first.py
6. **Run** the file
7. Change the file to first ask for your name and store the result in a variable **name** using the **input()** function
8. Use the print function to print "Hello Albert" (if Albert is your name).
9. Save and run the file

# Exercise 1.3 – Leapyear

Write a program that determines if a year is a leapyear.

1. Create a new Python module with a name like leapyear.py
2. Then ask the user to **input** a year.
3. Change the input to a number using **int()**
4. Calculate if the year is a leapyear
   1. a year is a leapyear if the **year can be divided by 4**
   2. but (and) the **year can not be divided by 100**
   3. except (or) if the **year can be divided by 400**
5. Print the result
6. Test your program for different years

Tip: Use the modulo operator to compare the remainder of a division with 0 to determine if a number can be divided by another number. E.g.: 2021 % 4 == 0.

# Exercise 1.4 – Dimensions of a circle

Write a program that calculates the area and circumference of a circle.

1. Create a new Python module with a name like circle.py
2. First **import** the math library.
3. Then ask the user to **input** the radius.
4. Change the input to a number using **float**() and assign to a variable **r**.
5. Calculate the area with **area = πr2**
6. Calculate the circumference with **circumference = 2πr**
7. Print the results

Tip: The math library has a value for **π** in math.pi.

# Exercise 1.5 – Dice

Write a program that simulates throwing 5 dice.

1. Create a new Python module with a name like dice.py
2. **Import** the random library
3. Generate a random number between 1 and 6 with **random.randint(1, 6)** and store the number in a variable **dice1**.
4. Repeat this 4 more times creating variables dice2 up to dice5.
5. Print the values of the dice
6. Also print the total **sum** of the values

# Exercise 1.6 – Working with strings

Experiment with strings.

1. Create a new python file. E.g. strings.py
2. Ask the user to input some tekst and store the response in a variable t
3. Print the tekst in all **uppercase** and also in all **lowercase** characters
4. Use the **capitalize()** and **title()** methods and print the results
5. Print the first three characters by using slicing
6. Check if the tekst ends with a question mark
7. Print the text in lowercase with all spaces replaced by an underscore by using the method **replace()**. This is called **snake\_case**

# Exercise 1.7 – Life stage

Print the stage of life depending on the age entered by the user.

Age Life stage

0 – 2 Baby

2 – 4 Toddler

4 – 13 Kid

13 – 20 Teenager

20 – 65 Adult

65 or older Elder

Tips:

* Create a new python module
* Use **input()** to ask for the age
* Assign the integer value to a variable. Use **int().**
* Use a serie of **if** and **elif** statements to determine which message to print depending on the age entered. The upper bound is exclusive.

Exercise 5.6 (page 85) from the Book "Python Crash Course"

# Exercise 1.8 – Count vowels

* Get some tekst from input and put this in a variable
* Loop through the vowels ['a', 'e', 'i', 'o', 'u', 'y']
* Count the number of occurances of each vowel in the text
* Print a message for each vowel indicating the number of occurances
* After looping through the vowels
* … print a message indicating the total length of the text
* … and the total number of vowels

Output:

Found the vowel 'a' 58 times

Found the vowel 'e' 97 times

Found the vowel 'i' 66 times

Found the vowel 'o' 39 times

Found the vowel 'u' 23 times

Found the vowel 'y' 8 times

The complete text contains 929 characters.

The text contains 291 vowels.

# Exercise 2.1 – List of entered names

Enter a number of names. If no name is entered (return) continue with the rest of the program and print the entered names. Sorted if possible.

Tips:

* + Start with an empty list **names = []**
  + Use a **while** loop to ask for a name with **name = input(...)**
  + Add the entered name to the list with **names.append(name)**
  + If no name has been entered **break** out of the loop
  + **Print** the entered names in a for loop.
  + Sort the list with **sorted(names)**

# Exercise 2.2 – Occurance of words

* Get an arbitrary piece of text from internet
* Create a python script that reads the complete tekst with **s = input()**
* Convert to lowercase and remove dots and commas
  + use **s.lower().replace('.', '').replace(',', '')**
  + or **s.lower().translate(str.maketrans('', '', '.,!?()[]'))**
  + or with a regular expresson **re.sub('[^a-z\s]', '', s.lower())**
* Split the text into words with **text.split()**
* Create a set of unique words
* For each unique word count the number of occurances
* Store the results in a dictionary: **d[word] = n**
* Print the results: **for word, n in d.items()**

# Exercise 2.3 – Password generator

Generate a password of at least 6 characters   
with at least 1 capital, 1 lowercase, 1 number and 1 special character.

Tips:

* Start with 4 strings with character families.
  + E.g. capitals = 'ABCDEF..', numbers = '0123456789'
* Use random library to select a sample from these strings. The results are lists.
  + import random
  + part1 = random.choices(capitals, k=3)
* Concatenate the lists together.
  + characters = part1 + part2 + part3 + part4
* Shuffle the order of the elements with random.shuffle(characters).
* Turn the list of characters into a string with join():
  + password = ''.join(characters)
* print the generated password.

# Exercise 2.4 – Playing cards

Select 5 random cards from a deck of playing cards.

Tips:

* Define the 4 suits in a **list**
  + suits = ['clubs', 'diamonds', 'hearts', 'spades']
* Define the 13 ranks in a **list**
  + ranks = '2,3,4,5,6,7,8,9,10,J,Q,K,A'.split(',')
* Combine these lists in a new **list** with all combinations using a double list comprehension:
  + cards = [r + s for r in ranks for s in suits]
* Shuffle the list with **random.shuffle(cards)**
* Select 5 cards with **cards.pop()**,
  + hand = [cards.pop() for \_ in range(5)]

# Exercise 2.5 – Banner

Create a function that prints text surrounded by stars. Like a banner.

\*\*\*\*\*\*\*\*\*\*\*  
\* Peter \*

\*\*\*\*\*\*\*\*\*\*\*

Tips:

* Define the function called **banner**
* Define one argument called text
* Print out the lines

# Exercise 2.6 – Range of floats

The range function can only generate integers. Create a generator function that kan generate a sequence of floats similar to the bulit-in function range.

Tip:

* Define a function drange with arguments start, stop, step and endpoint. The endpoint arguments specifies if the endpoit is included or not.
* Give default values 1 for the step and False for endpoint.
  + E.g. **def drange(start, stop, step=1.0, endpoint=False)**
* Create a loop that calculates the numbers from start to end with an increment of step.
  + E.g. **number += step**
* If endpoint is set to true also include the endpoint also.
* You can use standard floats to achieve this but using Decimal will improve the precision. E.g. **from decimal import Decimal**

# Exercise 2.7 – Sort a list

* Enter a piece of tekst and split into words
* Use the **sorted** function to sort these words
* Create a function called number\_of\_vowels to count the number of vowels
  + tip: **sum([word.count(v) for v in 'aeiou'])**
* Use this function to sort the list on number of vowels
  + tip: **sorted(words, key=number\_of\_vowels)**

# Exercise 2.8 – Fibonacci Generator

Write a Python generator function, called fibonacci\_generator(…), that generates Fibonacci numbers up to a specified limit.

The generator must produce consecutive Fibonacci numbers until the next number in the sequence exceeds the limit.

# Exercise 2.9 – Prime Generator

Write a Python generator function, called prime\_generator(…), that generates prime numbers up to a specified limit.

Tips:

* Make function is\_prime() to determine, if the number is a prime number or not.