Lab -3

Programing in Python

Instructor : AALWAHAB DHULFIQAR

Advisor : Dr. Tejfel Mate

python

What you will learn:

More about print

Setup the lab environment

Semester schedule.

The grade system.

What we expect from you.

More about print ()

1)
print(
Number: % d %%% 12 ,) , thats it")
2)
print("episode: {}/{}, time: {}, rep: {}, Session:
{:.2}".format("friends", 13, 12.30, 2, 5)
3)

```
name = "ABX"
type_of_company = "XYZ"
print(f"{name} is an {type_of_company}
company.")
```

Use your brain to find the answers.

2 == 2
2 ==2.
x = 5, y = 10, z = 8 print(x > y) print(y > z)
x, y, z = 5, 10, 8 print(x > z) print((y - 5) == x)
x, y, z = 5, 10, 8 x, y, z = z, y, x print(x > z) print((y - 5) == x)

x = 10
if x == 10:
  print(x == 10)
if x > 5:
  print(x > 5)
if x < 10:
  print(x < 10)
else:
  print("else")

Lab 3.1

Objectives

becoming familiar with the the input() function;
becoming familiar with comparison operators in Python.

Scenario

Using one of the comparison operators in Python, write a simple two-line program that takes the parameter n as input, which is an integer, and prints False if n is less than 100, and True if n is greater than or equal to 100.

Don't create any if blocks (we're going to talk about them very soon). Test your code using the data we've provided for you.

Test Data

Sample input: 55

Expected output: False

Sample input: 99

Expected output: False

Sample input: 100

Expected output: True

Use your brain to find the answers

```
n = 3

while n > 0:
    print(n + 1)
    n -= 1
else:
    print(n)
```

```
n = range(4)

for num in n:
    print(num - 1)
else:
    print(num)
```

```
for i in range(0, 6, 3):
    print(i)
```

```
x = 1
y = 0

z = ((x == y) and (x == y)) or not(x == y)
print(not(z))
```

```
x = 4
y = 1

a = x & y
b = x | y
c = ~x  # tricky!
d = x ^ 5
e = x >> 2
f = x << 2

print(a, b, c, d, e, f
```

```
lst = [1, 2, 3, 4, 5]
lst.insert(1, 6)
del lst[0]
lst.append(1)
print(lst)
```

```
lst = [1, 2, 3, 4, 5]
lst_2 = []
add = 0
for number in lst:
    add += number
    lst_2.append(add)
print(lst_2)
```

```
lst = []
del lst
print(lst)
```

```
lst = [1, [2, 3], 4]
print(lst[1])
print(len(lst))
```

Lab 3.2

Analyzing code samples

```
# Read two numbers
number1 = int(input("Enter the first number: "))
number2 = int(input("Enter the second number: "))
# Choose the larger number
if number1 > number2:
    larger_number = number1
else:
    larger_number = number2
# Print the result
print("The larger number is:", larger_number)
```

```
# Read two numbers
number1 = int(input("Enter the first number: "))
number2 = int(input("Enter the second number: "))

# Choose the larger number
if number1 > number2: larger_number = number1
else: larger_number = number2

# Print the result
print("The larger number is:", larger_number)
```

```
# Read three numbers
number1 = int(input("Enter the first number: "))
number2 = int(input("Enter the second number: "))
number3 = int(input("Enter the third number: "))
# We temporarily assume that the first number
# is the largest one.
# We will verify this soon.
largest_number = number1

# We check if the second number is larger than current
largest_number
# and update largest_number if needed.
if number2 > largest_number:
    largest_number = number2

# We check if the third number is larger than current
largest_number
# and update largest_number if needed.
if number3 > largest_number:
    largest_number = number3

# Print the result
print("The largest number is:", largest_number)
```

Lab 3.3

Objectives

Familiarize the student with:

- using the if-elif-else statement;
- finding the proper implementation of verbally defined rules;
- testing code using sample input and output.

Scenario

As you surely know, due to some astronomical reasons, years may be leap or common. The former are 366 days long, while the latter are 365 days long.

Since the introduction of the Gregorian calendar (in 1582), the following rule is used to determine the kind of year:

- if the year number isn't divisible by four, it's a common year;
- otherwise, if the year number isn't divisible by 100, it's a leap year;
- otherwise, if the year number isn't divisible by 400, it's a common year;
- otherwise, it's a leap year.

Look at the code in the editor - it only reads a year number, and needs to be completed with the instructions implementing the test we've just described.

The code should output one of two possible messages, which are Leap year or Common year, depending on the value entered.

It would be good to verify if the entered year falls into the Gregorian era, and output a warning otherwise: Not within the Gregorian calendar period. Tip: use the != and % operators.

Test your code using the data we've provided.

Sample input: 2000

Expected output: Leap year

Sample input: 2015

Expected output: Common year

Sample input: 1999

Expected output: Common year

Lab 3.4

Objectives

Familiarize the student with:

- using the while loop;
- reflecting real-life situations in computer code.

Scenario

A junior magician has picked a secret number. He has hidden it in a variable named secret_number. He wants everyone who run his program to play the Guess the secret number game, and guess what number he has picked for them. Those who don't guess the number will be stuck in an endless loop forever! Unfortunately, he does not know how to complete the code.

Your task is to help the magician complete the code in the editor in such a way so that the code:

- will ask the user to enter an integer number;
- will use a while loop;
- will check whether the number entered by the user is the same as the number picked by the magician. If the number chosen by the user is different than the magician's secret number, the user should see the message "Ha ha! You're stuck in my loop!" and be prompted to enter a number again. If the number entered by the user matches the number picked by the magician, the number should be printed to the screen, and the magician should say the following words: "Well done, muggle! You are free now."

The magician is counting on you! Don't disappoint him.

```
secret_number = 777

print(
"""
+================================+
| Welcome to my game, muggle!    |
| Enter an integer number        |
| and guess what number I've     |
| picked for you.                |
| So, what is the secret number? |
+================================+
""")
```

Lab 3.5

Objectives

Familiarize the student with:

- using the break statement in loops;
- reflecting real-life situations in computer code.

Scenario

The break statement is used to exit/terminate a loop.

Design a program that uses a while loop and continuously asks the user to enter a word unless the user enters "chupacabra" as the secret exit word, in which case the message "You've successfully left the loop." should be printed to the screen, and the loop should terminate.

Don't print any of the words entered by the user. Use the concept of conditional execution and the break statement.

Objectives

Familiarize the student with:

- **using the continue statement in loops;**
- **modifying and upgrading the existing code;**
- **reflecting real-life situations in computer code.**

Scenario

Your task here is even more special than before: you must redesign the (ugly) vowel eater from the previous lab (3.1.2.10) and create a better, upgraded (pretty) vowel eater! Write a program that uses:

- **a for loop;**
- **the concept of conditional execution (if-elif-else)**
- **the continue statement.**

Your program must:

- **ask the user to enter a word;**
- **use user_word = user_word.upper() to convert the word entered by the user to upper case; we'll talk about the so-called string methods and the upper() method very soon - don't worry;**
- **use conditional execution and the continue statement to "eat" the following vowels A, E, I, O, U from the inputted word;**
- **assign the uneaten letters to the word_without_vowels variable and print the variable to the screen.**

Look at the code in the editor. We've created word_without_vowels and assigned an empty string to it. Use concatenation operation to ask Python to combine selected letters into a longer string during subsequent loop turns, and assign it to the word_without_vowels variable.

```
word_without_vowels = ""

# Prompt the user to enter a word
# and assign it to the user_word variable.


for letter in user_word:
    # Complete the body of the loop.

# Print the word assigned to
word_without_vowels.
```

Sample input: Gregory

Expected output:
GRGRY

Sample input: abstemious

Expected output:
BSTMS

Lab 3.6

Objectives

Familiarize the student with:

- **using the while loop;**
- **converting verbally defined loops into actual Python code.**

Scenario

In 1937, a German mathematician named Lothar Collatz formulated an intriguing hypothesis (it still remains unproven) which can be described in the following way:

- **take any non-negative and non-zero integer number and name it c0;**
- **if it's even, evaluate a new c0 as c0 ÷ 2;**
- **otherwise, if it's odd, evaluate a new c0 as 3 × c0 + 1;**
- **if c0 ≠ 1, skip to point 2.**

The hypothesis says that regardless of the initial value of c0, it will always go to 1.

Of course, it's an extremely complex task to use a computer in order to prove the hypothesis for any natural number (it may even require artificial intelligence), but you can use Python to check some individual numbers. Maybe you'll even find the one which would disprove the hypothesis.

Write a program which reads one natural number and executes the above steps as long as c0 remains different from 1. We also want you to count the steps needed to achieve the goal. Your code should output all the intermediate values of c0, too.

Hint: the most important part of the problem is how to transform Collatz's idea into a while loop - this is the key to success.

Test your code using the data we've provided.

Sample input: 16

Expected output:
8
4
2
1
steps = 4

Sample input: 15

Expected output:
46
23
70
35
106
53
160
80
40
20
10
5
16
8
4
2
1
steps = 17

Lab 3.7

Level of difficulty

Very easy
Objectives

Familiarize the student with:

- **using basic instructions related to lists;**
- **creating and modifying lists.**

Scenario

There once was a hat. The hat contained no rabbit, but a list of five numbers: 1, 2, 3, 4, and 5.

Your task is to:

- **write a line of code that prompts the user to replace the middle number in the list with an integer number entered by the user (Step 1)**
- **write a line of code that removes the last element from the list (Step 2)**
- **write a line of code that prints the length of the existing list (Step 3).**

```
hat_list = [1, 2, 3, 4, 5]  # This is an existing list of numbers hidden in the hat.

# Step 1: write a line of code that prompts the user
# to replace the middle number with an integer number entered by the user.

# Step 2: write a line of code that removes the last element from the list.

# Step 3: write a line of code that prints the length of the existing list.

print(hat_list)
```

Lab 3.8

Objectives

Familiarize the student with:

- **creating and modifying simple lists;**
- **using methods to modify lists.**

Scenario

The Beatles were one of the most popular music group of the 1960s, and the best-selling band in history. Some people consider them to be the most influential act of the rock era. Indeed, they were included in Time magazine's compilation of the 20th Century's 100 most influential people.

The band underwent many line-up changes, culminating in 1962 with the line-up of John Lennon, Paul McCartney, George Harrison, and Richard Starkey (better known as Ringo Starr).

Write a program that reflects these changes and lets you practice with the concept of lists. Your task is to:

- **step 1: create an empty list named beatles;**
- **step 2: use the append() method to add the following members of the band to the list: John Lennon, Paul McCartney, and George Harrison;**
- **step 3: use the for loop and the append() method to prompt the user to add the following members of the band to the list: Stu Sutcliffe, and Pete Best;**
- **step 4: use the del instruction to remove Stu Sutcliffe and Pete Best from the list;**
- **step 5: use the insert() method to add Ringo Starr to the beginning of the list.**

```
# step 1
print("Step 1:", beatles)

# step 2
print("Step 2:", beatles)

# step 3
print("Step 3:", beatles)

# step 4
print("Step 4:", beatles)

# step 5
print("Step 5:", beatles)


# testing list legth
print("The Fab", len(beatles))
```

Use your brain to find the answers

```
list_1 = ["A", "B", "C"]
list_2 = list_1
list_3 = list_2
del list_1[0]
del list_2[0]
print(list_3)
```

```
list_1 = ["A", "B", "C"]
list_2 = list_1
list_3 = list_2
del list_1[0]
del list_2
print(list_3)
```

```
list_1 = ["A", "B", "C"]
list_2 = list_1
list_3 = list_2

del list_1[0]
del list_2[:]

print(list_3)
```

```
list_1 = ["A", "B", "C"]
list_2 = list_1[:]
list_3 = list_2[:]
del list_1[0]
del list_2[0]
print(list_3)
```

```
my_list = [1, 2, "in", True, "ABC"]
print(1 ??? my_list) # outputs True
print("A" ??? my_list) # outputs True
print(3 ??? my_list) # outputs True
print(False ??? my_list) # outputs False
```

```
lst = ["D", "F", "A", "Z"]
lst.sort()

print(lst)
```

```
a = 3
b = 1
c = 2
lst = [a, c, b]
lst.sort()
print(lst)
```

```
a = "A"
b = "B"
c = "C"
d = " "
lst = [a, b, c, d]
lst.reverse()
print(lst)
```

Lab 3.9

Objectives
Familiarize the student with:

- list indexing;
- utilizing the in and not in operators.

Scenario
Imagine a list - not very long, not very complicated, just a simple list containing some integer numbers. Some of these numbers may be repeated, and this is the clue. We don't want any repetitions. We want them to be removed.

Your task is to write a program which removes all the number repetitions from the list. The goal is to have a list in which all the numbers appear not more than once.

Note: assume that the source list is hard-coded inside the code - you don't have to enter it from the keyboard. Of course, you can improve the code and add a part that can carry out a conversation with the user and obtain all the data from her/him.

Hint: we encourage you to create a new list as a temporary work area - you don't need to update the list in situ.

We've provided no test data, as that would be too easy. You can use our skeleton instead.

```
my_list = [1, 2, 4, 4, 1, 4, 2, 6, 2, 9]
#
# Write your code here.
#
print("The list with unique elements only:")
print(my_list)
```

Lab 3.10

Objectives

familiarizing the student with classic notions and algorithms;
improving the student's skills in defining and using functions.

Scenario

A natural number is prime if it is greater than 1 and has no divisors other than 1 and itself.

Complicated? Not at all. For example, 8 isn't a prime number, as you can divide it by 2 and 4 (we can't use divisors equal to 1 and 8, as the definition prohibits this).

On the other hand, 7 is a prime number, as we can't find any legal divisors for it.

Your task is to write a function checking whether a number is prime or not.

The function:

is called is_prime;
takes one argument (the value to check)
returns True if the argument is a prime number, and False otherwise.

Hint: try to divide the argument by all subsequent values (starting from 2) and check the remainder - if it's zero, your number cannot be a prime; think carefully about when you should stop the process.

If you need to know the square root of any value, you can utilize the ** operator. Remember: the square root of x is the same as x0.5

Complete the code in the editor.

```
def is_prime(num):
#
# Write your code here.
#

for i in range(1, 20):
        if is_prime(i + 1):
                print(i + 1, end=" ")
print()
```

```
2 3 5 7 11 13 17 19
```

Lab 3.11

Objectives

improving the student's skills in defining, using and testing functions.

Scenario

A car's fuel consumption may be expressed in many different ways. For example, in Europe, it is shown as the amount of fuel consumed per 100 kilometers.

In the USA, it is shown as the number of miles traveled by a car using one gallon of fuel.

Your task is to write a pair of functions converting l/100km into mpg, and vice versa.

The functions:

are named liters_100km_to_miles_gallon and miles_gallon_to_liters_100km respectively;
take one argument (the value corresponding to their names)

Complete the code in the editor.

Run your code and check whether your output is the same as ours.

Here is some information to help you:

1 American mile = 1609.344 metres;
1 American gallon = 3.785411784 litres.

```
def liters_100km_to_miles_gallon(liters):
#
# Write your code here.
#
def miles_gallon_to_liters_100km(miles):
#
# Write your code here
#
print(liters_100km_to_miles_gallon(3.9))
print(liters_100km_to_miles_gallon(7.5))
print(liters_100km_to_miles_gallon(10.))
print(miles_gallon_to_liters_100km(60.3))
print(miles_gallon_to_liters_100km(31.4))
print(miles_gallon_to_liters_100km(23.5))
```

```
60.31143162393162
31.36194444444444
23.52145833333333
3.900739358761746
7.490910297239916
10.00913120567375
```

Use your brain to find the answers

The input() function is an example of a:

a) user-defined function
b) built-in function

```
hi()

def hi():
    print("hi!")
```

```
def hi():
    print("hi")

hi(5)
```

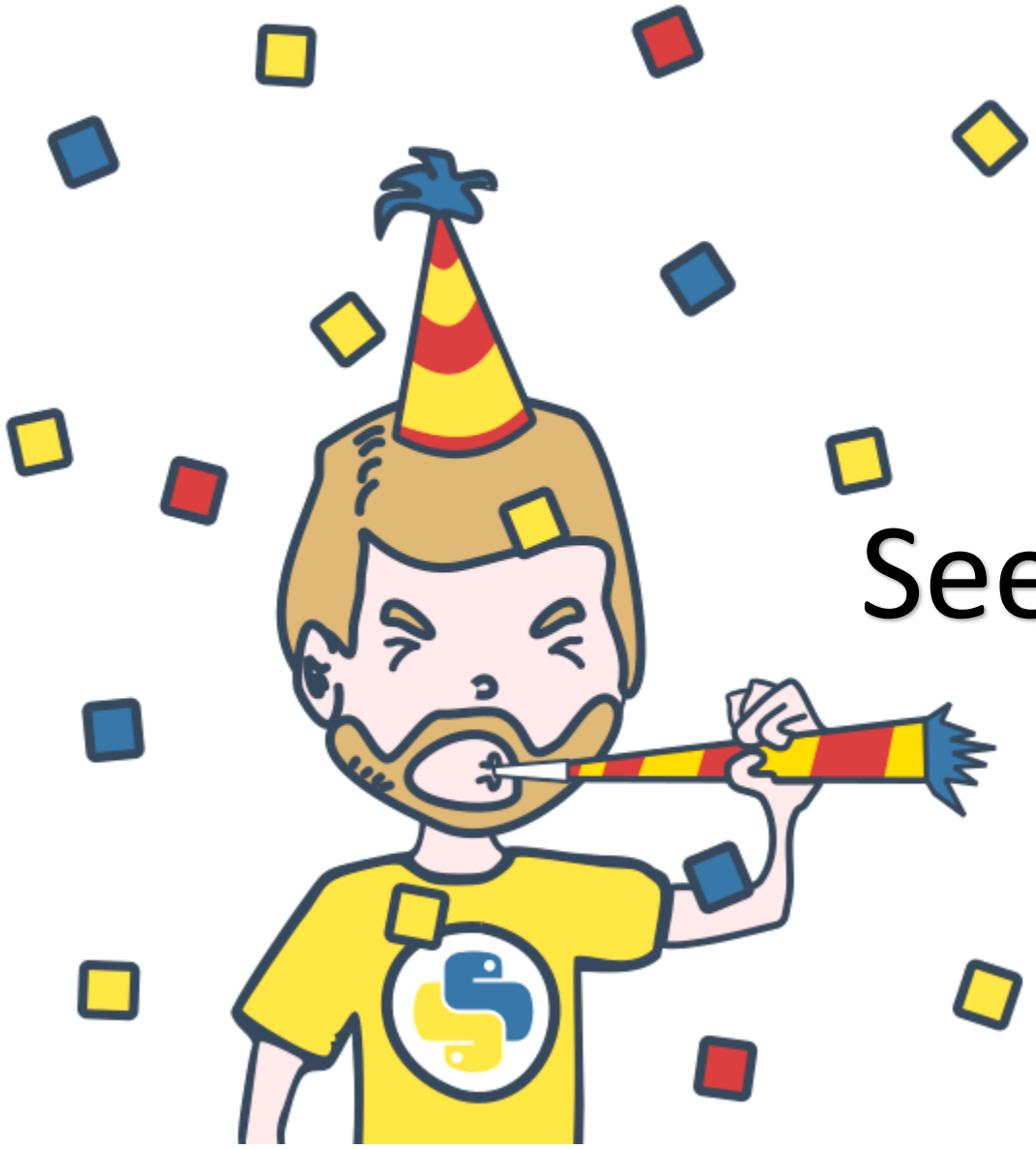
```
def hi():
    return
    print("Hi!")

hi()
```

```
def intro(a="James Bond", b="Bond"):
    print("My name is", b + ".", a + ".")

intro()
```

```
def add_numbers(a, b=2, c):
    print(a + b + c)

add_numbers(a=1, c=3)
```

```
def is_int(data):
    if type(data) == int:
        return True
    elif type(data) == float:
        return False

print(is_int(5))
print(is_int(5.0))
print(is_int("5"))
```

```
def message():
    alt = 1
    print("Hello, World!")

print(alt)
```



See you Next week ☺