The preview shows how the test is presented to participants. You can also see the correct answers for all questions except free text answers. No correct answers are shown to participants when they complete the test.

Total max score: 30

Start by reading through all questions. Peter will visit the room 45 minutes after the exam has started to clarify the questions you do not understand. Max score is 30 points.

- For grade 3, 40% of max score (12 points) is required.
- For grade 4, 60% of max score (18 points) is required.
- For grade 5, 80% of max score (24 points) is required.

You are not allowed to use the computer for anything else but answering the questions on this page.

Write your answers in either English or Swedish. If you write your answers in Swedish, make sure to not introduce any translation confusement. Write proper sentences (spelling, upper/lower case characters, punctuation, etc.). Answers that do not do this good enough/are vague/are ununderstandable cannot receive full score on the questions.

Good luck!

Place the following lines of code in such order they can be executed.

```
    return int(number) * 2
```

```
def get_number_times_two():
```

```
    number = input("Number: ")
```

```
number = get_number_times_two()
```

```
print(number)
```

Correct answer

```
def get_number_times_two():
```

```
   number = input("Number: ")
```

```
   return int(number) * 2
```

```
number = get_number_times_two()
```

```
print(number)
```

Max score: 1

How many statements and expressions does the following code contain?

```
numbers = [1, 2, 3]
sum = 0
for number in numbers:
   sum = sum + number
```

Enter you answers using digits, e.g. `0`, and not `zero`.

The code contains ______ (4) statements.

The code contains ______ (9) expressions.

Max score: 1

What type of error does the code below contain? The purpose of the function is to compute the average value of the two numbers it receives (e.g. `average(4, 6) → 5`).

```
def average(number_1, number_2):
    return (number_a + number_b) / 2
```

- ○ Syntax error
- ○ Runtime error
- ○ It does not contain an error at all.
- ○ Logic error

Max score: 1

What values will be stored in the variables after the following code has been executed?

```
res = 0
i = 0
while i <= 2:
    res = res + i
    i += 1
```

Enter you answers using digits, e.g. `0`, and not `zero`.

The variable `res` will store the number ______ (3).

The variable `i` will store the number ______ (3).

Max score: 1

What is what in the code below?

```
def get_doubled(x):
    return x * 2

doubled = get_doubled(3)
```

Pair each code piece with its corresponding name.

Note: Some code pieces may match multiple names, but there is only one way to get all 4 pairs right.

Correct answer

statement	`return x * 2`
variable	`doubled`
expression	x * 2
parameter	x

Max score: 1

Which one of the following statements is a bad suggestion on how to represent data?

○ To represent a playlist consisting of songs, one can use a list containing dicts, e.g. `playlist = [{"title": "The Hampsterdance Song", duration: 214}, {"title": "Cotton Eye Joe", duration: 194}]`.

○ To represent the number of pages in books at a library, one can use a list containing integers, e.g. `books_pages = [120, 59, 319]`.

○ To represent a song with a title and a duration, one can use a dictionary containing a strings and an integer, e.g. `song = {"title": "The Hampsterdance Song", duration: 214}`.

○ To represent a planet with a radius and a name, one can use a string, e.g. `planet = 'radius: 35483243, name: "Venus"'`.

○ To represent the number of ants in a wood, one can use an integer, e.g. `number_of_ants = 53866728`.

Max score: 1

The following expression:

```
range(3, 7, 2)
```

Creates a range containing some integers. What is the sum of the integers in that range?

Enter you answers using digits, e.g. `0`, and not `zero`.

The sum of the integers the range contains is ☐ (8).

Max score: 1

Below is a program that should:

1. Ask the user to enter his/her age.
2. Print the number of years left until the user become 100 years old.

```
entered_age = input("Enter your age: ")
years_left = 100 - entered_age
print("There is "+years_left+" years until you are 100 years old.")
```

When Bart runs the program, it does not work as intended. Explain why the program doesn't work the way it should, and suggest a solution to make it work.

Max score: 1

What integers should be assigned to the variables a and b to slice out the values "r", "s", "t" and "u" in the code below?

```
my_list = ["r", "s", "t", "u", "v"]
a = ?
b = ?
my_new_list = my_list[a:b]
# my_new_list should now be ["r", "s", "t", "u"]
```

Enter you answers using digits, e.g. `0`, and not `zero`.

The variable a should be assigned the integer (0).

The variable b should be assigned the integer (4).

Max score: 1

Write the following code:

```
characters = []
for x in "abcdef":
    characters.append(x)
```

As a list comprehension expression instead. Your own code should have the exact same meaning as the code above.

```
characters = [ENTER YOUR CODE HERE]
```

Max score: 1

Write the following code:

```
digits = []
for d in "21358372648":
    if d != "5":
        digits.append(int(d))
```

As a list comprehension expression instead. Your own code should have the exact same meaning as the code above.

```
digits = [WRITE YOUR CODE HERE]
```

Max score: 1

Here is a quite complex structure with information about the population in different areas:

```
populations = {
    "countries": [
        {"name": "Sweden", "population": 9903000},
        {"name": "Finland", "population": 5495000},
        {"name": "Norway", "population": 5233000}
    ],
    "cities": [
        {"name": "Stockholm", "population": 789024},
        {"name": "Jönköping", "population": 122952}
    ]
}
```

Given this structure, write an expression that evaluates to the population of the country with the name `Norway`, i.e. `5233000`.

Note: Do not write a statement, and simply writing `5233000` is of course not OK; the value needs to be retrieved from the structure.

Max score: 1

Suggest how the following data in Python:

```
cities = [
   {"name": "Stockholm", "population": 789024},
   {"name": "Jönköping", "population": 122952}
]
```

Can be written in XML code instead.

Max score: 1

Suggest how the following data in Python:

```
cities = [
   {"name": "Stockholm", "population": 789024},
   {"name": "Jönköping", "population": 122952}
]
```

Can be written in CSV format.

Max score: 1

The function `are_all_lower_than_10` below should check if all the numbers in the list of numbers passed to it are lower than 10, and in that case return `True`, othersie `False`.

```
def are_all_lower_than_10(numbers):
    for number in numbers:
        if number < 10:
            return True
        else:
            return False
```

At the moment, it does not work as it should. Explain why it does not work as it should, and explain what changes that needs to be made to make it work (do not write an entirely different solution; just describe the changes that needs to be made. There's no need to write any Python code, but if you think that makes your answer more readable, feel free to do so).

Max score: 1

What will be stored in the variable `sum` after the following code has been executed?

```
def a(number):
    number = number + 2

def b(list):
    for number in list:
        number = number + 3

def c(list):
    for i in range(len(list)):
        list[i] = list[i] + 4

my_list = [0, 0]
c(my_list)
b(my_list)
a(my_list[0])

sum = my_list[0] + my_list[1]
```

Enter you answer using digits, e.g. `0`, and not `zero`.

The variable `sum` will store the integer ______ (8).

Max score: 1

Implement the function `english_to_swedish`, which receives a string with the name of a digit in English (`"zero"`, `"one"` or `"two"`, and so on, all the way up to `"nine"`) and returns the name of the digit in Swedish.

Sample usage:

```
english_to_swedish("five") → "fem"
english_to_swedish("nine") → "nio"
```

Here is a translation table:

zero	noll
one	ett
two	två
three	tre
four	fyra
five	fem

six	sex
seven	sju
eight	åtta
nine	nio

Max score: 1

Write a program that keeps asking the user to enter an integer untill the user enters `stop`. The program should then print the smallest and the highest of all integers the user entered. When running the program, it can look like this (bold text represents text entered by the user).

```
Enter an integer or stop: 2
Enter an integer or stop: 6
Enter an integer or stop: 4
Enter an integer or stop: stop
The smallest integer is 2.
The highest integer is 6.
```

Note: The output should look precisely as in the example above (with the exception of the boldness).

Note: You can expect the user to actually enter numbers or stop (no error handling needed).

Note: You can expect the user to only enter positive integers.

Max score: 2

Implement the function `sum_between`, which receives two integers (the first one lower than the other), and returns the sum of the integers between (including) these. Write two different implementations of the function: one using a while loop, and another one using a for loop.

Sample usage:

```
sum_between(5, 8)    →   5 + 6 + 7 + 8   →  26
sum_between(1, 3)    →   1 + 2 + 3       →  6
```

Max score: 2

Implement the function `get_short_strings`, which receives a list with strings, and returns a new list containing the strings from the received list that are less than (including) 4 characters. To count the number of characters in a string, you can use the `len` function, e.g. `len("abc") → 3`.

Sample usage:

```
get_short_strings(["alice", "bob", "cloe", "david"]) → ["bob", "cloe"]
get_short_strings(["mac", "ios", "windows", "linux", "android"]) → ["mac", "ios"]
```

Max score: 2

The `movies` variable contains dicts representing different movies:

```
movies = [
    {"title": "GoldenEye",  "year": 1995, "length_in_minutes": 130},
    {"title": "Titanic",    "year": 1997, "length_in_minutes": 195},
    {"title": "Braveheart", "year": 1995, "length_in_minutes": 178},
    {"title": "I Robot",    "year": 2004, "length_in_minutes": 115},
    {"title": "Shrek",      "year": 2001, "length_in_minutes":  90},
    # And so on...
]
```

Write code that computes and prints the number of minutes it takes to watch all movies between the years 2000 (including) and 2007 (including) that are at least 100 minutes long, as well as how many those movies are.

Max score: 3

The class `NumberList` represents a list of unique numbers. It has the following constructor/methods:

Constructor/Method	**Description**
`NumberList()`	Creates a new `NumberList` containing no numbers.
`addNumber(number)`	Adds `number` to the `NumberList` if it does not already contain that number. Returns `True` if it didn't already contain the number, otherwise `False`.
`getCount()`	Returns the number of numbers in the `NumberList`.

Write a program making use of the class so a user can enter a number 5 times and have them added to the list (no duplicates). When running the code, it can look like this (bold text represents input from the user):

```
Enter a number: 5
Enter a number: 1
Enter a number: 2
Enter a number: 5
You have already entered that number.
Enter a number: 3
You have entered 4 unique numbers.
```

Note: The output should look precisely as in the example above (with the exception of the boldness).

Note: You can expect the user to actually enter numbers (no error handling needed).

Note: Make use of the class as much as possible, creating your own list is not OK.

Max score: 2

Another programmer wants you to create the class `Adder` that can be used like this:

```
my_adder = Adder()

my_adder.add(5)
my_adder.add(2)
my_adder.add(3)

sum = my_adder.get_sum()

print("The sum of 5, 2 and 3 is "+str(sum)+".")
```

When running the code above, the following should be printed to the console:

```
The sum of 5, 2 and 3 is 10.
```

Write the code implementing the `Adder` class so it works as explained above.

Max score: 2