

- If a multiple choice question has several correct answers, you must give all correct answers to receive credit.
  - **Write clearly.** Answers that are difficult to read may receive 0 points.
  - You must pass part A (4 correct out of 8 questions) to have your part B graded.
  - Write only on one side of each paper!
  - No **import** are allowed in your answers in part B (neither Python's standard modules nor external libraries) unless it is explicitly mentioned in the assignment. Built-in functions like **len**, **range**, **print**, and **sum** are allowed.
  - **Aids:** An A4 with as much information as you want. You can write on both sides.
  - **Grade thresholds:** E: 10, D: 12, C: 14, B: 16, A: 18, of maximum 20.
- 

## Part A: Multiple choice

Please collect your answers to part A on a single piece of paper.

1. What is printed by the code to the right?

- A. 1
- B. 9
- C. 11
- D. 19

```
y = 10
for x in [2, -4, 3]:
    y -= x
print(y)
```

E. An exception is raised.

2. Which list does the function `fcn` return?

- A. []
- B. [1, 2, 3, 4, 5]
- C. [2, 4, 6, 8, 10]
- D. [1, 4, 9, 16, 25]
- E. [0, 1, 4, 9, 16]

```
def fcn():
    res = []
    for i in range(5):
        res.append(i**2)
    return res
```

3. Which of these are correct syntax for creating a list `x` in Python?

- A. `x = []`
- B. `x = [2 , , , 7]`
- C. `x = [5, 9]`
- D. `x = [[1, 2], [3, 4]]`
- E. `x = [[1 ... 5], [1 ... 5]]`

4. What will be the value of the variable `text` after running the code to the right?

- A. `pot`
- B. `tp`
- C. `top`
- D. `t`
- E. `o`

```
text = ""
msg= "top"
i = 0
while i < len(msg):
    if i % 2 == 0:
        text += msg[i]
    i += 1
```

5. Which of the following statements are true for Unix (assuming the bash interpreter as used in the course compendium)?

- A. `cd` removes files and directories.
- B. `rm file.txt` removes the file `file.txt`
- C. `mkdir` can be used to create a folder.
- D. `wc` can count lines, characters, and words in files.
- E. `head` displays the beginning of a file.

6. What is printed by the code to the right?

- A. 0
- B. 1
- C. 4
- D. 8
- E. An exception is raised.

```
x = 1
y = 8
while y > 0:
    y //= 2
    x *= y
print(x)
```

7. What is the result of the code on the right?

- A. 0
- B. 2
- C. 4
- D. 8
- E. An error message

```
def f(x, y):
    return x + y

def g(y):
    return f(y, y) * 2

print(g(2))
```

8. What will be the contents of `d` after the code to the right is run?

- A. `{'D': 1, 'A': 1, '7': 1, '0': 1, '6': 2}`
- B. `{'D': 0, 'A': 1, '7': 2, '0': 3, '6': 9}`
- C. `{'D': 0, 'A': 1, '7': 2, '0': 3, '6': 5}`
- D. `{'D': 1, 'A': 1, '7': 1, '0': 1, '6': 4}`
- E. `{0: 'D', 1: 'A', 2: '7', 3: '0', 4: '6', 5: '6'}`

```
s = 'DA7066'
d = {}
for x in s:
    d[x] = s.count(x)
```

## Part B: Coding

Please use a separate piece of paper (or several) for each question in part B. Multipart questions such as 9A and 9B can be written on the same piece of paper.

9.

- A. Write a function `my_special_prod` that computes the product of all even integers in a list. For example, `[1, 2, 3, 4]` results in  $2 * 4 = 8$ . If there is only one even integer it should return the integer itself. If there is no even integer it should return 0. (2p)

**Example use:**

```
[In: ] print(my_special_prod([6, 2, 4]))
[Out:] 48
[In: ] print(my_special_prod([1, 3, 4]))
[Out:] 4
[In: ] print(my_special_prod([1, 1, 1]))
[Out:] 0
```

- B. Make `my_special_prod` able to handle lists that contain elements which are not of the type `int` (for example `str`, `float`, `bool`) by printing 'invalid datatype at list at index: X' for the items where this happens, where X is the position (i.e., index) of the variable in the list. (1p)

**Tip:** You can use the built-in function `type` that returns the type of the variable.

**Example usage:**

```
[In: ] print(my_special_prod([7.0, 6, 2, 4]))
[Out:] invalid datatype at list at index: 0
[Out:] 48
[In: ] print(my_special_prod([1, '1', 1, 1, False]))
[Out:] invalid datatype at list at index: 1
[Out:] invalid datatype at list at index: 4
[Out:] 0
```

10. The Fibonacci sequence  $f_0, f_1, f_2, \dots$  is a sequence of positive integers that starts with the two integers  $f_0 = 0$  and  $f_1 = 1$  where the  $i$ :th integer  $f_i = f_{i-1} + f_{i-2}$ ,  $i > 1$ . For example,  $f_2 = f_1 + f_0 = 1$  and  $f_3 = f_2 + f_1 = 2$ . Write a function `fibonacci` that returns the first  $n$  Fibonacci numbers in a list. (2p)

**Example usage:**

```
[In: ] print(fibonacci(1))
[Out:] [0]
[In: ] print(fibonacci(2))
[Out:] [0, 1]
[In: ] print(fibonacci(10))
[Out:] [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

11. Write a function `remove_duplicates` that takes a list and returns a new list with duplicates removed, preserving the original order. (2p)

**Example usage:**

```
[In: ] remove_duplicates([1, 2, 1, 2, 3, 4, 4, 5])
[Out:] [1, 2, 3, 4, 5]
```

12. Write a function `common_kv_pairs` that takes two dictionaries and returns a list of keys for which a key-value pair is identical between the two dictionaries. (2p)

**Example usage:**

```

[In: ] d1 = {1 : 1, 'a': 2, 'c': 3}
[In: ] d2 = {1 : 2, 'a': 5, 'c': 3}
[In: ] print(common_kv_pairs(d1, d2))
[Out:] ['c']
[In: ] d1 = {1 : 1, 'b': 2, 'z': 3}
[In: ] d2 = {'z' : 2, 2: 5, 3: 'z'}
[In: ] print(common_kv_pairs(d1, d2))
[Out:] []

```

13. Write a function `matrix_transpose` that takes a matrix represented as a two-dimensional list (a list of lists) and returns the transpose\* of the matrix. You do not have to check that the input `matrix` is correct (i.e., same length on all sublists). (3p)

**Example usage:**

```

[In: ] matrix = [[3, 5, 7], [1, 2, 3]]
[In: ] matrix_transpose(matrix)
[Out:] [[3, 1], [5, 2], [7, 3]]

```

\*The transpose  $A^T$  of a matrix  $A$  is an operator which, intuitively, 'flips a matrix over its diagonal'; that is, it switches the row and column indices of the matrix  $A$  by producing another matrix. For a two-by-two matrix  $A$ , we have

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \rightarrow A^T = \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{pmatrix}$$

For example, the transpose of

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

is

$$\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$$

The operation is also defined when the number of rows and columns are not identical. For example, the transpose of

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

is

$$\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$