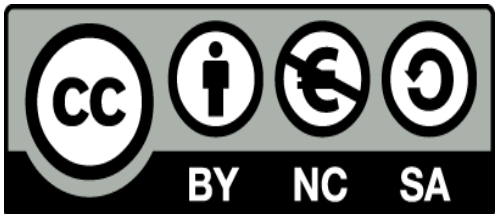
uc3m | Universidad Carlos III de Madrid



OPENCOURSEWARE
ADVANCED PROGRAMMING
STATISTICS FOR DATA SCIENCE

Ricardo Aler

MASTER IN STATISTICS FOR DATA SCIENCE. UC3M.

ADVANCED PROGRAMMING

2 HOURS

---

1. What is the expected result of this piece of code?

```
x=range(3)
y=x.append([4])
```

2. What would be printed by this piece of code?

```
x = [1, 2, '3']
y = x
y[0:1]=x[1:2]
print(x)
```

3. If *data* is a Pandas dataframe with a column named *'cn'*, would the following three options produce the same result? Why?

```
column_name = 'cn'

data[column_name] = 3             # option 1
data.column_name = 3              # option 2
data.loc[column_name] = 3         # option 3
```

4. What is the name of the Python concept that allows to do an operation like this:

```
np.array([1,2,3])+0
```

5. In order to do automatic hyper-parameter tuning, it is necessary to define several elements. Describe all those elements

6. What would be printed by this piece of code?

```
data = pd.DataFrame({'a': [1,2,3,4,5,6], 'b':[4,5,6,7,8,9]})
print(data.loc[3:5,'a'])
```

7. What would be the result of the following code?

```
mydictionary = {}
mydictionary[[1,2]] = 0
print(mydictionary)
```

8. What would be the contents of variable *result* after executing this piece of code? Why?

```
import numpy as np
result = np.array([1,2,3]+[4,4,4])+6
```

9. What would be the contents of variable *result* after executing this code?

```
result = 'abcdef'[5:1:-1]
```

10. Write two short pieces of code that do the same thing, but the first one uses a list, while the second one uses a tuple. The one for the list should work, the one for the tuple should be expected to raise an error.

11. What follows is the code for defining a new transformer step for pipelines. Modify this code so that the transformer apparently does the same thing, but now with data-leakage. Explain why data-leakage is happening in your new code:

```
class f(TransformerMixin):
    def __init__(self):
        pass
    def fit(self, X, y=None):
        self.sum_ = np.nansum(X)
        return(self)
    def transform(self, X):
        for j in range(X.shape[0]):
            for i in range(X.shape[1]):
                if(np.isnan(X[i,j])):
                    X[i,j]=self.sum_
        return(X)
```

12. Stan returns posterior distributions. How does Stan represent them?

13. What is the main difference between *sort* and *sorted*?

14. Why does Stan has a warmup period?

15. Write a piece of code that does the same thing than the one below, but it is much shorter:

```
mylist = [0,2,4,6]
result = []
for elem in mylist:
  result.append(elem+1)
print(result)
```