# LAB ĐIỂM CỘNG

## Python (15)

### Q1 (1)

Rename this file as `{fullname}.ipynb`.

### Q2 (1)

Create variable name `my\_name` as string and assign your name to this variable.

Print the `my\_name`.

# Your code here

my\_name="TranLuongHoa\_2174802010453"

print(my\_name)

### Q3 (2)

Develop class `CPStudent` with the following attributes

- `student\_id`:str -> vlu student id that start with `st`

- `name`:str -> Name of the associated student

- `score`:float -> Score of the midterm exam. The range is `[0.0, 100.0]`

During the object creation, assign `student\_id` and `name`. The `score` is set as `0.0` by default.

`s1 = DAstudent('207CT20000', 'Mr. Jo')`

# Your code

class CPStudent:

def \_\_init\_\_(self, student\_id: str, name: str):

# kiểm tra id của học sinh bắt đầu bằn st

if not student\_id.startswith('st'):

raise ValueError("student\_id must start with 'st'")

self.student\_id = student\_id

self.name = name

self.score = 0.0 #điểm mặc đinh được đặt là 0.0

def score(self):

return self.\_score

def score(self, value: float):

if not (0.0 <= value <= 100.0):

raise ValueError("điểm phải từ 0.0 hoặc =< 100.0")

self.\_score = value

def input\_score(self):

while True:

try:

user\_input = float(input("Enter the midterm score (0.0 - 100.0): "))

if 0.0 <= user\_input <= 100.0:

self.score = user\_input

break

else:

print("Điểm phải được nhập từ 0.0 đến 100.0 ")

except ValueError as e:

print(e)

print("hãy nhập điểm hợp lệ từ 0.0 <= 100.0")

def \_\_repr\_\_(self):

return f"CPStudent(student\_id='{self.student\_id}', name='{self.name}', score={self.score})"

s1 = CPStudent('st207CT20000', 'Mr. Jo')

print(s1)

s1.input\_score()

print(s1)

### Q4 (2)

Modify the class `DAStudent` to have function `LAB that takes `float` as the input.

This function will assign the input to `DAStudent.score`.

Example

```python

s1.lab(50.5)

print(s1.score)

>>> 50.5

```

### Q5 (3)

Check the correctness of the input. Inform the user when the input is wrong.

\*Hint: You will only get full mark if you use `raise`.\*

Example 1:

```python

s1.lab(150)

>>> ValueError: Some message

```

Example 2:

```python

s1.lab('150')

>>> ValueError: Some message

```

# Your code

class DAStudent:

def \_\_init\_\_(self, student\_id: str, name: str):

self.student\_id = student\_id

self.name = name

self.\_score = 0.0 # Default score is set to 0.0

@property

def score(self):

return self.\_score

@score.setter

def score(self, value: float):

if not isinstance(value, (int, float)):

raise ValueError("Score must be a number")

if not (0.0 <= value <= 100.0):

raise ValueError("Score must be between 0.0 and 100.0")

self.\_score = value

def lab(self, score: float):

if not isinstance(score, (int, float)):

raise ValueError("Score must be a number")

if not (0.0 <= score <= 100.0):

raise ValueError("Score must be between 0.0 and 100.0")

self.score = score

def \_\_repr\_\_(self):

return f"DAStudent(student\_id='{self.student\_id}', name='{self.name}', score={self.score})"

# Ví dụ sử dụng

s1 = DAStudent('st207CT20000', 'Mr. Jo')

print(s1) # DAStudent(student\_id='st207CT20000', name='Mr. Jo', score=0.0)

# Cho phép người dùng nhập điểm sử dụng phương thức lab

try:

s1.lab(50.5)

print(s1.score) # 50.5

except ValueError as e:

print(e)

# Thử gán điểm không hợp lệ

try:

s1.lab(150) # Sẽ raise ValueError

except ValueError as e:

print(e) # Score must be between 0.0 and 100.0

try:

s1.lab('150') # Sẽ raise ValueError

except ValueError as e:

print(e) # Score must be a number

### Q6 (6)

Once upon a time in the magical land, there are two engineer beavers living together.

Their hobby is to build a dam out of wood.

Years and years, dam after another dam, one of engineer beaver get bored. The beaver wants a better life. A more accurate calculation of dam not just keep building it anymore.

Now, they are deploying a rain sensor to collect data which will be used to predict water level of the river.

Finally, with their new skill set and hardworking, a predictor is obtained. They named their invention `The Oracle of the Dam (TOTD)`.

However, the `TOTD` only predicts the water level.

The engineer/data scientist beavers still needs to build the dam themselves but this time, it will be the most efficient dam of the beaver land ever.

Every 2 meters of the water level, the dam needs to be build 1 level higher. And every log (\*) uses will cost $3.5.

Here is how the 3-level dam look like from the cross section view

```

l\_3 \*

l\_2 \*\*\*

l\_1 \*\*\*\*\*

```

Here is how much this project will cost

```

Number of logs: 9

Cost: $31.5

```

Your task is simple, write a function `dam\_design` that takes `water\_level` as an input. Print the design of the dam and summarize the material and cost.

Example 1:

```python

dam\_design(0)

>>> \*

>>> Number of logs: 1

>>> Cost: $3.5

```

Example 2:

```python

dam\_design(1)

>>> \*

>>> Number of logs: 1

>>> Cost: $3.5

```

Example 3:

```python

dam\_design(2)

>>> \*

>>> \*\*\*

>>> Number of logs: 4

>>> Cost: $14.0

```

Example 4:

```python

dam\_design(3)

>>> \*

>>> \*\*\*

>>> Number of logs: 4

>>> Cost: $14.0

```

Example 5:

```python

dam\_design(4)

>>> \*

>>> \*\*\*

>>> \*\*\*\*\*

>>> Number of logs: 9

>>> Cost: $31.5

```

# Your code

def dam\_design(water\_level):

#tính số tầng đập

levels = water\_level // 2

#Tao tổng số phí logs và cost

total\_logs = 0

total\_cost = 0.0

# Print thiết kế đập và tính toán tổng bản ghis

for level in range(1, levels + 1):

logs = level # Số lượng logs ở cấp độ này

total\_logs += logs

# Print cấp độ có số lượng logs tương ứng

print(f"l\_{level} {'\*' \* logs}")

# tính tổng cost

total\_cost = total\_logs \* 3.5

# Print tổng số vật liệu và chi phí

print(f"\nNumber of logs: {total\_logs}")

print(f"Cost: ${total\_cost:.2f}")

water\_level = 6

dam\_design(water\_level)

Link github : <https://github.com/Daisy0401/Phan-Tich-Du-Lieu-Hoc-Sau_VLU_233_71ITDS30203_0103>