Functional Programming – MQMS

Below, are some functional programming concepts in the code snippets for `student_module.py`, `admin_module.py`, and `quiz_system.py`:

In the student_module.py:

1. Higher-Order Function

```
def process_results(results, display_function):
    for result in results:
        display_function(result)
```

2. Anonymous Function (Lambda)

```
display_result = lambda result: print(f"Question: {result['question']}, Answer: {result['answer']}")
```

3. Using map function

```
results = [{'question': 'Q1', 'answer': 'A1'}, {'question': 'Q2', 'answer': 'A2'}]
mapped_results = list(map(lambda r: f"Question: {r['question']}, Answer: {r['answer']}", results))
```

4. Using filter function

```
filtered_results = list(filter(lambda r: r['answer'] == 'A1', results))
```

In the admin_module.py:

1. Higher-Order Function

```
def process_quizzes(quizzes, processing_function):
  for quiz in quizzes:
    processing_function(quiz)
```

2. Pure Function

```
def add_quiz(quiz_list, new_quiz):
  return quiz_list + [new_quiz]
```

3. Using reduce function

```
from functools import reduce
total_questions = reduce(lambda acc, quiz: acc + len(quiz['questions']), quizzes, 0)
```

In the quiz_system.py:

1. Immutability

```
class QuizSystem:
  def __init__(self, quizzes):
    self.quizzes = quizzes

      def get_quizzes(self):
        return self.quizzes

      def add_quiz(self, new_quiz):
```

2. Creating a new instance with the added quiz

```
return QuizSystem(self.quizzes + [new_quiz])
```

3. Partial Function Application

```
def create_quiz_system_with_category(category):
  return lambda quizzes: QuizSystem([quiz for quiz in quizzes if quiz['category'] == category])
```

4. Using itertools for lazy evaluation

```
from itertools import count
count_up_to_10 = count(start=1, step=1)
```

5. Using generator expression

```
squared_numbers = (x**2 for x in range(10))
```

The above highlight showcases functional programming concepts like higher-order functions, anonymous functions (lambda), immutability, pure functions, and the use of functional programming tools like `map`, `filter`, and `reduce`. Take note that Python is not a purely functional language, so these aspects are used in a rational way to improve code readability and maintainability.