```java
public static List<CourseInfo> getCourses() {
  try {
    List<CourseInfo> result = new ArrayList<CourseInfo>();
    final List<CourseInfo> courseInfos =
        getFromStepic("courses", CoursesContainer.class).courses;
    for (CourseInfo info : courseInfos) {
      final String courseType = info.getType();
      if (StringUtil.isEmptyOrSpaces(courseType)) continue;
      final List<String> typeLanguage = StringUtil.split(courseType, " ");
      if (typeLanguage.size() == 2 && PYCHARM_PREFIX.equals(typeLanguage.get(0))) {
        result.add(info);
      }
    }
    return result;
  }
  catch (IOException e) {
    LOG.error("Cannot load course list " + e.getMessage());
  }
  return Collections.emptyList();
}
```

```java
public static List<CourseInfo> getCourses() {
  try {
    List<CourseInfo> result = new ArrayList<CourseInfo>();
    int pageNumber = 0;
    boolean hasNext = addCoursesFromStepic(result, pageNumber);
    while (hasNext) {
      pageNumber += 1;
      hasNext = addCoursesFromStepic(result, pageNumber);
    }
    return result;
  }
  catch (IOException e) {
    LOG.error("Cannot load course list " + e.getMessage());
  }
  return Collections.emptyList();
}

private static boolean addCoursesFromStepic(List<CourseInfo> result, int pageNumber)
    throws IOException {
  final String url = pageNumber == 0 ? "courses" : "courses?page=" +
    String.valueOf(pageNumber);
  final CoursesContainer coursesContainer = getFromStepic(url, CoursesContainer.class);
  final List<CourseInfo> courseInfos = coursesContainer.courses;
  for (CourseInfo info : courseInfos) {
    final String courseType = info.getType();
    if (StringUtil.isEmptyOrSpaces(courseType)) continue;
    final List<String> typeLanguage = StringUtil.split(courseType, " ");
    if (typeLanguage.size() == 2 && PYCHARM_PREFIX.equals(typeLanguage.get(0))) {
      result.add(info);
    }
  }
  return coursesContainer.meta.containsKey("has_next") &&
    coursesContainer.meta.get("has_next") == Boolean.TRUE;
}
```

Legend:
- Extracted Code
- Added Code
- Call to the Extracted Method