МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий

Кафедра математического обеспечения и стандартизации информационных технологий

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1

по дисциплине «Программирование на Python»

Тема: «Основы языка Python»

Выполнили студенты ИКБО-11-22 Иванов И. И.
группы Пупкин В. В.

Принял преподаватель Иванов П. F.

Работа выполнена «__» _____ 20__

(Подпись студента)

Зачтено «__» _____ 20__

(Подпись преподавателя)

Москва 2025

Содержание

1. Ход работы

1.1. Введение

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

1.2. Пример работы с листингами

Здесь можно увидеть как добавить листинг программы из файла

Листинг 1 — asd

```
func (a *App) GetDependencyGraph(ctx context.Context, packageName
string) ([]models.Edge, error) {

  // tasksWg counts remaining tasks, not goroutines
  tasksWg := &sync.WaitGroup{}

  // graph with all package dependencies
  var result []models.Edge
  resMutex := &sync.Mutex{}

  // A set of visited dependencies
  visited := make(map[string]struct{})
  visMutex := &sync.Mutex{}

  // channel with fetching tasks
  taskChan := make(chan fetchTask, _defaultConcurrency)

  // may contain first caught error
  var firstErr error
  var once sync.Once

  tasksWg.Add(1)
  taskChan <- fetchTask{packageName: packageName}
  ctx2, cancel := context.WithCancel(ctx)
  for i := 0; i < _defaultConcurrency; i++ {
    go func(ctx context.Context) {
      for {
        select {
        case <-ctx.Done():
          return
        case task, ok := <-taskChan:
          if !ok {
            return
          }
       if ok := pushIfNotExist(visited, visMutex, task.packageName); !
ok {
            tasksWg.Done()
            return
```

Продолжение листинга 1

```
          }

                  deps,  err  :=  a.DepsProvider.FetchPackageDeps(ctx,
task.packageName)

          if err != nil {
            tasksWg.Done()
            if errors.Is(err, context.Canceled) {
              return
            }

            once.Do(func() {
              firstErr = err
              cancel()
            })
            return
          }
          resMutex.Lock()
          for _, dep := range deps {
           result = append(result, models.Edge{From: task.packageName,
To: dep})
          }
          resMutex.Unlock()

          tasksWg.Add(len(deps))
          for _, dep := range deps {
            taskChan <- fetchTask{dep}
          }
          tasksWg.Done()
        }
      }
    }(ctx2)
  }

  tasksWg.Wait()
  close(taskChan)
  cancel()
  if firstErr != nil {
    return nil, firstErr
  }
  return result, nil
}

func (a *App) Run(ctx context.Context, packageName string, output
io.Writer) error {
  graph, err := a.GetDependencyGraph(ctx, packageName)
  if err != nil {
    return fmt.Errorf("can't receive dependency graph: %w", err)
  }

  if err := a.Serializer.Serialize(graph, output); err != nil {
    return fmt.Errorf("can't write output: %w", err)
  }
  return nil
}
```