

QUIZ

- 1) Как выбрать директорию?
- 2) Как создать папку? Как создать файл .txt? Как создать файл .html?
- 3) Что такое commit?
- 4) Как переключиться с одной ветки на другую?
- 5) Создайте git-директорию, создайте в ней файл text.txt, в нем напишите что-нибудь, сохраните изменения. Создайте новую ветку, в ней измените содержимое файла, сохраните. Переключитесь на основную ветку, тоже измените содержимое файла и сохраните. Сделайте слияние веток

Данные, базы данных и системы управления базами данных

Данные — это набор фактов в необработанной или неорганизованной форме (например, числа или символы).

База данных представляет собой набор связанных данных.

(**Database** is a collection of related data.)

Она описывает деятельность одной или нескольких родственных организаций.

(It describes the activities of one or more related organizations.)

MySQL

Microsoft SQL Server

PostgreSQL

Oracle

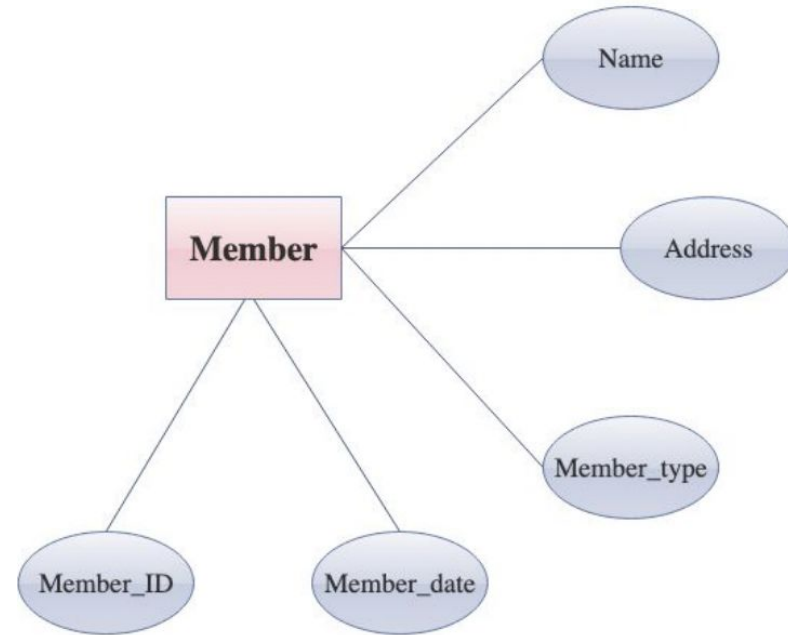
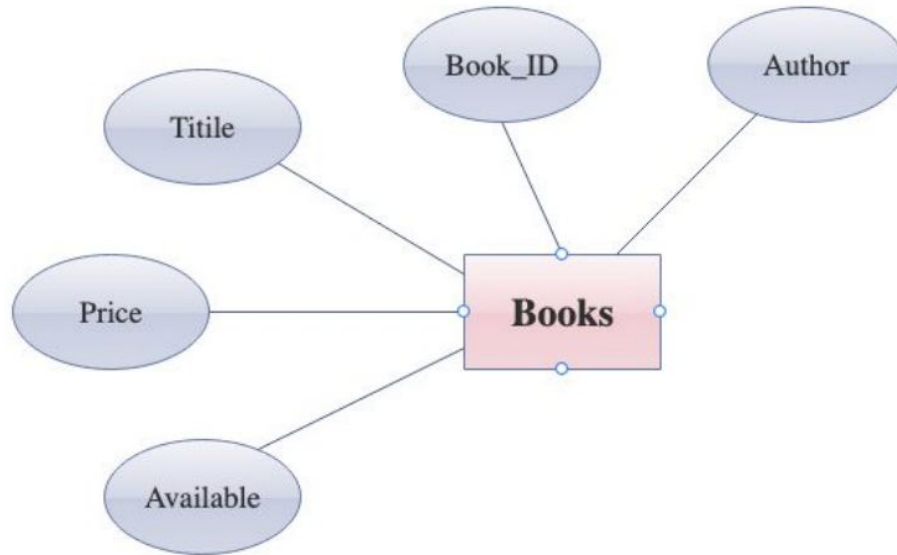
MongoDB

DB2

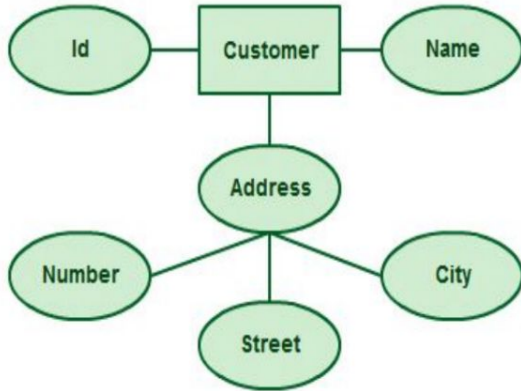
Microsoft Access

Redis

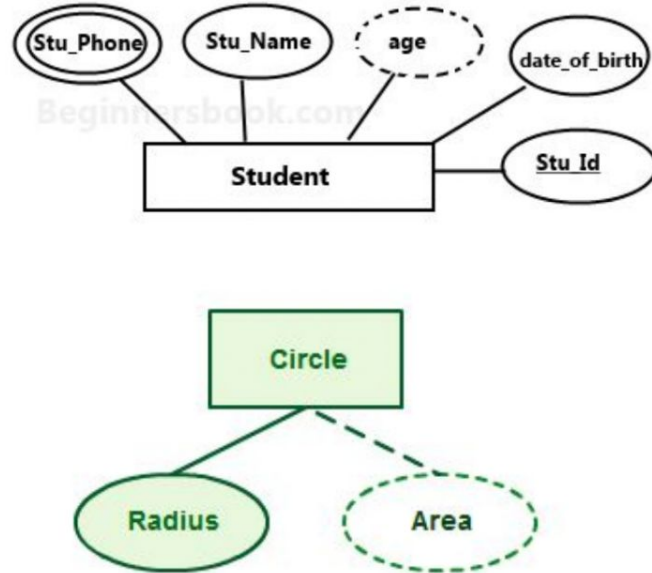
ER diagrams



Типы атрибутов



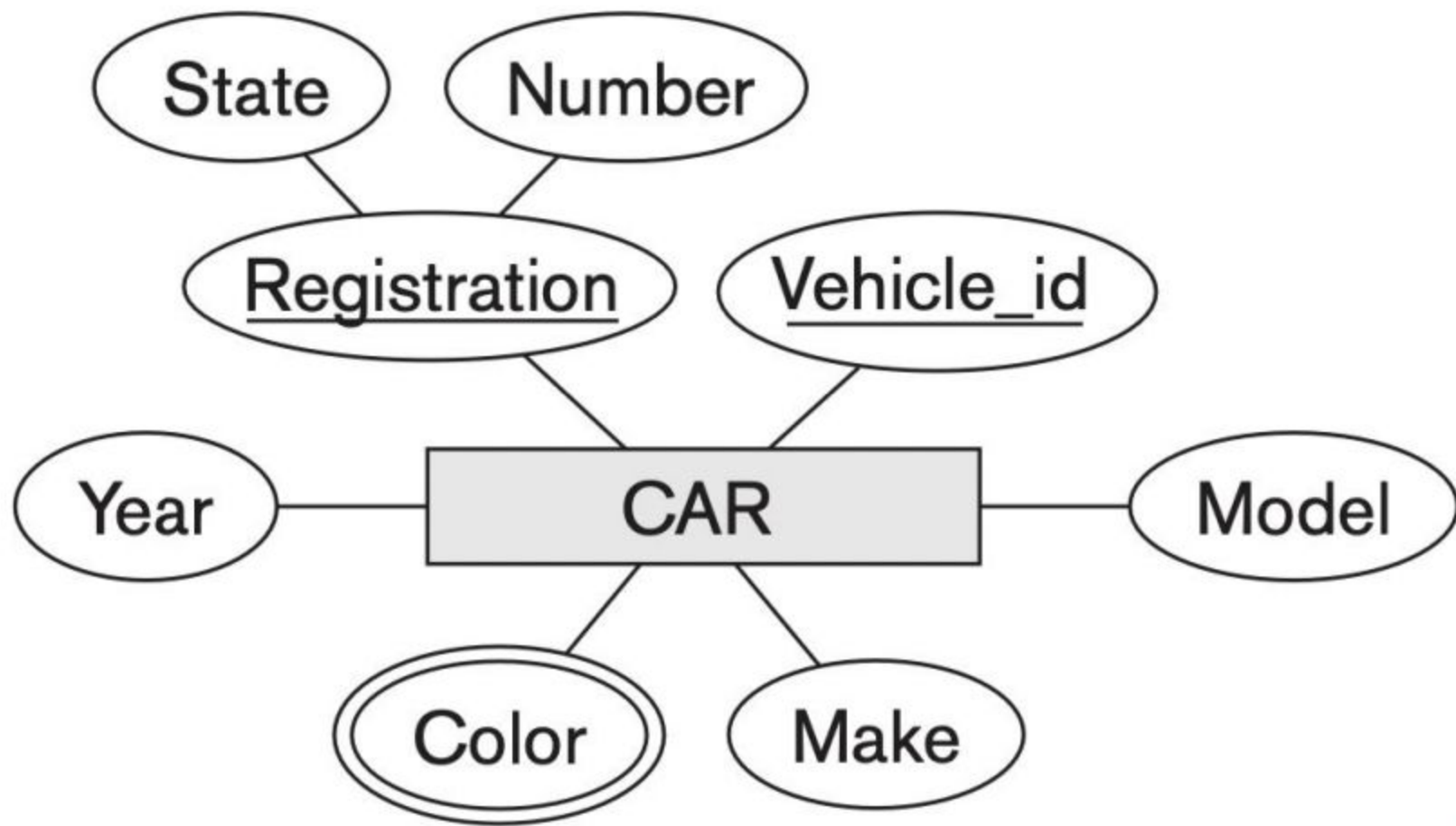
Composite vs. atomic attributes



Stored vs. Derived attributes

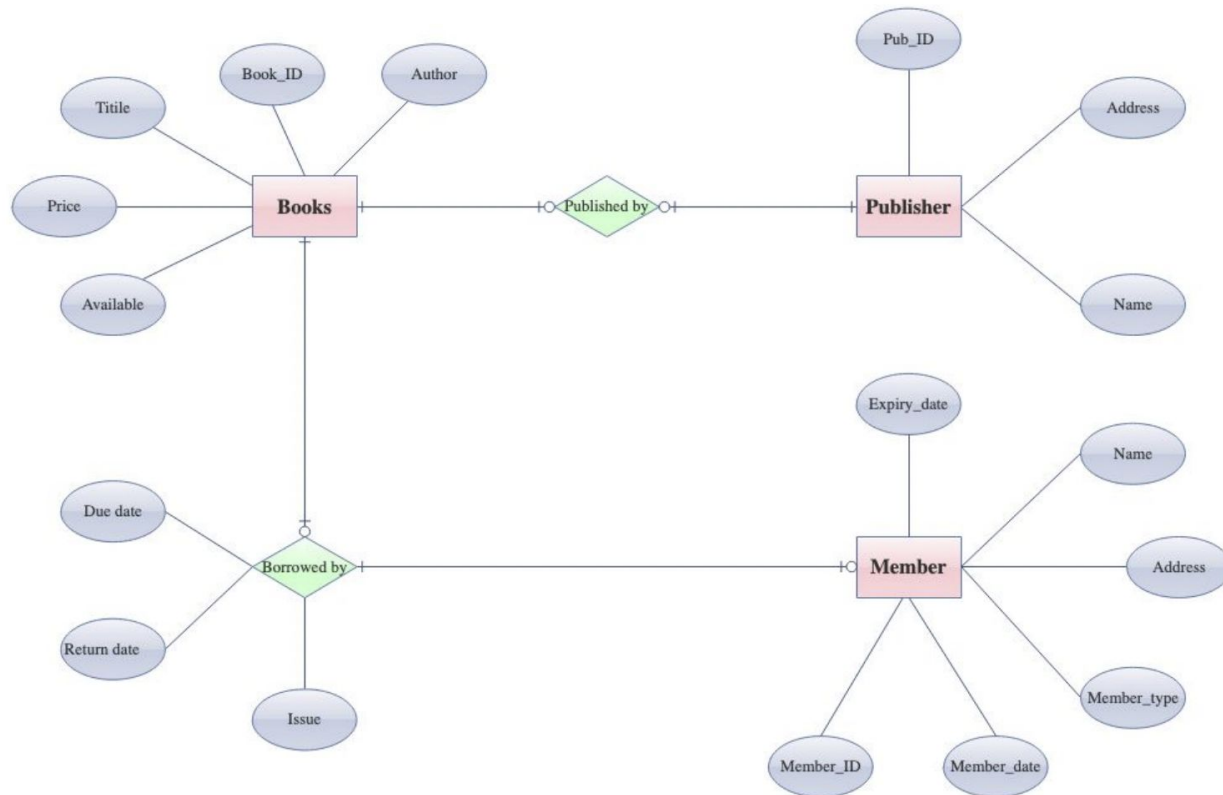



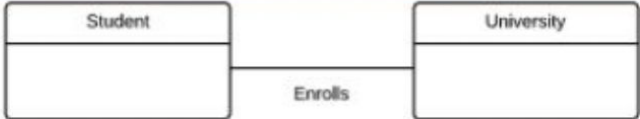












Multivalued
attributes



Пример

Library Management system



Notation	Meaning	Example
	Relationship	 <pre> graph LR Student[Student] --- Enrolls --- University[University] </pre>
	One	 <pre> graph LR Student[Student] === Has === StudentID[Student ID Number] </pre>
	Many	 <pre> graph LR Student[Student] } Attends } Class[Class] </pre>
	One and ONLY One	 <pre> graph LR Student[Student] === Uses === Chair[Chair] </pre>
	Zero or One	 <pre> graph LR Student[Student] === Has (o) SocialSecurity[Social Security Number] </pre>
	One or Many	 <pre> graph LR Instructor[Instructor] === Teaches } Class[Class] </pre>
	Zero or Many	 <pre> graph LR Classroom[Classroom] === Has (o) } Chair[Chair] </pre>

SQL (structured query language)

CREATE DATABASE testdb - Create testdb database

SELECT - extracts data from a database

UPDATE - updates data in a database

DELETE - deletes data from a database

INSERT INTO - inserts new data into a database

CREATE DATABASE - creates a new database

ALTER DATABASE - modifies a database

CREATE TABLE - creates a new table

ALTER TABLE - modifies a table

DROP TABLE - deletes a table

CREATE INDEX - creates an index (search key)

DROP INDEX - deletes an index

SELECT * FROM

Select * From

select * from

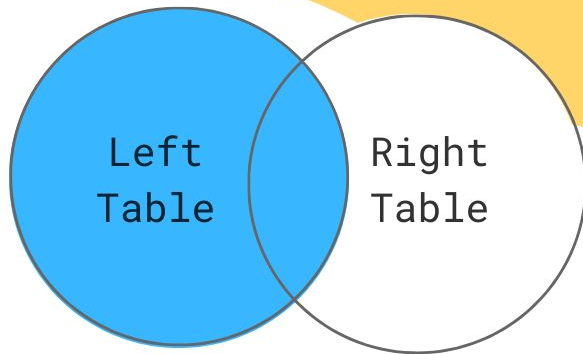
SeLEct * fRoM



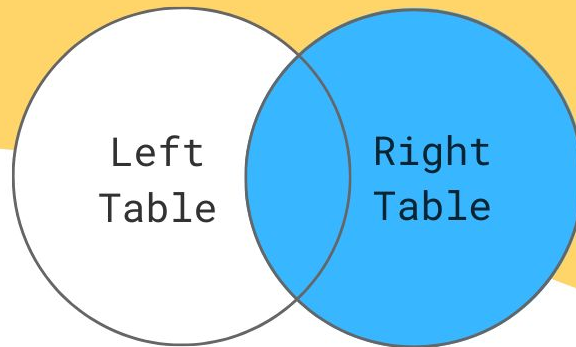
Joins

JOIN VISUAL	TYPE	DESCRIPTION
	INNER	DEFAULT: returns only the rows where matches were found
	LEFT OUTER	returns matches and all rows from the left listed table
	RIGHT OUTER	returns matches and all rows from the right listed table
	FULL OUTER	returns matches and all rows from both tables

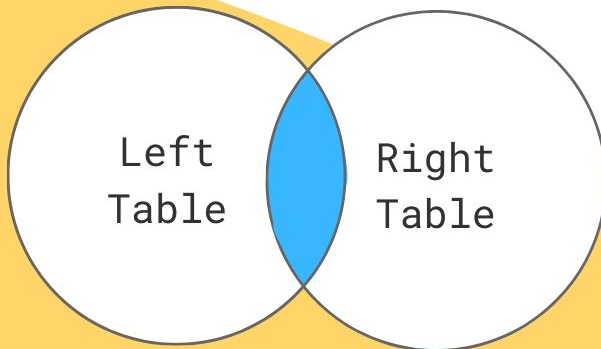
LEFT JOIN



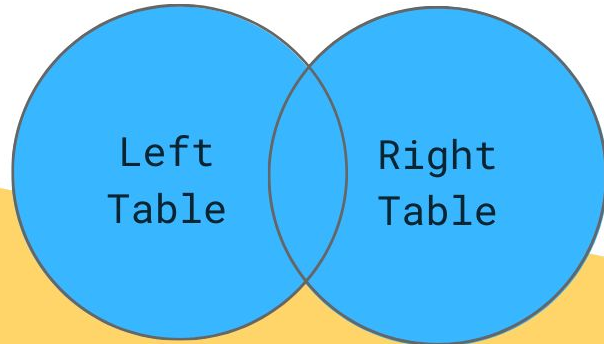
RIGHT JOIN



INNER JOIN



FULL JOIN



CREATE TABLE EMPLOYEE

(Fname	VARCHAR(15)	NOT NULL,
Minit	CHAR,	
Lname	VARCHAR(15)	NOT NULL,
Ssn	CHAR(9)	NOT NULL,
Bdate	DATE,	
Address	VARCHAR(30),	
Sex	CHAR,	
Salary	DECIMAL(10,2),	
Super_ssn	CHAR(9),	
Dno	INT	NOT NULL,

PRIMARY KEY (Ssn),

CREATE TABLE DEPARTMENT

(Dname	VARCHAR(15)	NOT NULL,
Dnumber	INT	NOT NULL,
Mgr_ssn	CHAR(9)	NOT NULL,
Mgr_start_date	DATE,	

PRIMARY KEY (Dnumber),

UNIQUE (Dname),

FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn));

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

Query 0. Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

Q0: **SELECT** Bdate, Address
 FROM EMPLOYEE
 WHERE Fname = 'John' **AND** Minit = 'B' **AND** Lname = 'Smith';

???

Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

```
SELECT      Fname, Lname, Address  
FROM        EMPLOYEE, DEPARTMENT  
WHERE       Dname = 'Research' AND Dnumber = Dno;
```

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

Q2: **SELECT** Pnumber, Dnum, Lname, Address, Bdate
 FROM PROJECT, DEPARTMENT, EMPLOYEE
 WHERE Dnum = Dnumber **AND** Mgr_ssn = Ssn **AND**
 Plocation = 'Stafford'

Query 8. For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.

Q8: **SELECT** E.Fname, E.Lname, S.Fname, S.Lname
 FROM EMPLOYEE **AS** E, EMPLOYEE **AS** S
 WHERE E.Super_ssn = S.Ssn;

Стандартная библиотека Python предоставляет минимальный, но полезный набор интерфейсов для работы с XML.

Двумя наиболее простыми и широко используемыми API для XML-данных являются интерфейсы **SAX** и **DOM**.

- Простой API для XML (**SAX**) — здесь вы регистрируете обратные вызовы для интересующих вас событий, а затем позволяете синтаксическому анализатору обрабатывать документ. Это полезно, когда ваши документы большие или у вас есть ограничения по памяти, он анализирует файл, когда читает его с диска, и весь файл никогда не сохраняется в памяти.
- API объектной модели документа (**DOM**) — это рекомендация Консорциума (Consortium) World Wide Web, при которой весь файл считывается в память и сохраняется в иерархической (древовидной) форме для представления всех функций XML-документа.

Асинхронный Питончик

event loop — цикл событий является ядром каждого асинхронного приложения. Циклы событий запускают асинхронные задачи и обратные вызовы, выполняют сетевые операции ввода-вывода и запускают подпроцессы.. По большей части всего лишь управляет выполнением различных задач: регистрирует поступление и запускает в подходящий момент.

корутины — специальные функции, похожие на генераторы python, от которых ожидают (await), что они будут отдавать управление обратно в цикл событий. Необходимо, чтобы они были запущены именно через цикл событий. Корутины содержат операторы yield, с помощью которых мы определяем места, где можно переключиться на другие ожидающие выполнения задачи.

футуры — объекты, в которых хранится текущий результат выполнения какой-либо задачи. Это может быть информация о том, что задача ещё не обработана или уже получен результат; а может быть вообще исключение

```
1  import asyncio
2
3  # корутина
4  async def foo():
5      print('Running in foo')
6      await asyncio.sleep(0)
7      print('Explicit context switch to foo again')
8
9  # корутина
10 async def bar():
11     print('Explicit context to bar')
12     await asyncio.sleep(0)
13     print('Implicit context switch back to bar')
14
15
16 ioloop = asyncio.get_event_loop()
17 tasks = [ioloop.create_task(foo()), ioloop.create_task(bar())]
18 wait_tasks = asyncio.wait(tasks)
19 ioloop.run_until_complete(wait_tasks)
20 ioloop.close()
```

Используя `await` в какой-либо корутине, мы таким образом объявляем, что корутина может отдавать управление обратно в `event loop`, который, в свою очередь, запустит какую-либо следующую задачу: `bar`. В `bar` произойдёт тоже самое: на `await asyncio.sleep` управление будет передано обратно в цикл событий, который в нужное время вернётся к выполнению `foo`.

```
Running in foo
```

```
Explicit context to bar
```

```
Explicit context switch to foo again
```

```
Implicit context switch back to bar
```

```
Process finished with exit code 0
```


threading module

```
1 import threading
2 import time
3
4 exitFlag = 0
5
6 class myThread (threading.Thread):
7     def __init__(self, threadID, name, counter):
8         threading.Thread.__init__(self)
9         self.threadID = threadID
10        self.name = name
11        self.counter = counter
12    def run(self):
13        print ("Starting " + self.name)
14        print_time(self.name, self.counter, 5)
15        print ("Exiting " + self.name)
```

```
17 def print_time(threadName, delay, counter):
18     while counter:
19         if exitFlag:
20             threadName.exit()
21         time.sleep(delay)
22         print ("%s: %s" % (threadName, time.ctime(time.time())))
23         counter -= 1
24
25 # Create new threads
26 thread1 = myThread(1, "Thread-1", 1)
27 thread2 = myThread(2, "Thread-2", 2)
28
29 # Start new Threads
30 thread1.start()
31 thread2.start()
32 thread1.join()
33 thread2.join()
34 print ("Exiting Main Thread")
```

Starting Thread-1

Starting Thread-2

Thread-1: Fri Feb 24 15:02:46 2023

Thread-2: Fri Feb 24 15:02:47 2023

Thread-1: Fri Feb 24 15:02:47 2023

Thread-1: Fri Feb 24 15:02:48 2023

Thread-2: Fri Feb 24 15:02:49 2023

Thread-1: Fri Feb 24 15:02:49 2023

Thread-1: Fri Feb 24 15:02:50 2023

Exiting Thread-1

Thread-2: Fri Feb 24 15:02:51 2023

Thread-2: Fri Feb 24 15:02:53 2023

run () — метод `run ()` является точкой входа для потока.

start () — метод `start ()` запускает поток, вызывая метод запуска.

join ([время]) — `join ()` ожидает завершения потоков.

isAlive () — метод `isAlive ()` проверяет, выполняется ли поток.

getName () — метод `getName ()` возвращает имя потока.

setName () — метод `setName ()` устанавливает имя потока.

Модуль потоков включает простой в реализации механизм блокировки, позволяющий синхронизировать потоки. Новая блокировка создается вызовом метода `Lock()`, который возвращает новую блокировку.

`acquire(blocking)` используется для принудительного выполнения потоков синхронно.

Метод `release()` нового объекта блокировки используется для снятия блокировки, когда она больше не требуется.

```
4  class myThread (threading.Thread):
5      def __init__(self, threadID, name, counter):
6          threading.Thread.__init__(self)
7          self.threadID = threadID
8          self.name = name
9          self.counter = counter
10     def run(self):
11         print ("Starting " + self.name)
12         # Get lock to synchronize threads
13         threadLock.acquire()
14         print_time(self.name, self.counter, 3)
15         # Free lock to release next thread
16         threadLock.release()
17
18     def print_time(threadName, delay, counter):
19         while counter:
20             time.sleep(delay)
21             print ("%s: %s" % (threadName, time.ctime(time.time())))
22             counter -= 1
```

```
24  threadLock = threading.Lock()
25  threads = []
26
27  # Create new threads
28  thread1 = myThread(1, "Thread-1", 1)
29  thread2 = myThread(2, "Thread-2", 2)
30
31  # Start new Threads
32  thread1.start()
33  thread2.start()
34
35  # Add threads to thread list
36  threads.append(thread1)
37  threads.append(thread2)
38
39  # Wait for all threads to complete
40  for t in threads:
41      t.join()
42  print_("Exiting Main Thread")
```


Starting Thread-1

Starting Thread-2

Thread-1: Fri Feb 24 15:13:07 2023

Thread-1: Fri Feb 24 15:13:08 2023



Thread-1: Fri Feb 24 15:13:09 2023

Thread-2: Fri Feb 24 15:13:11 2023

Thread-2: Fri Feb 24 15:13:13 2023

Thread-2: Fri Feb 24 15:13:15 2023

Exiting Main Thread

```
10   def run(self):
11     print("Starting " + self.name)
12     # Get lock to synchronize threads
13     # threadLock.acquire()
14     print_time(self.name, self.counter, 3)
15     # Free lock to release next thread
16     # threadLock.release()
17
```

un:  quiz2 x

Starting Thread-1

Starting Thread-2

Thread-1: Fri Feb 24 15:17:08 2023

Thread-2: Fri Feb 24 15:17:09 2023

Thread-1: Fri Feb 24 15:17:09 2023

Thread-1: Fri Feb 24 15:17:10 2023

Thread-2: Fri Feb 24 15:17:11 2023

Thread-2: Fri Feb 24 15:17:13 2023

Exiting Main Thread

GIT

<https://githowto.com/>

Git — мощная и сложная распределенная система контроля версий. Понимание всех возможностей git открывает для разработчика новые горизонты в управлении исходным кодом. Самый верный способ обучиться владению Git — испытать его своими руками.

<https://git-scm.com/download/mac>

<https://gitforwindows.org/>

```
[damirnurtdinov@MacBook-Pro ~ % cd Desktop/MyJob/Moscow  
[damirnurtdinov@MacBook-Pro Moscow % mkdir git_learning  
[damirnurtdinov@MacBook-Pro Moscow % ls  
CMakeLists.txt          git_learning  
cmake-build-debug       main.cpp  
[damirnurtdinov@MacBook-Pro Moscow % cd git_learning  
[damirnurtdinov@MacBook-Pro git_learning % mkdir hello  
[damirnurtdinov@MacBook-Pro git_learning % cd hello  
[damirnurtdinov@MacBook-Pro hello % touch hello.html  
damirnurtdinov@MacBook-Pro hello %
```

Где мы находимся?

```
[damirnurtdinov@MacBook-Pro hello % git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/damirnurtdinov/Desktop/MyJob/Moscow/git_learning/hello/.git/
damirnurtdinov@MacBook-Pro hello %
```

Добавим файл в гит

```
[damirnurtdinov@MacBook-Pro hello % git add hello.html  
[damirnurtdinov@MacBook-Pro hello % git commit -m "First commit"  
[master (root-commit) eb700b1] First commit  
 1 file changed, 1 insertion(+)  
 create mode 100644 hello.html  
damirnurtdinov@MacBook-Pro hello %
```

Залили все на гитхаб

```
damirnurtdinov@MacBook-Pro hello % git remote add origin git@github.com:damurka5
/NewHumanitarianSchool.git
git branch -M main
[git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 219 bytes | 219.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:damurka5/NewHumanitarianSchool.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
damirnurtdinov@MacBook-Pro hello %
```


КОММИТ

```
[damirnurtdinov@MacBook-Pro hello % git commit -m "beautiful line"  
[main ac60e5c] beautiful line  
 1 file changed, 17 insertions(+), 1 deletion(-)  
rewrite hello.html (100%)  
damirnurtdinov@MacBook-Pro hello %
```

Создадим ветку style

```
[damirnurtdinov@MacBook-Pro hello % git checkout -b style  
Switched to a new branch 'style'  
[damirnurtdinov@MacBook-Pro hello % git status  
On branch style  
nothing to commit, working tree clean  
damirnurtdinov@MacBook-Pro hello %
```

Слияние

```
[damirnurtdinov@MacBook-Pro hello % git checkout style
Switched to branch 'style'
[damirnurtdinov@MacBook-Pro hello % git merge main
Updating 9ea8b73..77bfd46
Fast-forward
 lib/hello.html | 6 +++---
 lib/style.css   | 3 +++
 2 files changed, 6 insertions(+), 3 deletions(-)
 create mode 100644 lib/style.css
damirnurtdinov@MacBook-Pro hello %
```

Удаление ветки

```
[damirnurtdinov@MacBook-Pro hello % git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)
[damirnurtdinov@MacBook-Pro hello % git branch -d style
Deleted branch style (was 77bfd46).
damirnurtdinov@MacBook-Pro hello %
```

```
[damirnurtdinov@MacBook-Pro hello % git branch
* main
damirnurtdinov@MacBook-Pro hello %
```