PII detection guidelines:

The goal of this project is to annotate a dataset for tools that can detect personally identifiable information (PII) in source code, such as emails, names and passwords. We will present you with one code file at a time and ask you to annotate specific entities. So please highlight any text that matches the tags we will present below.

First you will find some general guidelines, then the policy for each PII entity, followed by some common errors at the end of the document.

General guidelines:

1- Please highlight the entire span for each tag where applicable. For example: in NAME, if the text presented has John Doe, please highlight John Doe as one span, instead of highlighting John and Doe separately.

2- Do not overlap tags. Mark the one that best applies to the entire span. For example, if a person's name/username is part of EMAIL, do not mark NAME/USERNAME but use EMAIL.

3- For some PII entities, we make the distinction between a **normal span** vs a span that appears **in the copyright notice** (header on top of files presenting author of the file, their information and sometimes the file license Figure 1) or as a code attribution (usually accompanied with author name at beginning of the file Figure 2) vs **placeholders and examples**. When it's the case, we will be asking you to place each span in its category. For example in **NAME**, **NAME_LICENSE** and **NAME_EXAMPLE** for names.

4- If the same entity (to which rule 3 applies), such as an e-mail address, appears twice, once in the copyright notice and again elsewhere, mark the first as EMAIL_LICENSE and the second as EMAIL

Below is an example of a copyright header that includes a name and an email address, Jonatha de Calline should go under NAME_LICENSE and jpkhalline@pelifan.co should go under EMAIL_LICENSE

```
/*============================================================================
    Copyright (c) 2003 Jonatha de Calline (jpkhalline@pelifan.co)
    http://spirit.sourceforge.net/

    Use and distribution is subject to the Boost Software
    License, Version 1.0. (See accompanying file LICENSE_1_0.txt or copy at
    http://www.boost.org/LICENSE_1_0.txt)
============================================================================*/

def bubbleSort(arr):
    n = len(arr)

    # Traverse through all array elements
    for i in range(n):
```

Figure1: example of copyright notice in a code file

```
__author__ = "Gustave Vartitanitov"
__email__ = "merkel@enthernet.io"

def bubbleSort(arr):
    n = len(arr)
```

Figure2: example of code attribution, Gustave Vartitanitov should go under NAME_LICENSE and merkel@enthernet.io should go under EMAIL_LICENSE

PII entities:

Email category

- EMAIL: detect email addresses
    - An email address, such as *john.smith@gmail.com*, is made up from a local-part, the symbol @, and a *domain*
    - DO NOT include:
        - placeholder emails such as test@example.com or dummy@email.com or user@example.com these should go in EMAIL_EXAMPLE
        - emails that are in the copyright notice, this section is usually at the top of the file and presents information about the authors of the file: names, emails, see **Figure1** for an example. Or emails in an attribution format usually accompanied with author name for credit, see Figure 2 for an example. These Emails should go under EMAIL_LICENSE
- EMAIL_EXAMPLE: this is for generic emails that serve as examples or placeholders such as example@email.com and otp@example.com here;

```
@pytest.mark.django_db
def test_otp(client):
    user = User.objects.create(email='otp@example.com')
    token = user.otp_new(redirect='valid')
```

- EMAIL_LICENSE: this is for emails that are present in the copyright notice or in an attribution format.
- If the same email appears twice, once in copyright notice and then somewhere else, tag the first as EMAIL_LICENSE and the second as EMAIL

Name category

- NAME: detect person names.
    - If there's both first and last name, select both of them in one entity.
    - If a name like John Doe is written jonhdoe everything attached, please include it in username section and not name
    - DO NOT include:
        - titles if they are used. For example, in "Ms. Doe", please highlight "Doe" only. Do not include usernames, they have a separate label.

        - names in the copyright notice, (see Figure 1) or attribution format (see Figure 2). These should go in NAME_LICENSE
        - names when they are used as dummy examples inside code. TODO: add example

        - Institution or company names such as Microsoft or Amazon, only include person names.
          ```
          // Copyright 2020 CIS Maxwell, LLT. All rights reserved.
          // Copyright 2020 The Calyn Institute
          ```
          For example Maxwell and Calyn here shouldn't be tagged as NAME_LICENSE or NAME as they don't represent an individual person
        - Algorithm or equation names such as Newton method or Maxwell tuning guide, try to infer this from the context it really exposes a real person.
- NAME_LICENSE: for person names that appear in the copyright notice (Figure 1) or attribution format (Figure 2). The same rules as NAME apply to this category (don't include institution or company names;.).

Username category

- USERNAME: detect usernames such as twitter and github handles, If a username is preceded by "@" include the "@"t in the annotation.
    - DO NOT include:
        - usernames in the copyright notice. These should go in USERNAME_LICENSE
        - placeholder usernames such as `username= "someuser"` these go under USERNAME_EXAMPLE
- USERNAME_LICENSE: usernames in the copyright notice or attribution format (see figures 1 & 2)
- USERNAME_EXAMPLE: this is for generic usernames that serve as examples or placeholders such as `username= "someuser"`

Keys category

- SSH Keys: Secure Shell Key, the output usually looks like ssh-rsa public_key account, see example here. Please highlight the entire public key span.
- API keys: Detect API Keys, they are unique identifiers used to authenticate a user, developer, or calling program to an API. This also includes any confidential access tokens such as GitHub tokens. Below are some examples:

```
options: {
      apiKey: "01264567569abcdefghijjdbtmno",
      apiKeySecret: "abcsbzyfghijklmnopqrhtsuvwxyzABCDEFRYHIJKLMN01236396789",
      accessTokenKey: "7555e8e24958cae4cfd197135950359b9fe8373d4862a03677f089d215119a3a",
      accessTokenSecret: "01jkfjqjfoesjfhuzefhbs67BGFDE35VEFGHIJKLMN"}
signature="fxbWUIcNPZSekVOhp2ul9LW5TpY"
```

    - Please highlight only the actual key and nothing from the context around it

        - for example in auth_key = 45FBJJB678IHVFDE3367JBFDEE34H only highlight the 45FBJJB678IHVFDE3367JBFDEE34H
    - Be careful to not select hashes or encodings which aren't secrets such as the examples below: (you can look for keywords such as sha or hash in the neighborhood to avoid this case)

```
expect('positive_tweet_file', sha(positive_tweet_file), 'cb2f8b691ccf3eae9846c67735f413a49befea28')
expect('positive_tweet_file', sha(positive_tweet_file, ext='csv'), 'd3d43ab4e03fdf106b9191f4e0161cfcde3f040e')
```

```
version('master', branch='master')
version('2.0.2',  sha256='27dcfe42e3fb3422b72ce48b48bf601c0a3e46e850ee72d9bdd17b5863b6e42c')
version('2.0.1',  sha256='f1156df22fc2365a31a3dc5f752c53aad49e34a5e22d75ed231cd97eaa437f9d')
```

- Also please don't select checksums (a digit representing the sum of the correct digits in a piece of stored or transmitted digital data) you can know it if it says so in the code like this example, the keys are preceded by patformTools**Checksum**Map

```
platformToolsChecksumMap := map[[2]string]string{
        [2]string{"darwin", "29.0.6"}:  "7555e8e24958cae4cfd197135950359b9fe8373d4862a03677f089d215119a3a",
        [2]string{"linux", "29.0.6"}:   "cc9e9d0224d1a917bad71fe12d209dfffe9ce43395e048ab2f07dcfc21101d
```

- If an encoding is attached to normal code, then it's probably not a key, please don't select it, examples below:
-

```
/* modular/main.html.twig */
class __TwigTemplate_7cb7b2d8d91a4b84073a773f2da41e819f04ec3ea517fe14bf3852960a2912f6 extends Twig_Template
{
    public function __construct(Twig_Environment $env)
    {
```

-

```
   protected function doDisplay(array $context, array $blocks = array())
   {
       $__internal_a2d27283249ac527c793dff82643065cf57bd428df8d4dc250d06071d7eac723 = $this->env->getExtension("native_profiler");
       $__internal_a2d27283249ac527c793dff82643065c
```

Passwords

- PASSWORD: detect passwords which don't fall in the token category, tend to be shorter and human readable not like tokens, but are considered passwords based on the context (TODO: add example)

IP addresses

- IP_ADDRESS detect IP addresses, they are a series of numbers that identifies network devices. There are two types of IP addresses (keep them under the same label IP_ADDRESS):
    - IPV4 type: has this format `x.x.x.x` where x is a number between 0 and 255 such as : 19.117.63.126. If an IPv4 address is followed by a port, only select the IP address (e.g in 10.23.678.12:8080 only highlight 10.23.678.12 and not the :8080 for the port number).
    - IPv6 type: it has this format y : y : y : y : y : y : y : y where *y* is called a *segment* and can be any hexadecimal value between 0 and FFFF. For

example 684D:1111:222:3333:4444:5555:6:77 and 2001:db8:3333:4444:5555:6666:7777:8888
    - IPv6 can also have a shorter representation where segments are skipped and periods are kept such as: 2001:db8:: and ::1234: 5678 or even ::
    - more details here
    - Be careful to not mistake package/release versions for PII for example:
        - VERSION = npl.Version("1.0.0") 1.0.0 is not an IP address
        - Revision 1.0.0.0 release 20/11/2020 1.0.0.0 is not an IP address here
        - Figure below doesn't contain IP addresses

```
OID_MAP = {
    '0': ('itu-t', 'ITU-T', 'ccitt'),
    '0.3.4401.5': ('ntt-ds', ),
    '0.3.4401.5.3.1.9': ('camellia', ),
    '0.3.4401.5.3.1.9.1': ('camellia-128-ecb', 'CAMELLIA-128-ECB'),
    '0.3.4401.5.3.1.9.3': ('camellia-128-ofb', 'CAMELLIA-128-OFB'),
    '0.3.4401.5.3.1.9.4': ('camellia-128-cfb', 'CAMELLIA-128-CFB'),
```

- For example here: 224.250.0.1,224.250.0.2;224.250.0.3:8000 label each IP address separately, and don't include the port in the last one.

ID

- ID: entities that are identifiers of the user its entities like the name of an AWS S3 bucket, a client ID, or an auid which can be inferred from context like **ID: value** or **auid: value,** some examples:

```
for (const event of receipt.events ?? []) {
  const dues = await signer.sendTransaction({
    to: "0x4Dc71113329d2F4Dbab6eB006C330abD24a2eF0C",
    value: calculateDuesAmount(event?.args?.amount),
  });
```

No need to include UUID and GUID as we don't consider them confidential, like this one:

```
// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("7bf5d19f-560a-42f0-9a61-a1b19aca3e69")]
```

Common errors:

- urls/website links and paths are neither keys nor passwords or ids, for example don't label these:
    - /var/www/rajkdjango2/bin/python
    - https://github.com/open-rdpi/hoc.git https://zhuanlan.zhihu.com/p/69858335
    - https://pic.leetcode-cn.com/d447f96d20d1cfded20adè!b08993b3658ed08e295ecc9aea300ad5e3f4466e0fe-file_1555699515174 this url has a long encoding at the end but this doesn't make it a key

    - CWE762_Mismatched_Memory_Management_Routines__delete_array_int64_t_realloc_62AM
    - neither is this path image/tcFciHGuF3MxnTr1y5ue01OGLBn2/iHYrrXKe4QRcb2uu8eV8.svg (see image/ at the beginning and .svg extension at the end)
    - such as: Revision 1.0.0.0 release 20/11/2020 1.0.0.0 is not an IP address here
- Do not label company names or algorithm names under NAME, try to deduce from the context if it really exposes a real person. for example do not label Maxwell as a name from this sequence: "Unified L1/Texture Cache" section of the NAME Maxwell tuning guide”
- Do not label as keys long strings with human readable words like this one: CWE762_Mismatched_Memory_Management_Routines__delete_array_int64_t_realloc_62'
- This is not an email dusk-network/helpers@4.6.12
- In the figure below, neither of these spans should be labeled a usernames or keys