1.10 String representations of objects: str() vs repr()

`str()` and `repr()` are builtin functions used to represent the object in the form of string.

Suppose we have an object `x`.

`str(x)` would be calling the dunder (double underscore) `__str__` method of `x` as `x.__str__()`

`repr(x)` would be calling the dunder (double underscore) `__repr__` method of `x` as `x.__repr__()`

😑 Well, what all are these new terms `__str__` and `__repr__` 🤔?

As we know that Python is object oriented language, and so supports inheritance. In Python, all the classes would inherit from the base class `object`. `object` class has the methods `__str__`, `__repr__` and a lot more (which can be deepdived in someother notebook 😉). Hence every class would be having `__str__` and `__repr__` implicitly 😊

Python's official documentations states that `__str__` should be used to represent a object which is human readable(informal), whereas `__repr__` is used for official representation of an object.

In [1]:
```
from datetime import datetime

now = datetime.now()

print(f"The repr of now is: {repr(now)}")
print(f"The str of now is: {str(now)}")
```

```
The repr of now is: datetime.datetime(2021, 5, 28, 13, 19, 7, 751471)
The str of now is: 2021-05-28 13:19:07.751471
```

In [2]:
```
class ProgrammingLanguage:
    def __init__(self, language: str):
        self.language = language


language_obj = ProgrammingLanguage(language="Python")
print(f"The repr of language_obj is: {repr(language_obj)}")
print(f"The str of language_obj is: {str(language_obj)}")
```

```
The repr of language_obj is: <__main__.ProgrammingLanguage object at 0x7faa74
0420a0>
The str of language_obj is: <__main__.ProgrammingLanguage object at 0x7faa740
420a0>
```

In the above example we see that output to be something like:

```
The repr of language_obj is: <__main__.Language object at
0x7f1580c67190>
The str of language_obj is: <__main__.Language object at
0x7f1580c67190>
```

The address of the object might be different for everyone

Now let's try to override the `__str__` and `__repr__` methods and see how the

representations work

In [3]:

```
class Human:
    def __init__(self, name: str, age: int):
        self.name = name
        self.age = age

    # overriding __str__ method
    def __str__(self):
        return f"I am {self.name} of age {self.age}"

    # overriding __repr__ method
    def __repr__(self):
        return f"Human(name={self.name}, age={self.age}) object at {hex(id(se


human_obj = Human(name="IronMan", age=48)
print(f"The repr of human_obj is: {repr(human_obj)}")
print(f"The str of human_obj is: {str(human_obj)}")
```

```
The repr of human_obj is: Human(name=IronMan, age=48) object at 0x7faa74090be
0
The str of human_obj is: I am IronMan of age 48
```

We see that the result representations of the `human_obj` have been changed as we have overridden the `__str__` and `__repr__` methods 😊