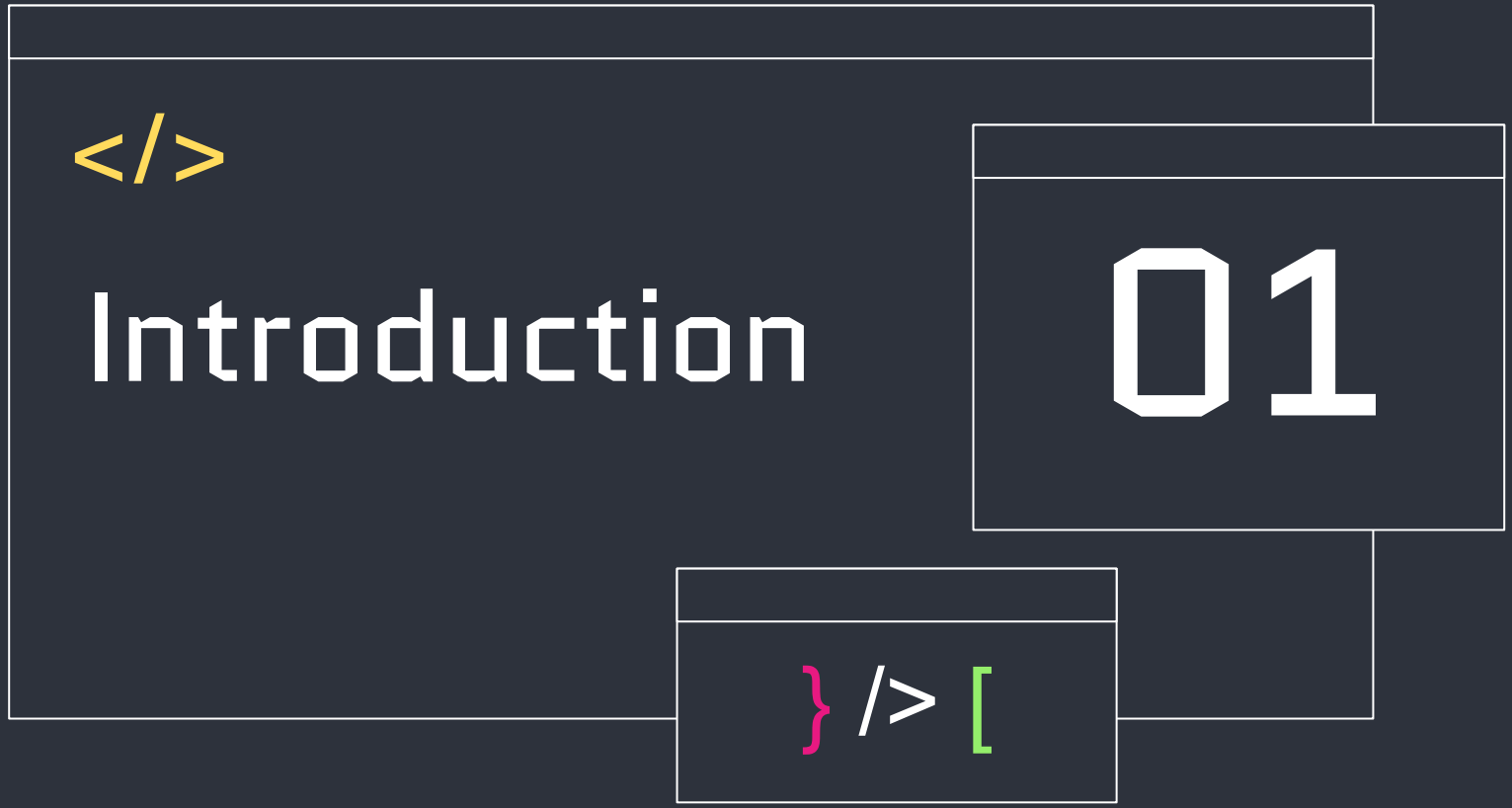
1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1

</ Table of contents

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1



1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1

Setting the Stage

1- Install python: https://python.org/

2- Create a directory.

3- Inside the dir. Create a file with extension py.

4- Run using: python main.py

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Python Data types

- **Basic Data Types:**
  - int
  - float
  - str
  - bool
  - list
  - dict
- **Mutable VS Immutable**
  - Immutable: Can't change after creation
    - int, float, str, bool, tuple.
    - If you "change" it, Python makes a new one.
  - Mutable: Can change after creation
    - list, dict, set.
    - You can modify them without making new ones.
- **Everything is an Object**

- **Dot (.) Notation**

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Classes and Objects

- What is a Class?
  - A class is like a blueprint or a cookie cutter. It defines:
    - What information an object will have (attributes)
    - What actions it can do (methods)

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def greet(self):
        print(f"Hello, my name is {self.name} and I'm {self.age} years old!")
```

- What is an Object?
  - An object is one specific thing made from the class blueprint (like one specific cookie from the cookie cutter).

```
person1 = Person("Sara", 22)
person2 = Person("Ali", 25)
print(person1.name)
print(person2.age)
person1.greet()
person2.greet()
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Key Points

- `class keyword`:
  - Defines a new blueprint.
    `Class Person:`
- `__init__ method`:
  - Special method that runs automatically when creating a new object.
    - self refers to the object being created.
    - We set the object's attributes.
- Creating object:
  - Like making cookies from the cutter.
    `person1 = Person("Sara", 22)`
- Attributes
  - Information stored in the object.
    `person1.name`
- Methods:
  - Actions the object can do.
    `person1.greet()`

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Methods in Classes

1. Instance Methods:
   - used with individual objects (instance) of a class.

```
Class Dog:
     def __init__(self,name):
          self.name = name

     def speak(self):
          return f"{self.name} says woof!"

my_dog = Dog("Koko")
print(my_dog.speak())
```

2. Class Methods:

```
class Dog:
     species = "Canine"

     @classmethod
     def get_species(cls):
          return cls.species
print(Dog.get_species())
```

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

3. Static Methods:

```
class Math:
    @staticmethod
    def add(x, y):
        Return x + y
print(Math.add(5, 3))
```

When to Use Each:

1. **Instance Method: Do something with THIS object**
   - Most common - when you need to work with object data.
2. **Class Method: Do something with ALL objects of this class**
   - When you need to work with the whole class.
3. **Static Method: Do something unrelated to objects or the class**
   - When the method doesn't need object or class data

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Real World Model

Build a real-world student management system using Python classes.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Lab

1. Create the following variables:
   - name, age, list of skills (min:3), dictionary.
2. Print type of each variable.
3. Add one more skill to the list.
4. Create a class called “Employee”, adding the __init__ with 3 parameters (self,name, department, salary)
5. Create 2 employees and print their data.
6. Add method called “didplay_info” return employee name, department, salary
7. Add method called “year_salry” calculate salary for the year.
8. Use the 2 methods for the same object.
9. Add class variable title = ‘PythonTech’
10. Add class method get company returns the title.
11. Add static method called valid salary returns true if salary greater than or equal 6000.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1