lecture_11

September 27, 2022

1 Lecture 11

[9]:
```
class Students:
    def __init__(self, name, date_of_birth, class_num, subjects):
        self.name = name
        self.date_of_birth = date_of_birth
        self.class_num = class_num
        self.grades = {subject: None for subject in subjects}

    @property
    def subjects(self):
        return self.grades.keys()

    def __repr__(self):
        return f"{self.__class__.__name__}({self.name},{self.
 ↪date_of_birth},{self.class_num},{self.subjects})"
```

[10]:
```
import random
class Teachers:
    def __init__(self, name, subject):
        self.name = name
        self.subject = subject

    def grade(self, student):
        if self.subject not in student.grades:
            raise Exception("This student does not take this subject")
        student.grades[self.subject] = random.randint(1, 5)

    def __repr__(self):
        return f"{self.__class__.__name__}({self.name},{self.subject})"
```

[11]:
```
student1 = Students(
    name="Adam",
    date_of_birth="2022-08-15",
    class_num=3,
    subjects=["math", "physics"],
)
```

```
teacher1 = Teachers(
    name="Leyla",
    subject="math",
)

teacher1.grade(student1)
```

[12]:
```
student1.grades
```

[12]:
```
{'math': 5, 'physics': None}
```

[18]:
```
class Foo:
    def f(self):
        print("Foo.f()")


class Bar:
    def b(self):
        print("Bar.b()")
```

[19]:
```
class Baz(Foo, Bar):
    pass
```

[20]:
```
obj = Baz()
```

[21]:
```
obj.f()
```

```
Foo.f()
```

[22]:
```
obj.b()
```

```
Bar.b()
```

[28]:
```
class Employee:
    def __init__(self, employee_id, name):
        self.id = employee_id
        self.name = name

    def calculate_payroll(self):
        raise NotImplementedError


class SalaryEmployee(Employee):
    def __init__(self, employee_id, name, weekly_salary):
        super().__init__(employee_id, name)
        self.weekly_salary = weekly_salary
```

```
    def calculate_payroll(self):
        return self.weekly_salary


class HourlyEmployee(Employee):
    def __init__(self, employee_id, name, hourly_salary, hours_per_week):
        super().__init__(employee_id, name)
        self.hourly_salary = hourly_salary
        self.hours_per_week = hours_per_week

    def calculate_payroll(self):
        return self.hourly_salary * self.hours_per_week


class CommissionEmployee(SalaryEmployee):
    def __init__(self, id, name, weekly_salary, commission):
        super().__init__(id, name, weekly_salary)
        self.commission = commission

    def calculate_payroll(self):
        fixed = super().calculate_payroll()
        return fixed + self.commission
```

[29]:
```
class Manager(SalaryEmployee):
    def work(self, hours):
        print(f'{self.name} screams and yells for {hours} hours.')

class Secretary(SalaryEmployee):
    def work(self, hours):
        print(f'{self.name} expends {hours} hours doing office paperwork.')

class SalesPerson(CommissionEmployee):
    def work(self, hours):
        print(f'{self.name} expends {hours} hours on the phone.')

class FactoryWorker(HourlyEmployee):
    def work(self, hours):
        print(f'{self.name} manufactures gadgets for {hours} hours.')
```

[32]:
```
employee_1 = Secretary(employee_id=1, name="Adam", weekly_salary=100_000)
```

[33]:
```
employee_1.calculate_payroll()
```

[33]:
```
100000
```

[35]:
```
employee_1.work(5)
```

```
Adam expends 5 hours doing office paperwork.
```

[37]:
```
class PayrollSystem:
    @staticmethod
    def pay_salary(employee):
        return employee.calculate_payroll()
```

[38]:
```
payroll_system = PayrollSystem()
```

[39]:
```
payroll_system.pay_salary(employee_1)
```

[39]:
```
100000
```

[49]:
```
class CompanyBudget:
    def __init__(self, yearly_budget):
        self.yearly_budget = yearly_budget

    def __isub__(self, other):
        self.yearly_budget -= other
```

[50]:
```
budget_for_2022 = CompanyBudget(20_000_000)
```

[51]:
```
class PayrollSystem:
    def __init__(self, budget):
        self.current_budget = budget

    def pay_salary(self, employee):
        salary = employee.calculate_payroll()
        self.current_budget -= salary
        return salary
```

[52]:
```
payroll_system = PayrollSystem(budget_for_2022)
```

[53]:
```
payroll_system.pay_salary(employee_1)
```

[53]:
```
100000
```

[54]:
```
budget_for_2022.yearly_budget
```

[54]:
```
19900000
```

[59]:
```
class Foo:
    def test(self):
        print("testing Foo")


class Bar:
```

```
    def test(self):
        print("testing Bar")


class Baz(Foo, Bar):
    pass
```

[60]:
```
obj = Baz()
```

[61]:
```
obj.test()
```

```
testing Foo
```

[62]:
```
class Baz(Bar, Foo):
    pass
```

[63]:
```
obj = Baz()
obj.test()
```

```
testing Bar
```

[64]:
```
class Test(Bar):
    pass
```

[65]:
```
class Baz(Test, Foo):
    pass
```

[66]:
```
obj = Baz()
```

[67]:
```
obj.test()
```

```
testing Bar
```

[87]:
```
class HourlyEmployee(Employee):
    def __init__(self, employee_id, name, hourly_salary, hours_per_week):
        super().__init__(employee_id, name)
        self.hourly_salary = hourly_salary
        self.hours_per_week = hours_per_week

    def calculate_payroll(self):
        return self.hourly_salary * self.hours_per_week
```

[88]:
```
class TemporarySecratary(HourlyEmployee, Secretary):
    pass
```

[89]:
```
employee_2 = TemporarySecratary(employee_id=2, name="Jack",␣
 ↪hourly_salary=20_000, hours_per_week=5)
```

super

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [89], in <cell line: 1>()
----> 1 employee_2 =␣
 ↪TemporarySecratary(employee_id=2, name="Jack", hourly_salary=20_000, hours_pe _week=5)

Input In [87], in HourlyEmployee.__init__(self, employee_id, name,␣
 ↪hourly_salary, hours_per_week)
      2 def __init__(self, employee_id, name, hourly_salary, hours_per_week):
      3     print(super().__class__.__name__)
----> 4     super().__init__(employee_id, name)
      5     self.hourly_salary = hourly_salary
      6     self.hours_per_week = hours_per_week

TypeError: __init__() missing 1 required positional argument: 'weekly_salary'
```

[90]:
```
TemporarySecratary.__mro__
```

[90]:
```
(__main__.TemporarySecratary,
 __main__.HourlyEmployee,
 __main__.Secretary,
 __main__.SalaryEmployee,
 __main__.Employee,
 object)
```

[123]:
```
class Employee:
    def __init__(self, employee_id, name):
        self.id = employee_id
        self.name = name

    def calculate_payroll(self):
        raise NotImplementedError

class HourlyEmployee(Employee):
    def __init__(self, employee_id, name, hourly_salary, hours_per_week):
        super(Employee).__init__(employee_id, name)
        self.hourly_salary = hourly_salary
        self.hours_per_week = hours_per_week

    def calculate_payroll(self):
        return self.hourly_salary * self.hours_per_week
```

[124]:
```
class TemporarySecratary(HourlyEmployee, Secretary):
    def __init__(self, employee_id, name, hourly_salary, hours_per_week):
```

```
        HourlyEmployee.__init__(self, employee_id, name, hourly_salary,␣
 ↪hours_per_week)
```

[125]:
```
TemporarySecratary.__mro__
```

[125]:
```
(__main__.TemporarySecratary,
 __main__.HourlyEmployee,
 __main__.Employee,
 __main__.Secretary,
 __main__.SalaryEmployee,
 __main__.Employee,
 object)
```

[126]:
```
employee_2 = TemporarySecratary(employee_id=2, name="Jack",␣
 ↪hourly_salary=20_000, hours_per_week=5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [126], in <cell line: 1>()
----> 1 employee_2 =␣
 ↪TemporarySecratary(employee_id=2, name="Jack", hourly_salary=20_000, hours_pe_week=5)

Input In [124], in TemporarySecratary.__init__(self, employee_id, name,␣
 ↪hourly_salary, hours_per_week)
      2 def __init__(self, employee_id, name, hourly_salary, hours_per_week):
----> 3     ␣
 ↪HourlyEmployee.__init__(self, employee_id, name, hourly_salary, hours_per_wee)

Input In [123], in HourlyEmployee.__init__(self, employee_id, name,␣
 ↪hourly_salary, hours_per_week)
     10 def __init__(self, employee_id, name, hourly_salary, hours_per_week):
---> 11     super(Employee).__init__(employee_id, name)
     12     self.hourly_salary = hourly_salary
     13     self.hours_per_week = hours_per_week

TypeError: super() argument 1 must be type, not int
```

[127]:
```
class A:
    def test(self):
        print("testing A")


class B:
    def test(self):
        print("testing B")
```

```
class C(A, B):
    pass
```

[128]:
```
obj = C()
```

[129]:
```
obj.test()
```

```
testing A
```

[131]:
```
C.__mro__
```

[131]: (__main__.C, __main__.A, __main__.B, object)

[136]:
```
class A:
    def test(self):
        print("testing A")


class B:
    def test(self):
        print("testing B")


class C(A):
    def test(self):
        print("testing C")


class D(C, B):
    pass
```

[137]:
```
obj = D()
```

[138]:
```
obj.test()
```

```
testing C
```

[139]:
```
D.__mro__
```

[139]: (__main__.D, __main__.C, __main__.A, __main__.B, object)

[164]:
```
class A:
    def test(self):
        print("testing A")
```

[165]:
```
class B(A):
    pass
```

[166]:
```
class C(A):
    pass
```

[167]:
```
class D(B, C):
    pass
```

[168]:
```
obj = D()
```

[169]:
```
obj.test()
```

```
testing A
```

[170]:
```
D.__mro__
```

[170]:
```
(__main__.D, __main__.B, __main__.C, __main__.A, object)
```

[ ]:
```

```