# ЛАБОРАТОРНАЯ РАБОТА №1.
# АРХИТЕКТУРА ВЫПОЛНЕНИЯ ПРОГРАММ В PYTHON

В данной лабораторной работе исследуются следующие положения, рассмотренные в главе 1:

- переменная как ссылка на объект;
- различие между равенством объектов и их идентичностью;
- влияние изменяемости на поведение программы;
- область видимости;
- механизм исключений.

## 1. ЦЕЛЬ РАБОТЫ

Исследовать архитектуру выполнения программ в Python и закрепить понимание ссылочной модели переменных, изменяемости объектов, области видимости и обработки исключений.

## 2. ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ

В рамках лабораторной работы необходимо:

1) Подготовить рабочую среду (см. Приложение 1).
2) Создать файл `lab1.ipynb`.
3) Выполнить все эксперименты, представленные в пункте 3 «Экспериментальная часть» лабораторной работы.
4) Выполнить исследовательские задания, представленные в пункте 4 «Исследовательские задания» лабораторной работы.
5) Зафиксировать результаты и сформулировать аналитические выводы.
6) Подготовить отчёт и защитить работу.

## 3. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

Каждый эксперимент должен содержать:

- программный код;
- демонстрация программы;
- аналитическое пояснение результата.

### Эксперимент 1. Ссылочная модель переменных

#### Шаг 1. Создание объектов

```
a = 100
b = a
```

```
print("id(a):", id(a))
print("id(b):", id(b))
```

**Шаг 2. Изменение значения**

```
a = 200

print("id(a) после изменения:", id(a))
print("id(b):", id(b))
```

**Вопросы для анализа:**

1) Совпадают ли идентификаторы?
2) Почему после изменения `a` объект изменился?
3) Что фактически хранит переменная?

**Эксперимент 2. Изменяемые объекты и побочные эффекты**

```
numbers = [1, 2, 3]
alias = numbers

print("id(numbers):", id(numbers))
print("id(alias):", id(alias))
```

Изменение списка:

```
numbers.append(4)
print(alias)
```

**Вопросы для анализа:**

1) Почему изменился `alias`?
2) Как создать независимую копию списка?
3) Проверить через `id()` поведение копии.

**Эксперимент 3. Передача аргументов в функцию**

```
def modify_list(lst):
    lst.append(999)

data = [10, 20]
modify_list(data)


print(data)
```

**Вопросы и задания для анализа:**

1) Почему список изменился?
2) Изменится ли число при аналогичном эксперименте?
3) Сделать эксперимент с типом `int`.

**Эксперимент 4. Область видимости**

```
x = 5

def test():
    x = 10
    print("Внутри функции:", x)

test()
print("Снаружи:", x)
```

**Задание для анализа:** Добавить `global x` и сравнить поведение.

**Эксперимент 5. Исключения и устойчивость программы**

Создать программу:

```
while True:
    try:
        value = int(input("Введите число: "))
        break
    except ValueError:
        print("Ошибка ввода.")

print("Квадрат:", value ** 2)
```

**Вопросы и задания для анализа:**

1) Что произойдёт без `try-except`?
2) Почему программа становится устойчивой?
3) Как можно расширить обработку ошибок?

## 4. ИССЛЕДОВАТЕЛЬСКИЕ ЗАДАНИЯ

**Задание 1. Исследование различий между `==` и `is`.**

1.1 Создать несколько объектов одинакового значения.

1.2 Сравнить их с помощью ==.

1.3 Сравнить их с помощью is.

1.4 Объяснить различие.

**Задание 2. Исследование интернирования целых чисел.**

Пример исследования – Проверить поведение:

```
a = 256
b = 256
print(a is b)
```

```
a = 257
b = 257
print(a is b)
```

Проанализировать результат.

**Задание 3. Исследование поверхностного копирования объектов.**
Пример исследования – Сравнить:

```
a = [1, 2]
b = a
c = a.copy()
```

Проанализировать различия в поведении при изменении `a`.

**Задание 4. Исследование глубокого копирования вложенных структур:**

- реализовать пример;
- использовать модуль `copy;`
- сравнить `id()` вложенных элементов;
- сформулировать вывод о механизме хранения объектов в памяти.

Пример исследования– Создать список списков:

```
matrix = [[1, 2], [3, 4]]
copy_matrix = mat rix.copy()
```

Изменить элемент:

```
matrix[0][0] = 999
```

**Проанализировать:**

1) Почему изменилась копия?
2) В чём отличие поверхностного и глубокого копирования?
3) Реализовать глубокую копию с использованием модуля `copy.`
4) Сравнить `id()` внутренних объектов.

Сформулировать вывод о механизме хранения вложенных структур.

## 5. ЗАДАНИЯ К САМОСТОЯТЕЛЬНОЙ РАБОТЕ ОБУЧАЮЩЕГОСЯ (СРО)

1) Объяснить, что означает утверждение «в Python всё является объектом».
2) В чём различие между равенством значений и равенством объектов?
3) Почему понимание изменяемости критично для анализа данных?
4) В каких случаях использование `global` является нежелательным?

5) Что такое устойчивость программы к ошибкам?

## 6. ТРЕБОВАНИЯ К ОТЧЁТУ

Отчёт по лабораторной работе должен содержать:

- цель работы;
- описание каждого эксперимента;
- описание исследований;
- программный код с комментариями;
- результаты выполнения;
- аналитические выводы по каждому эксперименту и исследованию;
- обобщающий вывод о механизме выполнения программ в Python.

В заключительной части отчёта рекомендуется представить анализ следующих положений:

1) Что хранит переменная в Python – значение или ссылку на объект?
2) В чём различие между операторами `==` и `is`?
3) Почему изменяемость объектов влияет на корректность программ анализа данных?
4) В чём заключается риск неявного изменения объектов?
5) Как связаны область видимости переменных и надёжность программы?

Работа защищается с демонстрацией выполнения экспериментов и полученных результатов.

## 7. РЕКОМЕНДУЕМЫЕ ИСТОЧНИКИ

1. Русскоязычная документация по Python: https://pylessons.readthedocs.io/ru/latest/
2. Документация по языку Python3. https://docs-python.ru/
3. Документация Jupyter: https://docs.jupyter.org/
4. Справочник по виртуальным окружениям Python: https://docs.python.org/3/library/venv.html