**Создание калькулятора с использованием Python и библиотеки PyQt5**

* **Создаем новый проект в PyCharm с названием Calculator**
* **Устанавливаем библиотеку для создания оконных приложений PyQt5**

pip3 install PyQt5

* **В проекте создаем новый файл с названием calculator.py**
* **Импортируем необходимые библиотеки для создания приложения**

import sys

from PyQt5.QtWidgets import QApplication, QWidget, QLineEdit, QHBoxLayout, QVBoxLayout, QPushButton

Библиотека sys используется для получения информации об операционной системе.

Библиотека PyQt5 используется для создания оконных приложений.

PyQt5 используется в нашем проекте не полностью для уменьшения объема зависимостей приложения. Поэтому мы импортируем лишь некоторые классы из нее.

QApplication – управляет потоком управления и основными настройками приложения с графическим интерфейсом

QWidget – является базовым классом для всех объектов пользовательского интерфейса

QLineEdit – виджет, который разрешает вводить и редактировать одну строку текста

QHBoxLayout – выстраивает виджеты по горизонтали

QVBoxLayout – выстраивает виджеты по вертикали

QPushButton – кнопка, на которую можно нажимать

* **Создаем класс Calculator и наследуем его от класса QWidgets**

class Calculator(QWidget):

def \_\_init\_\_(self):

super(Calculator, self).\_\_init\_\_()

* **Внутри конструктора создаем оси выравнивания**

self.vbox = QVBoxLayout(self)

self.hbox\_input = QHBoxLayout()

self.hbox\_first = QHBoxLayout()

self.hbox\_result = QHBoxLayout()

self.vbox.addLayout(self.hbox\_input)

self.vbox.addLayout(self.hbox\_first)

self.vbox.addLayout(self.hbox\_result)

Вертикальная ось будет главной в окне.

К ней привязываем горизонтальные оси выравнивания с помощью функции addLayout()

* **В конструкторе создаем виджеты и привязываем их к соответствующим осям выравнивания**

self.input = QLineEdit(self)

self.hbox\_input.addWidget(self.input)

self.b\_1 = QPushButton("1", self)

self.hbox\_first.addWidget(self.b\_1)

self.b\_2 = QPushButton("2", self)

self.hbox\_first.addWidget(self.b\_2)

self.b\_3 = QPushButton("3", self)

self.hbox\_first.addWidget(self.b\_3)

self.b\_plus = QPushButton("+", self)

self.hbox\_first.addWidget(self.b\_plus)

self.b\_result = QPushButton("=", self)

self.hbox\_result.addWidget(self.b\_result)

Привязка виджетов к осям осуществляется с помощью функции addWidget()

* **Создаем события, отвечающие за реакции на нажатия по кнопкам**

self.b\_plus.clicked.connect(lambda: self.\_operation("+"))

self.b\_result.clicked.connect(self.\_result)

self.b\_1.clicked.connect(lambda: self.\_button("1"))

self.b\_2.clicked.connect(lambda: self.\_button("2"))

self.b\_3.clicked.connect(lambda: self.\_button("3"))

Функция connect(<имя\_функции/метода>), вызывает функцию/метод с именем указанным в аргументах. В указанную функцию/метод нельзя передавать аргументы. Для решения этой проблемы используем lambda-функции.

* **Создаем метод класса для обработки кнопок, отвечающих за ввод цифр в линию ввода текста**

def \_button(self, param):

line = self.input.text()

self.input.setText(line + param)

Уже существующая строка в линии ввода конкатенируется с аргументом param и устанавливается как отображаемый в линии ввода текст.

* **Создаем метод класса для обработки нажатия на кнопку математической операции**

def \_operation(self, op):

self.num\_1 = int(self.input.text())

self.op = op

self.input.setText("")

Запоминаем первое введенное число в целочисленном типе данных.

Запоминаем в качестве операции аргумент op.

Очищаем линию ввода.

* **Создаем метод класса для обработки нажатия на кнопку результата**

def \_result(self):

self.num\_2 = int(self.input.text())

if self.op == "+":

self.input.setText(str(self.num\_1 + self.num\_2))

Запоминаем второе введенное число в целочисленном типе данных.

Производим вычисление в зависимости от операции и устанавливаем его в качестве текста в линию ввода.

* **Запускаем приложение**

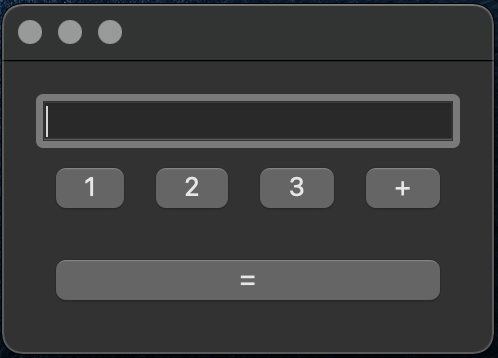
app = QApplication(sys.argv)

win = Calculator()

win.show()

sys.exit(app.exec\_())

Приложение должно выглядеть подобным образом:



**Домашнее задание:**

* Обработать все возможные исключения
* Добавить кнопку для добавления плавающей точки
* Добавить кнопки для математических операций вычитания, умножения, деления
* Создать для этих кнопок методы-обработчики