```
def iv_black_scholes(price: Union[float, np.ndarray], S: float, K:
Union[float, np.ndarray], t: Union[float, np.ndarray], r: float,
is_call: bool):
```

Описание:

Эта функция вычисляет подразумеваемую волатильность из цен заданных опционов.
Вход:

- массив цен опционов
- спот
- массив страйков
- массив остатков времени до срока экспирации
- базовая ставка рефинансирования (безрисковая процентная ставка)
- булевая переменная: call или put опцион

Происходит обращение к внешней функции `implied_volatility_vec()`, она описана в 5 строке `from py_vollib.black_scholes.implied_volatility import implied_volatility` (обращение к библиотеке)
Вывод:
Подразумеваемая волатильность в виде массива (в десятичной форме, например 0.20 для 20%).

Raises:
ValueError: Если входные параметры имеют несовместимые формы (например, price и K — массивы разной длины).

```
def heston_cf(s, v, kappa, theta, sigma, rho, u, t):
    """Characteristic function of the log-price in the Heston model."""
    ///
  return cmath.exp(C + D * v + u * math.log(s) * 1j)
```

Характеристическая функция логарифмической цены в модели стохастической волатильности Хестона.
Здесь и далее `1j` – мнимая единица
Вычисляет характеристическую функцию φ(u) для логарифмического процесса цены ln(S_t) в модели Хестона, которая используется для ценообразования опционов с помощью методов преобразования Фурье.

Args:
s (float): Текущая (честнее сказать, начальная) цена актива (S_0).
v (float): Начальная дисперсия (v_0).
kappa (float): Средняя скорость возврата дисперсии (κ).
theta (float): Долгосрочная средняя дисперсия (θ).
sigma (float): Волатильность дисперсии (σ).
rho (float): Корреляция между ценой актива и дисперсией (ρ).
u (float): Аргумент характеристической функции (часто связанный с переменной преобразования Фурье).

t (float): Время до погашения опциона в годах (T).

Returns:
комплексное число: Значение характеристической функции φ(u) в точке u.

Дополнительные рекомендации:
Для полной реализации модели Хестона стоит добавить:
[1] Heston, S. L. (1993). A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. The Review of Financial Studies, 6(2), 327-343.
Если функция будет использоваться в численных методах, можно добавить предупреждение:

Warning:
Для |u| > 100 происходит потеря точности из-за работы с числами с плавающей точкой.

Стоит обратить внимание на опасные значения переменных:
ValueError: If sigma <= 0 or |rho| > 1 or v <= 0

```
def heston_cf_copy(s, v, kappa, theta, sigma, rho, u, t):
    """Characteristic function of the log-price in the Heston model."""
    d = cmath.sqrt((rho * sigma * u * 1j - kappa) ** 2 + sigma ** 2 *
(u * 1j + u ** 2))
    g = ((rho * sigma * u * 1j - kappa + d) / (rho * sigma * u * 1j -
kappa - d))
    C = (kappa * theta / sigma ** 2 * (
        t * (kappa - rho * sigma * u * 1j - d) -
        2 * cmath.log((1 - g * cmath.exp(-d * t)) / (1 - g))))
    D = ((kappa - rho * sigma * u * 1j - d) / sigma ** 2 *
         ((1 - cmath.exp(-d * t)) / (1 - g * cmath.exp(-d * t))))
    return cmath.exp(C + D * v + u * math.log(s) * 1j), d, g, C, D,
math.log(s)
```

Отличие от `heston_cf` в дополнительном выводе внутренних параметров, кроме `cmath.exp(C + D * v + u * math.log(s) * 1j)` выводятся:
d – дискриминант характеристического уравнения в модели Хестона
g – отношение, которое определяет "вес" экспоненциальных компонентов в решении дифференциального уравнения для волатильности. По сути, это коэффициент, управляющий затуханием колебаний.
C – часть характеристической функции, отвечающая за временную эволюцию процесса. Включает линейный по времени член (t * ...) и логарифмическую поправку, связанную с релаксацией волатильности к долгосрочному уровню theta.

D – коэффициент, определяющий зависимость от начальной волатильности v. Управляет "переходом" между текущей волатильностью (v) и долгосрочной (theta).
math.log(s) – Логарифм начальной цены актива ($\ln(S_0)$). Используется для связи характеристической функции с лог-ценой.

```
def heston_integrand(u, t, k,  s, r, v, kappa, theta, sigma, rho):
    """Integrand in Heston's formula."""
    return (cmath.exp(-1j * u * math.log(k)) / (1j * u) *
            (cmath.exp(r * t) * heston_cf(s, v, kappa, theta, sigma,
rho, u - 1j, t) -
             k * heston_cf(s, v, kappa, theta, sigma, rho, u, t))).real
```

"""Вычисляет подынтегральное выражение для ценообразования опционов посредством обратного преобразования Фурье в модели Хестона.

Эта функция представляет собой основное подынтегральное выражение, используемое в замкнутом решении Хестона для
европейских цен опционов, которое требует численного интегрирования этого выражения.

Аргументы:
u (float): Переменная Фурье (интегральная переменная).
t (float): Срок до погашения опциона в годах.
k (float): Цена исполнения опциона.
s (float): Текущая (начальная) цена актива (спот).
r (float): Безрисковая процентная ставка (годовая, непрерывное начисление процентов).
v (float): Начальная дисперсия (квадрат волатильности).
kappa (float): Средняя скорость возврата дисперсии.
theta (float): Долгосрочная средняя дисперсия.
sigma (float): Волатильность дисперсии.
rho (float): Корреляция между ценой актива и дисперсией.

Возвращаем:
число с плавающей точкой: действительную часть от комплексного подынтегрального значения в точке u.

...

```
def heston_scalar(t: float, k: float, s: float, r: float, is_call:
bool, heston_params: HestonParams) -> float:
    call_price = (0.5 * (s - math.exp(- r * t) * k) +
                  1 / math.pi * math.exp(- r * t) *
                  intg.quad(
                    heston_integrand,
                    0, math.inf,
```

```
                    args=(t, k, s, r, heston_params.v,
heston_params.kappa, heston_params.theta,
                    heston_params.sigma, heston_params.rho))[0])
    if is_call:
        return call_price

    return call_price + k * np.exp(- r * t) - s
```

Функция heston_scalar вычисляет цену опциона (колл или пут) по модели Хестона с использованием полуаналитической формулы. Это вспомогательная функция, работающая со скалярными (!!) значениями времени и страйка.

Параметры
t (float): Время до экспирации опциона (в годах)
k (float): Страйк-цена опциона
s (float): Текущая цена базового актива
r (float): Безрисковая процентная ставка (годовая)
is_call (bool): Флаг типа опциона:
True - колл опцион
False - пут опцион

heston_params (HestonParams): Объект с параметрами модели Хестона:
v - начальная волатильность
kappa - скорость возврата к среднему
theta - долгосрочная волатильность
sigma - волатильность волатильности
rho - корреляция между движением цены и волатильностью

Возвращаемое значение
float: Расчетная цена опциона указанного типа
Для пут-опциона цена пересчитывается согласно паритету пут-колл

```
def heston(t: Union[float, np.ndarray],
           k: Union[float, np.ndarray],
           s: float, r: float,
           is_call: Union[bool, np.ndarray],
           heston_params: HestonParams) -> Union[float, np.ndarray]:
    """Computes the price of a call option by Heston's semi-closed
formula."""
    b = np.broadcast(t, k)
    if b.nd:  # Vector arguments were supplied
        return np.fromiter(
```

```
            (heston_scalar(t_, k_, s, r, is_call, heston_params) for
(t_, k_) in b),
            count=b.size, dtype=float).reshape(b.shape)
    else:
        # Исправленная строка: вызов scalar-функции вместо рекурсии
        return heston_scalar(t, k, s, r, is_call, heston_params)
```

Параметры
t (массив): Времена до экспирации опциона (в годах)
k (массив): Страйк-цены опционов
s (float): Текущая цена базового актива
r (float): Безрисковая процентная ставка (годовая)
is_call (bool): Флаг типа опциона:
True - колл опцион
False - пут опцион

heston_params (HestonParams): Объект с параметрами модели Хестона:
v - начальная волатильность
kappa - скорость возврата к среднему
theta - долгосрочная волатильность
sigma - волатильность волатильности
rho - корреляция между движением цены и волатильностью

Возвращаемое значение
массив: Расчетные цены опционов указанного типа (обращение к `heston_scalar`)
Для пут-опциона цена пересчитывается согласно паритету пут-колл

```
def objective(p, t, k, s, r, iv):
    prices_heston = np.zeros((t.size, k.size))
    ivs_bs = np.zeros((t.size, k.size))
    for i, strike in enumerate(k):
        is_call = True if strike > s*math.exp(-r*t) else False
        prices_heston[:, i] = heston(t=times, k=strike, s=s, r=r,
is_call=is_call, heston_params=HestonParams(*p))
        ivs_bs[:, i] = iv_black_scholes(price=prices_heston[:, i], S=s,
K=strike, t=t, r=r, is_call=is_call)

    return np.linalg.norm(ivs_bs - iv)
```

Функция objective вычисляет разницу между подразумеваемой волатильностью (IV), полученной из модели Хестона, и заданными значениями подразумеваемой волатильности.
Параметры
p (tuple/list/array): Вектор параметров модели Хестона в следующем порядке:
v0: Начальная волатильность
vT: Долгосрочная волатильность
rho: Корреляция между движением цены и волатильностью
k: Скорость возврата к среднему
sigma: Волатильность волатильности
t (numpy.array): Массив времен до экспирации (в годах)
k (numpy.array): Массив страйк-цен
s (float): Текущая цена базового актива
r (float): Безрисковая процентная ставка (годовая)
iv (numpy.array): Массив рыночных значений подразумеваемой волатильности (размерность должна соответствовать t.size × k.size)

Возвращаемое значение
float: Евклидова норма (L2-норма) разницы между вектором расчетных значений IV из модели Хестона и вектором рыночных значений IV

```
def calibrate(t: Union[float, np.ndarray], k: np.ndarray,
              iv: np.ndarray, s: float, r: float = 0, min_method: str =
"L-BFGS-B"):
          v0 = iv[np.abs(k-s).argmin()]**2  # ATM variance
        res = optimize.minimize(
            fun=objective,
            x0=(v0, 1.0, v0, 1.0, -0.5),  # (v, kappa, theta, sigma,
rho)
            method=min_method,
            args=(t, k, s, r, iv),
            bounds=[(0, math.inf), (0, math.inf), (0, math.inf), (0,
math.inf),
                    (-1, 1)])
        return res
```

Функция calibrate выполняет калибровку параметров модели Хестона, подбирая такие значения параметров, которые минимизируют разницу между рыночными значениями подразумеваемой волатильности и значениями, полученными из модели Хестона.

Параметры
t (float или np.ndarray): Время до экспирации опциона (в годах). Может быть скаляром или массивом.

k (np.ndarray): Массив страйк-цен опционов.
iv (np.ndarray): Массив рыночных значений подразумеваемой волатильности.
s (float): Текущая цена базового актива.
r (float, optional): Безрисковая процентная ставка (годовая). По умолчанию 0.
min_method (str, optional): Метод оптимизации, используемый в scipy.optimize.minimize.
Должен поддерживать ограничения (bounds). По умолчанию "L-BFGS-B".
Возвращаемое значение
OptimizeResult: Объект результата оптимизации из scipy.optimize.minimize, содержащий:
x (ndarray): Оптимальные параметры модели Хестона в порядке [v, kappa, theta, sigma, rho].