```
import numpy as np
import re

"""
1. Extraer el nombre de un archivo de una trayectoria del sistema de archivos
"""

string = "$HOME/proyecto1/modulo5/programa3.py"
print(re.search(r'[^/]+\.[A-Za-z/d]+', string).group(0))

"""
2. Escribir la función date_in_spanish. Use re.sub para sustituir los nombres
de los meses.
"""


def date_in_spanish(date):
    """
    Translates a string date to spanish. That is, all references to months
    abbreviations like 'Jan', 'Feb', 'Mar' and so on are changed to 'Ene',
    'Feb', 'Mar', respectively.

    Parameters
    ----------
    date : str
        Date to be translated.

    Returns
    ------
        str
        The translated base_date.

    Examples
    --------
    >>> date_in_spanish("23-Apr-2021")
    23-Abr-2021
    >>> date_in_spanish("Dec-24-2020")
    Dic-24-2020
    """
    change_date = {"Jan": "Ene", "Apr": "Abr", "Aug": "Ago", "Dec": "Dic"}
    for key in change_date.keys():
        if re.search(key, date) is not None:
            return re.sub(key, change_date[key], date)
    return date


""""
3. Escribir la siguiente función
"""


def from_standard_equity_option_convention(code: str) -> dict:
    """
    Transform a standard equity option convention code to record representation.

    Parameters
    ----------
    code : str
        Standard equity option convention code (see
        https://en.wikipedia.org/w. Programar el método siguiente.
    Returns
    -------
        dict
        A dictionary containing:
        'symbol': Symbol name
        'expire': Option expiration base_date
        'right': Put (P) or Call (C).
        'strike': Option strike

    Examples:
    >>> from_standard_equity_option_convention('YHOO150416C00030000')
    {'symbol': 'YHOO', 'expire': '20150416', 'right': 'C', 'strike': 30.0}
    """
    a = re.findall("[A-Z]+", code)

    b = re.findall(r"\d+", code)

    symbol = a[0]
    right = a[1]
    expire = "20"+b[0]
    strike = int(b[1]) / 1000

    return {"symbol": symbol, "expire": expire, "right": right, "strike": strike}


print(from_standard_equity_option_convention("YHOO150416C00030000"))

"""
4. Explique con palabras qué hace la siguiente instrucción
symbols_str = re.sub(r"'", "''", str(symbols))

Cada comilla simple (') de la cadena str(symbols) se sustituye por ('').
"""


"""
5. Escriba una cadena 'account' apropiada para que se ejecute la instrucción print
if re.match(r'DU[0-9]{7}', account):
    print("Account: ", account)
"""
account = "DU1234567"
if re.match(r'DU[0-9]{7}', account):
    print("Account: ", account)


"""
6. Escriba la expresión regular de manera más sintética pero preservando la funcionalidad.
if re.match('^([0-9][0-9][0-9][0-9][0-9][0-9])$', text):
    LOGGER.info("Correct OTP format: %s.", text)

if re.match('^(\\d{6})$', text):
    LOGGER.info("Correct OTP format: %s.", text)
"""


"""
7. ¿Cuál es el valor de 'reg_exp' que hace funcionar el código siguiente?
if re.match(reg_exp, text) is None:
    error_message = \
        "Try again, your answer does not correspond to a comma " + \
        "separated integers list. Type something like '1, 2, 3' " + \
        "without the apostrophes."
```

```
Un posible valor para reg_exp que hace funcionar el codigo es por ejemplo reg_exp = "abcd"
"""


"""
8.Programar el método siguiente.
"""


def collect_commission_adjustment(data):
    """
    Retrieve a commision adjustment record from the section "Commission
    Adjustments" in one Interactive Brokers activity report.

    PARAMETERS
    ----------
    data : list[]
        Line from the activity report in the "Commission Adjustment" section
        in list format. That is, each element in the list is a comma
        separated item from the line.

    RETURNS
    -------
        dict
        Containing the open position information in dictionary format.

    Examples
    --------
    >>> collect_commission_adjustment(['Commission Adjustments', 'Data', 'USD',
    ... '2021-04-23',
    ... 'Commission Computed After Trade Reported (C210430C00069000)',
    ... '-1.0906123', '\\n'])
    {'end_date': '20210423', 'symbol': 'C', 'expire': '20210430', \
    'right': 'C', 'strike': 69.0, 'sectype': 'OPT', 'amount': -1.0906123}
    >>> collect_commission_adjustment(
    ... ['Commission Adjustments', 'Data', 'USD', '2021-02-19',
    ... 'Commission Computed After Trade Reported (ALB)', '-0.4097', '\\n'])
    {'end_date': '20210219', 'symbol': 'ALB', 'sectype': 'STK', \
    'amount': -0.4097}
    >>> collect_commission_adjustment(
    ... ['Commission Adjustments', 'Data', 'USD', '2021-02-19',
    ... 'Commission Computed After Trade Reported (ALB)', '-0.4097', '\\n'])
    {'end_date': '20210219', 'symbol': 'ALB', 'sectype': 'STK', \
    'amount': -0.4097}
    """
    dat = str(data)
    reg_exp = "[A-Z]+[0-9]{6}[CP][0-9]{8}"
    opt_stk = re.search(reg_exp, dat)

    date = re.search("[0-9]{4}-[0-9]{2}-[0-9]{2}", dat).group(0)
    end_date = re.sub("-", "", date)
    reg_amount = r"[-][0-9]+\.[0-9]+"
    amount = float(re.search(reg_amount, dat).group(0))

    if opt_stk is not None:
        opt_conv_letters = re.findall("[A-Z]+", opt_stk.group(0))
        opt_conv_numbers = re.findall("[0-9]+", opt_stk.group(0))
        symbol = opt_conv_letters[0]
        expire = end_date[0]+end_date[1] + opt_conv_numbers[0]
        right = opt_conv_letters[1]
        strike = int(opt_conv_numbers[1])/1000
        sectype = "OPT"

        return {"end_date": end_date, "symbol": symbol, "expire": expire, "right": right,
"strike": strike,
                "sectype": sectype, "amount": amount}

    else:
        reg_stk = "[(][A-Z]+[)]"
        sym = re.search(reg_stk, dat).group(0)
        symbol = re.sub("[()]", "", sym)
        sectype = "STK"
        return {"end_date": end_date, "symbol": symbol, "sectype": sectype, "amount": amount}


data = ['Commission Adjustments', 'Data', 'USD', '2021-04-23',
        'Commission Computed After Trade Reported (C210430C00069000)', '-1.0906123', '\\n']

data2 = ['Commission Adjustments', 'Data', 'USD', '2021-02-19', 'Commission Computed After
Trade Reported (ALB)',
         '-0.4097', '\\n']

print(collect_commission_adjustment(data))


"""
9. De dos ejemplos de uso del siguiente método. En el primero el método debe
regresar un número de punto flotante y en el segundo np.nan
"""


def banxico_value(tag, data):
    """
    Get data values from Banxico portals.
    Parameters
    ----------
    tag : str
        Internal tag name of the variable to retrieve.
    data : str
        Html page to locate the tag value.

    Returns
    --------
        float
        The associated tag value.
    """
    float_nt = "[^0-9-]*([-]*[0-9]+.[0-9]+)[^0-9]"
    try:
        res = float(re.search(tag + float_nt, data).group(1))
    except AttributeError:
        res = np.nan
    return res


print(banxico_value("23", "file:///home/daniel/23-193.html"))
print(banxico_value("00", "file:///home/daniel/23-193.html"))


"""
10. Describa en palabras qué hace el siguiente código.
"""
dat_dfcolumns = ["Imf0imf", "iMF90", "imF67", "da"]
```

```
col_sel = list(
        map(
            lambda s: s if re.match("[Ii][Mm][Ff][0-9]+", s) else None,
            dat_dfcolumns,
        ))
col_sel = [c for c in col_sel if c is not None]

"""
para cada elemento s del iterable dat_df.columns se comprueba si s comienza con la letra I o
i, seguida de
la letra M o m, seguida de la letra F o f, y seguido de uno o más números enteros entre el 0 y
el 9,
en caso de que así sea, s se agrega a la lista que lleva por nombre col_sel, en caso contrario
se agrega None.
Posteriormente se actualiza la lista col_sel unicamente con los elementos s que cumple la
condición
descrita anteriormente.
"""
```