

Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación.

IC5701 – Compiladores e Intérpretes.

Proyecto 2 Explorador – Ciruelas Extendido

Profesor: Aurelio Sanabria.

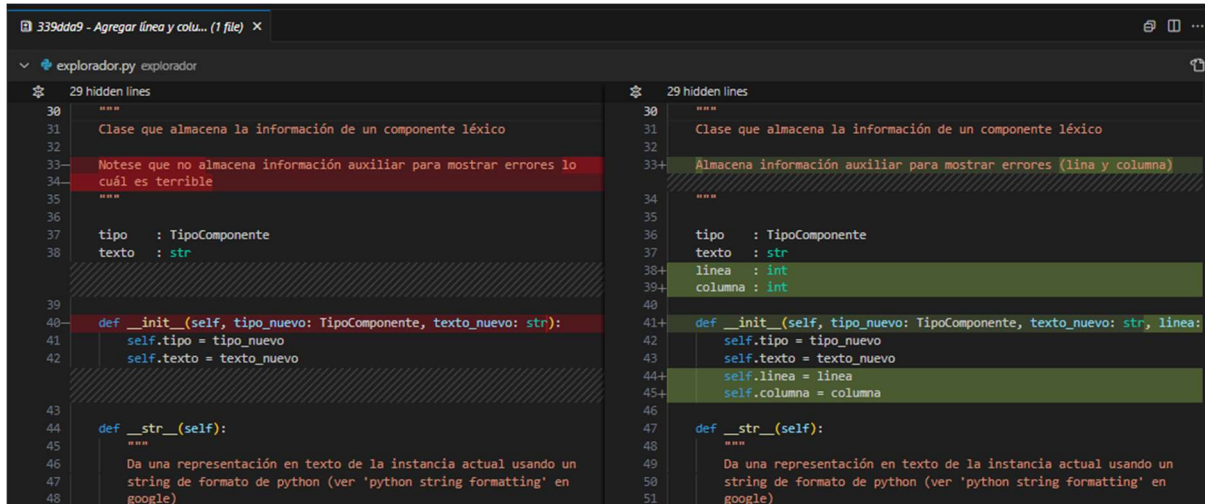
Estudiante:

Tamara Nicole Rodríguez Luna.

19 de abril del 2025.

Demostración y discusión

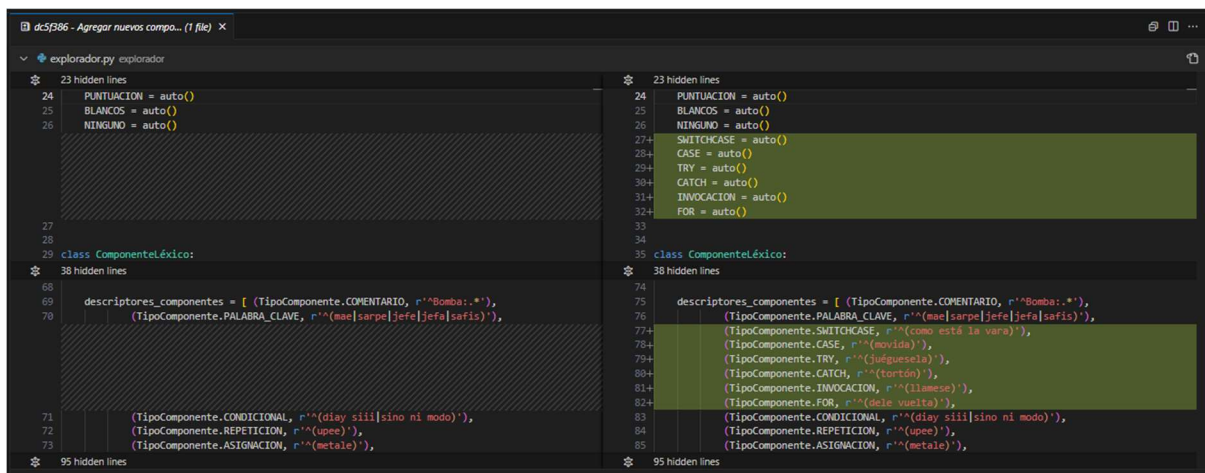
En Ciruelas el explorador no almacenaba la línea y columna de los componentes, por lo tanto, los agregué para poder mostrar errores más precisos, de paso adaptando el comentario para que esté acorde con el código



```
30 """
31 Clase que almacena la información de un componente léxico
32
33- Notese que no almacena información auxiliar para mostrar errores lo
34- cuál es terrible
35 """
36
37 tipo : TipoComponente
38 texto : str
39
40- def __init__(self, tipo_nuevo: TipoComponente, texto_nuevo: str):
41- self.tipo = tipo_nuevo
42- self.texto = texto_nuevo
43
44- def __str__(self):
45- """
46- Da una representación en texto de la instancia actual usando un
47- string de formato de python (ver 'python string formatting' en
48- google)
49- """
50- return self.texto
```

```
30 """
31 Clase que almacena la información de un componente léxico
32
33+ Almacena información auxiliar para mostrar errores (línea y columna)
34 """
35
36 tipo : TipoComponente
37 texto : str
38+ línea : int
39+ columna : int
40
41+ def __init__(self, tipo_nuevo: TipoComponente, texto_nuevo: str, línea:
42+ self.tipo = tipo_nuevo
43+ self.texto = texto_nuevo
44+ self.línea = línea
45+ self.columna = columna
46
47+ def __str__(self):
48+ """
49+ Da una representación en texto de la instancia actual usando un
50+ string de formato de python (ver 'python string formatting' en
51+ google)
52+ """
53+ return self.texto
```

Seguidamente añadí los nuevos componentes léxicos de la gramática que definí para ciruelas



```
24 PUNTUACION = auto()
25 BLANCOS = auto()
26 NINGUNO = auto()
27
28
29 class ComponenteLéxico:
30
31- descriptors_componentes = { (TipoComponente.COMENTARIO, r'^Bomba:.*'),
32- (TipoComponente.PALABRA_CLAVE, r'^mae[sarpe]jefe[jefa[safis])'),
33-
34-
35- (TipoComponente.CONDICIONAL, r'^diay siil[sino ni modo]'),
36- (TipoComponente.REPETICION, r'^upee)'),
37- (TipoComponente.ASIGNACION, r'^metale)'),
38-
39-
40-
41-
42-
43-
44-
45-
46-
47-
48-
49-
50-
51-
52-
53-
54-
55-
56-
57-
58-
59-
60-
61-
62-
63-
64-
65-
66-
67-
68-
69-
70-
71-
72-
73-
74-
75-
76-
77-
78-
79-
80-
81-
82-
83-
84-
85-
86-
87-
88-
89-
90-
91-
92-
93-
94-
95-
96-
97-
98-
99-
100-
101-
102-
103-
104-
105-
106-
107-
108-
109-
110-
111-
112-
113-
114-
115-
116-
117-
118-
119-
120-
121-
122-
123-
124-
125-
126-
127-
128-
129-
130-
131-
132-
133-
134-
135-
136-
137-
138-
139-
140-
141-
142-
143-
144-
145-
146-
147-
148-
149-
150-
151-
152-
153-
154-
155-
156-
157-
158-
159-
160-
161-
162-
163-
164-
165-
166-
167-
168-
169-
170-
171-
172-
173-
174-
175-
176-
177-
178-
179-
180-
181-
182-
183-
184-
185-
186-
187-
188-
189-
190-
191-
192-
193-
194-
195-
196-
197-
198-
199-
200-
201-
202-
203-
204-
205-
206-
207-
208-
209-
210-
211-
212-
213-
214-
215-
216-
217-
218-
219-
220-
221-
222-
223-
224-
225-
226-
227-
228-
229-
230-
231-
232-
233-
234-
235-
236-
237-
238-
239-
240-
241-
242-
243-
244-
245-
246-
247-
248-
249-
250-
251-
252-
253-
254-
255-
256-
257-
258-
259-
260-
261-
262-
263-
264-
265-
266-
267-
268-
269-
270-
271-
272-
273-
274-
275-
276-
277-
278-
279-
280-
281-
282-
283-
284-
285-
286-
287-
288-
289-
290-
291-
292-
293-
294-
295-
296-
297-
298-
299-
300-
301-
302-
303-
304-
305-
306-
307-
308-
309-
310-
311-
312-
313-
314-
315-
316-
317-
318-
319-
320-
321-
322-
323-
324-
325-
326-
327-
328-
329-
330-
331-
332-
333-
334-
335-
336-
337-
338-
339-
340-
341-
342-
343-
344-
345-
346-
347-
348-
349-
350-
351-
352-
353-
354-
355-
356-
357-
358-
359-
360-
361-
362-
363-
364-
365-
366-
367-
368-
369-
370-
371-
372-
373-
374-
375-
376-
377-
378-
379-
380-
381-
382-
383-
384-
385-
386-
387-
388-
389-
390-
391-
392-
393-
394-
395-
396-
397-
398-
399-
400-
401-
402-
403-
404-
405-
406-
407-
408-
409-
410-
411-
412-
413-
414-
415-
416-
417-
418-
419-
420-
421-
422-
423-
424-
425-
426-
427-
428-
429-
430-
431-
432-
433-
434-
435-
436-
437-
438-
439-
440-
441-
442-
443-
444-
445-
446-
447-
448-
449-
450-
451-
452-
453-
454-
455-
456-
457-
458-
459-
460-
461-
462-
463-
464-
465-
466-
467-
468-
469-
470-
471-
472-
473-
474-
475-
476-
477-
478-
479-
480-
481-
482-
483-
484-
485-
486-
487-
488-
489-
490-
491-
492-
493-
494-
495-
496-
497-
498-
499-
500-
501-
502-
503-
504-
505-
506-
507-
508-
509-
510-
511-
512-
513-
514-
515-
516-
517-
518-
519-
520-
521-
522-
523-
524-
525-
526-
527-
528-
529-
530-
531-
532-
533-
534-
535-
536-
537-
538-
539-
540-
541-
542-
543-
544-
545-
546-
547-
548-
549-
550-
551-
552-
553-
554-
555-
556-
557-
558-
559-
560-
561-
562-
563-
564-
565-
566-
567-
568-
569-
570-
571-
572-
573-
574-
575-
576-
577-
578-
579-
580-
581-
582-
583-
584-
585-
586-
587-
588-
589-
590-
591-
592-
593-
594-
595-
596-
597-
598-
599-
600-
601-
602-
603-
604-
605-
606-
607-
608-
609-
610-
611-
612-
613-
614-
615-
616-
617-
618-
619-
620-
621-
622-
623-
624-
625-
626-
627-
628-
629-
630-
631-
632-
633-
634-
635-
636-
637-
638-
639-
640-
641-
642-
643-
644-
645-
646-
647-
648-
649-
650-
651-
652-
653-
654-
655-
656-
657-
658-
659-
660-
661-
662-
663-
664-
665-
666-
667-
668-
669-
670-
671-
672-
673-
674-
675-
676-
677-
678-
679-
680-
681-
682-
683-
684-
685-
686-
687-
688-
689-
690-
691-
692-
693-
694-
695-
696-
697-
698-
699-
700-
701-
702-
703-
704-
705-
706-
707-
708-
709-
710-
711-
712-
713-
714-
715-
716-
717-
718-
719-
720-
721-
722-
723-
724-
725-
726-
727-
728-
729-
730-
731-
732-
733-
734-
735-
736-
737-
738-
739-
740-
741-
742-
743-
744-
745-
746-
747-
748-
749-
750-
751-
752-
753-
754-
755-
756-
757-
758-
759-
760-
761-
762-
763-
764-
765-
766-
767-
768-
769-
770-
771-
772-
773-
774-
775-
776-
777-
778-
779-
780-
781-
782-
783-
784-
785-
786-
787-
788-
789-
790-
791-
792-
793-
794-
795-
796-
797-
798-
799-
800-
801-
802-
803-
804-
805-
806-
807-
808-
809-
810-
811-
812-
813-
814-
815-
816-
817-
818-
819-
820-
821-
822-
823-
824-
825-
826-
827-
828-
829-
830-
831-
832-
833-
834-
835-
836-
837-
838-
839-
840-
841-
842-
843-
844-
845-
846-
847-
848-
849-
850-
851-
852-
853-
854-
855-
856-
857-
858-
859-
860-
861-
862-
863-
864-
865-
866-
867-
868-
869-
870-
871-
872-
873-
874-
875-
876-
877-
878-
879-
880-
881-
882-
883-
884-
885-
886-
887-
888-
889-
890-
891-
892-
893-
894-
895-
896-
897-
898-
899-
900-
901-
902-
903-
904-
905-
906-
907-
908-
909-
910-
911-
912-
913-
914-
915-
916-
917-
918-
919-
920-
921-
922-
923-
924-
925-
926-
927-
928-
929-
930-
931-
932-
933-
934-
935-
936-
937-
938-
939-
940-
941-
942-
943-
944-
945-
946-
947-
948-
949-
950-
951-
952-
953-
954-
955-
956-
957-
958-
959-
960-
961-
962-
963-
964-
965-
966-
967-
968-
969-
970-
971-
972-
973-
974-
975-
976-
977-
978-
979-
980-
981-
982-
983-
984-
985-
986-
987-
988-
989-
990-
991-
992-
993-
994-
995-
996-
997-
998-
999-
1000-
1001-
1002-
1003-
1004-
1005-
1006-
1007-
1008-
1009-
1010-
1011-
1012-
1013-
1014-
1015-
1016-
1017-
1018-
1019-
1020-
1021-
1022-
1023-
1024-
1025-
1026-
1027-
1028-
1029-
1030-
1031-
1032-
1033-
1034-
1035-
1036-
1037-
1038-
1039-
1040-
1041-
1042-
1043-
1044-
1045-
1046-
1047-
1048-
1049-
1050-
1051-
1052-
1053-
1054-
1055-
1056-
1057-
1058-
1059-
1060-
1061-
1062-
1063-
1064-
1065-
1066-
1067-
1068-
1069-
1070-
1071-
1072-
1073-
1074-
1075-
1076-
1077-
1078-
1079-
1080-
1081-
1082-
1083-
1084-
1085-
1086-
1087-
1088-
1089-
1090-
1091-
1092-
1093-
1094-
1095-
1096-
1097-
1098-
1099-
1100-
1101-
1102-
1103-
1104-
1105-
1106-
1107-
1108-
1109-
1110-
1111-
1112-
1113-
1114-
1115-
1116-
1117-
1118-
1119-
1120-
1121-
1122-
1123-
1124-
1125-
1126-
1127-
1128-
1129-
1130-
1131-
1132-
1133-
1134-
1135-
1136-
1137-
1138-
1139-
1140-
1141-
1142-
1143-
1144-
1145-
1146-
1147-
1148-
1149-
1150-
1151-
1152-
1153-
1154-
1155-
1156-
1157-
1158-
1159-
1160-
1161-
1162-
1163-
1164-
1165-
1166-
1167-
1168-
1169-
1170-
1171-
1172-
1173-
1174-
1175-
1176-
1177-
1178-
1179-
1180-
1181-
1182-
1183-
1184-
1185-
1186-
1187-
1188-
1189-
1190-
1191-
1192-
1193-
1194-
1195-
1196-
1197-
1198-
1199-
1200-
1201-
1202-
1203-
1204-
1205-
1206-
1207-
1208-
1209-
1210-
1211-
1212-
1213-
1214-
1215-
1216-
1217-
1218-
1219-
1220-
1221-
1222-
1223-
1224-
1225-
1226-
1227-
1228-
1229-
1230-
1231-
1232-
1233-
1234-
1235-
1236-
1237-
1238-
1239-
1240-
1241-
1242-
1243-
1244-
1245-
1246-
1247-
1248-
1249-
1250-
1251-
1252-
1253-
1254-
1255-
1256-
1257-
1258-
1259-
1260-
1261-
1262-
1263-
1264-
1265-
1266-
1267-
1268-
1269-
1270-
1271-
1272-
1273-
1274-
1275-
1276-
1277-
1278-
1279-
1280-
1281-
1282-
1283-
1284-
1285-
1286-
1287-
1288-
1289-
1290-
1291-
1292-
1293-
1294-
1295-
1296-
1297-
1298-
1299-
1300-
1301-
1302-
1303-
1304-
1305-
1306-
1307-
1308-
1309-
1310-
1311-
1312-
1313-
1314-
1315-
1316-
1317-
1318-
1319-
1320-
1321-
1322-
1323-
1324-
1325-
1326-
1327-
1328-
1329-
1330-
1331-
1332-
1333-
1334-
1335-
1336-
1337-
1338-
1339-
1340-
1341-
1342-
1343-
1344-
1345-
1346-
1347-
1348-
1349-
1350-
1351-
1352-
1353-
1354-
1355-
1356-
1357-
1358-
1359-
1360-
1361-
1362-
1363-
1364-
1365-
1366-
1367-
1368-
1369-
1370-
1371-
1372-
1373-
1374-
1375-
1376-
1377-
1378-
1379-
1380-
1381-
1382-
1383-
1384-
1385-
1386-
1387-
1388-
1389-
1390-
1391-
1392-
1393-
1394-
1395-
1396-
1397-
1398-
1399-
1400-
1401-
1402-
1403-
1404-
1405-
1406-
1407-
1408-
1409-
1410-
1411-
1412-
1413-
1414-
1415-
1416-
1417-
1418-
1419-
1420-
1421-
1422-
1423-
1424-
1425-
1426-
1427-
1428-
1429-
1430-
1431-
1432-
1433-
1434-
1435-
1436-
1437-
1438-
1439-
1440-
1441-
1442-
1443-
1444-
1445-
1446-
1447-
1448-
1449-
1450-
1451-
1452-
1453-
1454-
1455-
1456-
1457-
1458-
1459-
1460-
1461-
1462-
1463-
1464-
1465-
1466-
1467-
1468-
1469-
1470-
1471-
1472-
1473-
1474-
1475-
1476-
1477-
1478-
1479-
1480-
1481-
1482-
1483-
1484-
1485-
1486-
1487-
1488-
1489-
1490-
1491-
1492-
1493-
1494-
1495-
1496-
1497-
1498-
1499-
1500-
1501-
1502-
1503-
1504-
1505-
1506-
1507-
1508-
1509-
1510-
1511-
1512-
1513-
1514-
1515-
1516-
1517-
1518-
1519-
1520-
1521-
1522-
1523-
1524-
1525-
1526-
1527-
1528-
1529-
1530-
1531-
1532-
1533-
1534-
1535-
1536-
1537-
1538-
1539-
1540-
1541-
1542-
1543-
1544-
1545-
1546-
1547-
1548-
1549-
1550-
1551-
1552-
1553-
1554-
1555-
1556-
1557-
1558-
1559-
1560-
1561-
1562-
1563-
1564-
1565-
1566-
1567-
1568-
1569-
1570-
1571-
1572-
1573-
1574-
1575-
1576-
1577-
1578-
1579-
1580-
1581-
1582-
1583-
1584-
1585-
1586-
1587-
1588-
1589-
1590-
1591-
1592-
1593-
1594-
1595-
1596-
1597-
1598-
1599-
1600-
1601-
1602-
1603-
1604-
1605-
1606-
1607-
1608-
1609-
1610-
1611-
1612-
1613-
1614-
1615-
1616-
1617-
1618-
1619-
1620-
1621-
1622-
1623-
1624-
1625-
1626-
1627-
1628-
1629-
1630-
1631-
1632-
1633-
1634-
1635-
1636-
1637-
1638-
1639-
1640-
1641-
1642-
1643-
1644-
1645-
1646-
1647-
1648-
1649-
1650-
1651-
1652-
1653-
1654-
1655-
1656-
1657-
1658-
1659-
1660-
1661-
1662-
1663-
1664-
1665-
1666-
1667-
1668-
1669-
1670-
1671-
1672-
1673-
1674-
1675-
1676-
1677-
1678-
1679-
1680-
1681-
1682-
1683-
1684-
1685-
1686-
1687-
1688-
1689-
1690-
1691-
1692-
1693-
1694-
1695-
1696-
1697-
1698-
1699-
1700-
1701-
1702-
1703-
1704-
1705-
1706-
1707-
1708-
1709-
1710-
1711-
1712-
1713-
1714-
1715-
1716-
1717-
1718-
1719-
1720-
1721-
1722-
1723-
1724-
1725-
1726-
1727-
1728-
1729-
1730-
1731-
1732-
1733-
1734-
1735-
1736-
1737-
1738-
1739-
1740-
1741-
1742-
1743-
1744-
1745-
1746-
1747-
1748-
1749-
1750-
1751-
1752-
1753-
1754-
1755-
1756-
1757-
1758-
1759-
1760-
1761-
1762-
1763-
1764-
1765-
1766-
1767-
1768-
1769-
1770-
1771-
1772-
1773-
1774-
1775-
1776-
1777-
1778-
1779-
1780-
1781-
1782-
1783-
1784-
1785-
1786-
1787-
1788-
1789-
1790-
1791-
1792-
1793-
1794-
1795-
1796-
1797-
1798-
1799-
1800-
1801-
1802-
1803-
1804-
1805-
1806-
1807-
1808-
1809-
1810-
1811-
1812-
1813-
1814-
1815-
1816-
1817-
1818-
1819-
1820-
1821-
1822-
1823-
1824-
1825-
1826-
1827-
1828-
1829-
1830-
1831-
1832-
1833-
1834-
1835-
1836-
1837-
1838-
1839-
1840-
1841-
1842-
1843-
1844-
1845-
1846-
1847-
1848-
1849-
1850-
1851-
1852-
1853-
1854-
1855-
1856-
1857-
1858-
1859-
1860-
1861-
1862-
1863-
1864-
1865-
1866-
1867-
1868-
1869-
1870-
1871-
1872-
1873-
1874-
1875-
1876-
1877-
1878-
1879-
1880-
1881-
1882-
1883-
1884-
1885-
1886-
1887-
1888-
1889-
1890-
1891-
1892-
1893-
1894-
1895-
1896-
1897-
1898-
1899-
1900-
1901-
1902-
1903-
1904-
1905-
1906-
1907-
1908-
1909-
1910-
1911-
1912-
1913-
1914-
1915-
1916-
1917-
1918-
1919-
1920-
1921-
1922-
1923-
1924-
1925-
1926-
1927-
1928-
1929-
1930-
1931-
1932-
1933-
1934-
1935-
1936-
1937-
1938-
1939-
1940-
1941-
1942-
1943-
1944-
1945-
1946-
1947-
1948-
1949-
1950-
1951-
1952-
1953-
1954-
1955-
1956-
1957-
1958-
1959-
1960-
1961-
1962-
1963-
1964-
1965-
1966-
1967-
1968-
1969-
1970-
1971-
1972-
1973-
1974-
1975-
1976-
1977-
1978-
1979-
1980-
1981-
1982-
1983-
1984-
1985-
1986-
1987-
1988-
1989-
1990-
1991-
1992-
1993-
1994-
1995-
1996-
1997-
1998-
1999-
2000-
2001-
2002-
2003-
2004-
2005-
2006-
2007-
2008-
2009-
2010-
2011-
2012-
2013-
2014-
2015-
2016-
2017-
2018-
2019-
2020-
2021-
2022-
2023-
2024-
2025-
2026-
2027-
2028-
2029-
2030-
2031-
2032-
2033-
2034-
2035-
2036-
2037-
2038-
2039-
2040-
2041-
2042-
2043-
2044-
2045-
2046-
2047-
2048-
2049-
2050-
2051-
2052-
2053-
2054-
2055-
2056-
2057-
2058-
2059-
2060-
2061-
2062-
2063-
2064-
2065-
2066-
2067-
2068-
2069-
2070-
2071-
2072-
2073-
2074-
2075-
2076-
2077-
2078-
2079-
2080-
2081-
2082-
2083-
2084-
2085-
2086-
2087-
2088-
2089-
2090-
2091-
2092-
2093-
2094-
2095-
2096-
2097-
2098-
2099-
2100-
2101-
2102-
2103-
2104-
2105-
2106-
2107-
2108-
2109-
2110-
2111-
2112-
2113-
2114-
2115-
2116-
2117-
2118
```

```

63
64+ class ErrorExplorador:
65+     """
66+     Clase que almacena errores léxicos encontrados por el explorador
67+     """
68+     def __init__(self, mensaje, texto, linea, columna):
69+         self.mensaje = mensaje
70+         self.texto = texto
71+         self.linea = linea
72+         self.columna = columna
73+
74+     def __str__(self):
75+         return f"ERROR: {self.mensaje} en línea {self.linea}, columna {self.columna} - Texto: '{self.texto}'"
76+
77+ class Explorador:
78+     """
79+     Clase que lleva el proceso principal de exploración y deja listos los
80+
81+ 30 hidden lines
82+
110     def __init__(self, contenido_archivo):
111         self.texto = contenido_archivo
112         self.componentes = []
113         self.errores = []
114
115     def explorar(self):
116         """
117
118 10 hidden lines
119
127         self.componentes = self.componentes + resultado
128         indice_linea += 1
129
130         # Si se encontró 1 o más errores se imprimen y se detiene el programa
131         if len(self.errores) > 0:
132             self.imprimir_errores()
133             sys.exit()
134
135     def imprimir_componentes(self):
136         """
137         Imprime en pantalla en formato amigable al usuario los componentes
138
139 4 hidden lines
140
142         print(componente) # Esto funciona por que el print llama al
143         | | | | | método __str__ de la instancia
144
145     def imprimir_errores(self):
146         """
147         Imprime todos los errores encontrados por el explorador en una estructura legible para el programador
148         """
149         print("\n==== ERRORES ENCONTRADOS ====")
150         for error in self.errores:
151             print(error)
152         print(f"Total de errores: {len(self.errores)}")
153
154     def registrar_error(self, mensaje, texto, linea, columna):
155         """
156         Registra un error para ser mostrado al final de la exploración
157         """
158         error = ErrorExplorador(mensaje, texto, linea, columna)
159         self.errores.append(error)
160
161     def procesar_linea(self, linea, indice_linea):
162         """
163
164 8 hidden lines

```

El único error que pude identificar es cuando el caracter no corresponde a ninguno de los ya establecidos

```

125 def procesar_lineas(self, linea, indice_linea):
126     """
127     """
128     # hidden lines
129
130     # Toma una línea y le va cortando pedazos hasta que se acabe
131     while(linea != ""):
132
133
134
135
136
137
138
139
140     # Separa los descriptores de componente en dos variables
141     for tipo_componente, regex in self.descriptores_componentes:
142
143
144     # Trata de hacer match con el descriptor actual
145     respuesta = re.match(regex, linea)
146
147
148
149
150     # Si hay coincidencia se procede a generar el componente
151     # léxico final
152     if respuesta is not None :
153         texto_coincidencia = respuesta.group()
154         # Se actualiza el índice de la columna que corresponde al componente
155         indice_columna_componente = indice_columna
156
157     # 21 hidden lines
158
159
160     # Se elimina el pedazo que hizo match
161     linea = linea[respuesta.end():]
162     break;
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076

```

Por alguna razón, Ciruelas original me estaba dando error al tratar de procesar los caracteres con tildes o diéresis o la ñ, para solucionar eso hice que re funcione en unicode, agregué la codificación utf-8 y todos los caracteres especiales del español a las expresiones regulares, es algo que ya había contemplado en mi documento de gramática. Por otro lado, en este mismo commit, vi que el orden de las expresiones regulares importa, tuve que mover la expresión regular de los flotantes antes de los enteros, sino todos los números los detectaba como enteros.

The image shows a code editor with two files open: `ciuelas.py` on the left and `explorador.py` on the right. Both files are written in Python and contain code for parsing command-line arguments and processing file contents. The code is written in Spanish and includes comments in Spanish.

ciuelas.py

```

1 from generador import Generador
2
3 import argparse
4
5
6
7
8
9
10
11 parser = argparse.ArgumentParser(description='Interprete para Ciuelas (el lenguaje)')
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
9
```

Ciruelas es un lenguaje que me parece muy similar a Python en su ausencia de ; o algún otro símbolo para terminar una línea, aún así Ciruelas sigue utilizando las llaves para delimitar una función, por ejemplo, lo que en Python veríamos así:

```
def imprimir(texto):
```

```
print(texto)
```

En Ciruelas se vería así:

```
mae imprimir (texto){  
  sueltele(texto)  
}
```

Esto entiendo que es por la simplicidad de buscar un símbolo específico para saber que ya terminó una función, hacer eso es mucho más seguro que el manejo de indentación que hace Python. Pero al centrarnos en las funciones nuevas que estoy proponiendo en Python tenemos el try-except que se ve así:

```
try:
```

```
  resultado = a / b
```

```
except:
```

```
  print("No puedes dividir por cero")
```

Mientras que en Ciruelas mi propuesta es que se vea así, es menos limpio con respecto a cantidad de caracteres, pero más fácil de procesar:

```
juéguesela{  
  resultado metale a desmadeje b  
}tortón{  
  sueltele(~No puedes dividir por cero~)  
}
```

Además en Python hay un for que lo podemos ver en el siguiente ejemplo:

```
for i in range(len(texto)):
```

```
  print(texto[i])
```

El de Ciruelas que propongo pienso que es bastante diferente, pero queda más claro todas las variables involucradas y el "step" del for, que en este caso sube de 1 en 1:

```
dele vuelta (i metale 0 / i poquitico largo_texto / i metale i  
hechele 1){  
  sueltele(viene_bolita(texto / i))  
}
```

Por último, el switch-case es extraño de comparar puesto que nada similar existía antes de Python 3.10, es curioso que se tardara tanto en implementar, sí técnicamente no hace falta, pero muchas veces se termina viendo más limpio que otras opciones como solo implementaciones alternativas de tratar de hacer lo mismo únicamente con if, diccionarios o funciones, luego se implementó el match que tiene la siguiente forma:

```
match x:
    case 1:
        print("El resultado es 1")
    case 2:
        print("El resultado es 2")
    case _:
        print("El resultado es diferente a 1 y 2")
```

En la extensión de Ciruelas yo propuse que tuviera una forma similar, pues no me agradan los breaks que se pueden encontrar en los switches de otros lenguajes, pero con más llaves para saber dónde empieza y dónde termina cada caso, como se muestra a continuación:

```
como está la vara(x){
    movida 1{
        sueltele(~El resultado es 1~)
    }
    movida 2{
        sueltele(~El resultado es 2~)
    }
    sino ni modo{
        sueltele(~El resultado es diferente a 1 y 2~)
    }
}
```

Lecciones aprendidas

Lo que voy aprendiendo de esto es primeramente no fiarse del código ya hecho, no hay que optimizar absolutamente todo si no hace falta para evitar un problema mayor, pero el caso de los caracteres especiales del español cuando vi que no se procesaron supe que tenía que

modificar lo que ya estaba hecho, igualmente cuando vi que el explorador asumía que todos los números eran enteros únicamente porque esa definición estaba antes que la de flotante, entiendo que eso pasa porque el intérprete de Python funciona de arriba hacia abajo y como vio un match no se molestó en verificar los demás. Por último, me sorprendió que el manejo de columnas fuera tan raro, pensé que era un estandar, pero cuando vi las inconsistencias entre la columna que me indicaba Visual Studio Code y el block de notas pude aprender que cada editor de texto considera al tabulador de formas diferentes, algunos lo ven como solo 1 caracter, otros como 2, 4 u 8 caracteres, esa era la razón, entonces tuve que decidir un número que fue 4, principalmente para seguir lo que indica VS Code.

Memes



Python
revisando todas
las expresiones
regulares



Python siendo
un vago y
tomando la
primera que sirve





