

Fundamentos de Programação (T)

Licenciatura em Videojogos

Ano 1 / Semestre 1

Sumário

- Ficheiros
 - Json
- Executáveis
- Exceções
 - Erros Clássicos vs. Exceções
 - Tratamento
 - Lançamento
 - Propagação
 - Parametrização
- Asserções



Ficheiros



Ficheiros

- Vamos ver como conseguimos tratar ficheiros
- Começamos pela leitura:
 - Para conseguirmos ler o ficheiro, primeiro temos que o abrir

```
file = open("lorem.txt", "rt")
```

- Sintaxe:

Nome do
ficheiro

Modo de abertura:
r = read
t = ficheiro de texto

Ficheiros

- Ler ficheiros:

- Para abriro ficheiro:

```
file = open("lorem.txt", "rt")
```

- Depois lemos a primeira linha:

```
first_line = file.readline()
```

- No final, temos sempre que fechar o ficheiro:

```
file.close()
```

Ficheiros

```
file = open("lorem.txt", "rt")
```

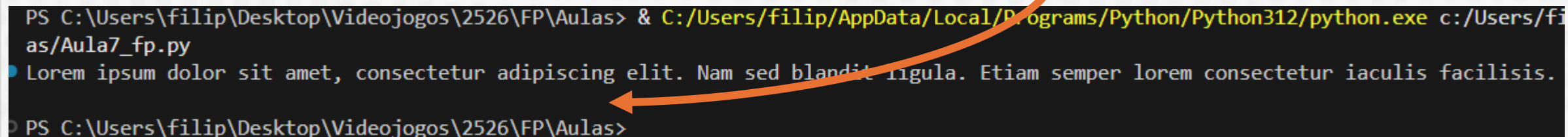
```
first_line = file.readline()
```

```
print(first_line)
```

```
file.close()
```

Porquê este espaço?

Porque nós lemos uma linha (que acaba num `return "\n"`) e o `print` também coloca outro no fim.



```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas> & C:/Users/filip/AppData/Local/Programs/Python/Python312/python.exe c:/Users/filip/Desktop/Videojogos/2526/FP/Aulas/Aula7_fp.py
• Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam sed blandit ligula. Etiam semper lorem consectetur iaculis facilisis.
• PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
```




Ficheiros

- Ler ficheiros:
 - Primeiro temos que o abrir:

```
file = open("lorem.txt", "rt")
```

- Para lermos todas as linhas:

```
all_lines = file.readlines()
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas> & C:/Users/filip/AppData/Local/Programs/Python/Python312/python.exe c:/Users/filip/Desktop/Videojogos/2526/FP/Aulas/Aula7_fp.py
• Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam sed blandit ligula. Etiam semper lorem consectetur iaculis facilisis.

PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
```

Cria uma lista de strings, em que cada elemento é uma linha do ficheiro txt.
O `first_line` anterior seria o `all_lines[0]`

```
• ['Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam sed blandit ligula. Etiam semper lorem consectetur iaculis facilisis. \n', 'Duis magna tortor, mollis nec ultrices eu, porttitor ut libero. Mauris ex eros, pellentesque a odio non, posuere ultrices purus. \n', 'Aliquam dapibus tellus mi, ultricies ullamcorper turpis consequat id. Aliquam erat volutpat. Suspendisse imperdiet iaculis finibus. \n', 'Aliquam suscipit turpis vitae lectus maximus rhoncus. Quisque congue posuere congue. \n', 'Morbi id condimentum ipsum. \n', 'Vestibulum nisi mi, mattis ut mauris nec, mattis euismod dolor. \n', 'Sed pellentesque dolor quis arcu rutrum vulputate. Nullam sit amet lacus quis nisl sagittis finibus. Etiam dictum et est venenatis ullamcorper. \n', 'Ut auctor nunc ac nulla varius mollis. Curabitur vehicula fermentum odales. Ut vel fermentum diam. In hac habitasse platea dictumst. \n', 'Quisque tellus lorem, pulvinar in malesuada ac, viverra vitae erat. Suspendisse facilisis metus lacinia sagittis fermentum. \n', 'Nunc pellentesque nunc ac sem bibendum efficitur. Vivamus dapibus ligula mauris, vel euismod metus faucibus at. Donec fermentum posuere ex id auctor. \n', 'Nulla vestibulum nisi eget molestie tincidunt. Nullam dictum laoreet consectetur. Pellentesque eget eleifend nunc. \n', 'Aenean tempus, nisi vel elementum elementum, arcu urna auctor mauris, in aliquet ligula dui nec diam. Morbi tempor sodales congue. \n', 'Integer quis metus aliquam lorem egestas egestas. In vel elit, posuere quis iaculis et, malesuada et tellus. \n', 'Vestibulum facilisis elit in nunc sagittis semper quis eget eros. \n', 'Integer eget consectetur lectus, a lacinia dui. Maecenas blandit volutpat augue, sit amet bibendum nunc faucibus eget. \n', 'Praesent pharetra gravida nisi sed tempor. Donec eu placerat est, quis scelerisque purus. Sed tincidunt sem vel lacus mollis dictum. \n', 'Integer quis arcu mauris. Nam libero libero, pretium ut dictum nec, tempor in diam. Nulla nec urna lectus. \n', 'Duis blandit mi ac lacus pretium, at lobortis erat venenatis. Donec rhoncus neque id magna luctus, sed fermentum eros finibus. \n', 'Sed sem odio, rhoncus quis massa vel, sagittis sagittis lectus. \n', 'Morbi laoreet odio in aliquam blandit. Curabitur cursus neque cursus purus mollis, non malesuada sem laoreet. Duis ut eros diam. \n', 'Donec eu odio egestas, eleifend libero in, tristique est. Phasellus ultrices velit quam, a sagittis tortor pellentesque et. \n', 'Maecenas quis tristique nisi. Cras laoreet sit amet diam ac fermentum. In dictum a odio ac varius. Quisque dapibus maximus consectetur. \n', 'Aliquam a ligula ac sem placerat rhoncus. \n']
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
```

Ficheiros

```
file = open("lorem.txt", "rt")

all_lines = file.readlines()

for line in all_lines:

    print(line.strip())

file.close()
```

```
1 2 3 4 5 6 7 8 9 10 11 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 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 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 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474 2475 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487 2488 2489 2490 2491 2492 2493 2494 2495 2496 2497 2498 2499 2500 2501 2502 2503 2504 2505 2506 2507 2508 2509 2510 2511 2512 2513 2514 2515 2516 2517 2518 2519 2520 2521 2522 2523 2524 2525 2526 2527 2528 2529 2530 2531 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541 2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555 2556 2557 2558 2559 2560 2561 2562 2563 2564 2565 2566 2567 2568 2569 2570 2571 2572 2573 2574 2575 2576 2577 2578 2579 2580 2581 2582 2583 2584 2585 2586 2587 2588 2589 2590 2591 2592 2593 2594 2595 2596 2597 2598 2599 2600 2601 2602 2603 2604 2605 2606 2607 2608 2609 2610 2611 2612 2613 2614 2615 2616 2617 2618 2619 2620 2621 2622 2623 2624 2625 2626 2627 2628 2629 2630 2631 2632 2633 2634 2635 2636 2637 2638 2639 2640 2641 2642 2643 2644 2645 2646 2647 264
```


Ficheiros

- Podemos depois tratar o `readline()` usando operações de strings:

```
file = open("lorem.txt", "rt")
```

```
first_line = file.readline()
```

```
first_line_no_punctuation = first_line.replace(".", "").replace(",", "")
```

```
first_line_list = first_line_no_punctuation.split(" ")
```

```
first_line_caps = first_line.upper()
```

```
print(first_line)
```

```
print(first_line_no_punctuation)
```

```
print(first_line_list)
```

```
print(first_line_caps)
```

```
file.close()
```

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam sed blandit ligula. Et  
iam semper lorem consectetur iaculis facilisis.
```

```
Lorem ipsum dolor sit amet consectetur adipiscing elit Nam sed blandit ligula Etiam  
semper lorem consectetur iaculis facilisis
```

```
['Lorem', 'ipsum', 'dolor', 'sit', 'amet', 'consectetur', 'adipiscing', 'elit', 'Na  
m', 'sed', 'blandit', 'ligula', 'Etiam', 'semper', 'lorem', 'consectetur', 'iaculis  
, 'facilisis', '\n']
```

```
LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT. NAM SED BLANDIT LIGULA. ET  
IAM SEMPER LOREM CONSECTETUR IACULIS FACILISIS.
```

Ficheiros

- Modos de abertura de um ficheiro:

- "r" : **read** Lê simplesmente o ficheiro
- "w" : **write** Se o ficheiro existir, será apagado, se não existir será criado
- "a" : **append** Se o ficheiro existir, novo conteúdo será acrescentado no fim, se não existir será criado
- "t" : modo texto
- "b" : modo binário

Ficheiros

- **Modo texto vs. Modo binário**

- Em sistemas baseados em Posix (Mac, Linux, Solaris, RedHat, etc) não existe diferença nenhuma
- Em sistemas Windows, existe uma questão de conversão de caracteres:
 - Quando se escreve um `\n`, é escrito de facto `\r\n`
 - Quando se lê um `\r\n`, é lido para a string um `\n`
 - Quando se lê um código hexadecimal `0x1a`, o input é considerado que chegou ao fim (legacy do CP/M)
- Resumindo:
 - Modo texto = ler/escrever texto, com codificação (UTF-8) e possíveis traduções de `\n` em Windows.
 - Modo binário = ler/escrever bytes exatamente como estão, usado para ficheiros não textuais.



Ficheiros

```
file = open("lorem.txt", "wt")
```

```
for i in range(0,10):
```

```
    file.write("Line " + str(i) + "\n")
```

```
file.close()
```

≡ lorem.txt

```
1 Line 0
2 Line 1
3 Line 2
4 Line 3
5 Line 4
6 Line 5
7 Line 6
8 Line 7
9 Line 8
10 Line 9
11 |
```

***** OLD VERSION

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam sed blar
Duis magna tortor, mollis nec ultrices eu, porttitor ut libero. Mauri
Aliquam dapibus tellus mi, ultricies ullamcorper turpis consequat id.
Aliquam suscipit turpis vitae lectus maximus rhoncus. Quisque congue
Vestibulum nisi mi, mattis ut mauris nec, mattis euismod dolor.

Sed pellentesque dolor quis arcu rutrum vulputate. Nullam sit amet la
Ut auctor nunc ac nulla varius mollis. Curabitur vehicula fermentum s

Quisque tellus lorem, pulvinar in malesuada ac, viverra vitae erat. S
Nunc pellentesque nunc ac sem bibendum efficitur. Vivamus dapibus lig
Nulla vestibulum nisi eget molestie tincidunt. Nullam dictum laoreet
Aenean tempor, nisi vel elementum elementum, arcu urna auctor mauris,
Integer quis metus aliquam lorem egestas egestas. In velit elit, posu
Vestibulum facilisis elit in nunc sagittis semper quis eget eros.

Integer eget consectetur lectus, a lacinia dui. Maecenas blandit volu
Praesent pharetra gravida nisi sed tempor. Donec eu placerat est, qui
Integer quis arcu mauris. Nam libero libero, pretium ut dictum nec, t
Duis blandit mi ac lacus pretium, at lobortis erat venenatis. Donec r
Sed sem odio, rhoncus quis massa vel, sagittis sagittis lectus.

Morbi laoreet odio in aliquam blandit. Curabitur cursus neque cursus
Donec eu odio egestas, eleifend libero in, tristique est. Phasellus u
Maecenas quis tristique nisi. Cras laoreet sit amet diam ac fermentum
Aliquam at ligula ac sem placerat rhoncus.

***** USING WT

***** NEW VERSION

```
Line 0
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
```

Ficheiros

```
file = open("lorem.txt", "at")

for i in range(10,20):

    file.write("Line " + str(i) + "\n")

file.close()
```

```
1 Line 0
2 Line 1
3 Line 2
4 Line 3
5 Line 4
6 Line 5
7 Line 6
8 Line 7
9 Line 8
10 Line 9
11 Line 10
12 Line 11
13 Line 12
14 Line 13
15 Line 14
16 Line 15
17 Line 16
18 Line 17
19 Line 18
20 Line 19
```

```
***** NEW VERSION
```

```
Line 0
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
```

```
***** USED APPEND
```

```
Line 0
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10
Line 11
Line 12
Line 13
Line 14
Line 15
Line 16
Line 17
Line 18
Line 19
```




JSON

JSON

- JSON – JavaScript Object Notation
- É um formato aberto que usa texto legível para transferir dados em formato chave-valor (*key-value*)
- A generalidade das linguagens de alto nível têm bibliotecas default para ler e escrever JSON
- O Python tem suporte nativo para JSON, e suporta objectos simples
- Para objectos mais complexos é preferível usar outras bibliotecas

```
{  
  "firstName": "John",  
  "lastName": "Smith",  
  "isAlive": true,  
  "age": 27,  
  "address": {  
    "streetAddress": "21 2nd Street",  
    "city": "New York",  
    "state": "NY",  
    "postalCode": "10021-3100"  
  },  
  "phoneNumbers": [  
    {  
      "type": "home",  
      "number": "212 555-1234"  
    },  
    {  
      "type": "office",  
      "number": "646 555-4567"  
    },  
    {  
      "type": "mobile",  
      "number": "123 456-7890"  
    }  
  ],  
  "children": [],  
  "spouse": null  
}
```

JSON

- Primeiro vamos declarar uma estrutura de dados que queremos exportar:

```
data = {  
    "points" : [  
        { "x" : 0, "y" : 1 },  
        { "x" : 1, "y" : 3 },  
        { "x" : 4, "y" : 7 }  
    ],  
    "n_points" : 3  
}
```

Nós já vimos esta estrutura de dados

JSON

- Se escrevemos os dados já obtemos algo semelhante a uma estrutura de um JSON

```
print(data)
```

```
{'points': [{'x': 0, 'y': 1}, {'x': 1, 'y': 3}, {'x': 4, 'y': 7}], 'n_points': 3}
```

Dicionário em Python e ficheiros JSON seguem a mesma estrutura

JSON

- Se quiséssemos escrever isto para um ficheiro JSON, podíamos fazer:

```
import json
```

```
file = open("data.json", "wt")
```

```
json_data = json.dumps(data)
```

```
file.write(json_data)
```

```
file.close()
```

```
{ } data.json > ...
```

```
1  [{"points": [{"x": 0, "y": 1}, {"x": 1, "y": 3}, {"x": 4, "y": 7}], "n_points": 3}]
```


JSON

- Podemos também ler o JSON e importar os dados para reconstruir o array:

```
import json

file = open("data.json", "rt")
json_data = file.read()
data = json.loads(json_data)
print(data)

file.close()
```

```
• {'points': [{'x': 0, 'y': 1}, {'x': 1, 'y': 3}, {'x': 4, 'y': 7}], 'n_points': 3}
```

JSON

- Não é possível serializar classes com a biblioteca default
- Serialização é o que se chama a operação de converter estruturas de dados de uma linguagem de programação num formato bom para ficheiros.
- A operação inversa chama-se “desserialização”
- Há outras bibliotecas para este propósito (por exemplo o marshmallow-dataclass, JSONPickle).
- Para coisas simples, como guardar opções do jogador, ou high-scores, o sistema default serve.
- Resumindo:

O JSON default no Python só sabe trabalhar com tipos básicos. Se precisarem de guardar classes/objetos, precisam de uma biblioteca extra. Mas para configs de jogo, high-scores ou dados simples, a biblioteca default é mais do que suficiente.

JSON - Serialização

- Serializar significa converter uma estrutura de dados num formato linear e “guardável” em JSON
- **Por exemplo:**

```
save = {  
    "player_name" : "Phil",  
    "class" : "rogue",  
    "level" : "77",  
    "map_position" : { "location" : "titan", "x" : 245, "y" : 123, "z" : 45}  
}
```

- Converter isto num JSON seria um processo de serialização

```
{  
  "player_name": "Phil", "class": "rogue", "level": "77", "map_position": {"location": "titan", "x": 245, "y": 123, "z": 45}  
}
```

JSON - Desserialização

- Desserializar significa o processo inverso, converter um ficheiro JSON numa estrutura de dados em Python
- No caso específico do JSON que criámos com save data, poderíamos fazer um “load” quando voltamos ao jogo

```
import json
player_data = {}
file = open("save_game.json", "rt")
json_data = file.read()
player_data = json.loads(json_data)
print("loaded player data:", player_data)
file.close()
```

```
loaded player data: {'player_name': 'Phil', 'class': 'rogue', 'level': '77', 'map_position': {'location': 'titan', 'x': 245, 'y': 123, 'z': 45}}
```



Executáveis

Executáveis

- É possível tornar uma aplicação Python num executável “normal” -> um programa
- Um executável otimiza o código, fazendo com que a aplicação corra mais rapidamente consumindo menos recursos
- Criar um executável fará com que o nosso programa fique agarrado a uma plataforma
 - Se criarem um executável para windows, só vai funcionar no windows, e o mesmo para outros sistemas operativos
- Um executável esconde o código, dificultando que seja visível pelos utilizadores
 - O que é sempre importante num jogo

Executáveis

- Existem várias aplicações para criar executáveis
- Infelizmente, há alguma confusão com versões (Python 2 vs Python 3).

- O PyInstaller é o mais simples de usar, basta instalar:

```
pip install pyinstaller
```

- Depois, só é preciso fazer:

```
python -m PyInstaller --onefile <mainFile.py>
```

- Na pasta dist vai estar um executável com o nosso programa, para a plataforma onde fizemos esta operação.
- Não esquecer de copiar quaisquer ficheiros necessários para a execução (imagens, sons, etc).

```
MINGW64 /d/projects/fc_2019_exemplos (master)
pyinstaller --onefile adventure9.py
version: 3.5
python: 3.7.3
platform: Windows-10-10.0.17134-SP0
path: D:\projects\fc_2019_exemplos\adventure9.spec
spec not available.
adding PYTHONPATH with paths
D:\projects\fc_2019_exemplos, 'D:\projects\fc_2019_exemplos'
adding Analysis
adding Analysis because Analysis-00.toc is non existent
analyzing module dependency graph...
analyzing module graph hooks...
adding base_library.zip ...
adding Analysis Analysis-00.toc
adding Microsoft.Windows.Common-Controls to dependent as
D:\opt\python\python.exe
adding module hooks...
analyzing D:\projects\fc_2019_exemplos\adventure9.py
adding module hooks...
adding module hook "hook-encodings.py"...
adding module hook "hook-pydoc.py"...
adding module hook "hook-xml.py"...
looking for ctypes DLLs
analyzing run-time hooks ...
looking for dynamic libraries
looking for eggs
adding Python library d:\opt\python\python37.dll
and binding redirects:
...
things written to D:\projects\fc_2019_exemplos\build\adventure9
ph cross-reference written to D:\projects\fc_2019_exemplos\build\adventure9
adding PYZ
adding PYZ because PYZ-00.toc is non existent
adding PYZ (ZlibArchive) D:\projects\fc_2019_exemplos\build\adventure9
adding PYZ (ZlibArchive) D:\projects\fc_2019_exemplos\build\adventure9
adding PKG
adding PKG because PKG-00.toc is non existent
adding PKG (CArchive) PKG-00.pkg
adding PKG (CArchive) PKG-00.pkg completed successfully
adding bootloader d:\opt\python\lib\site-packages\PyInstaller\bootloader
adding EXE
adding EXE because EXE-00.toc is non existent
adding EXE from EXE-00.toc
adding archive to EXE D:\projects\fc_2019_exemplos\dist\adventure9.exe
adding EXE from EXE-00.toc completed successfully.
```



Exercícios

Exercícios

1. Gestão de diálogos

- Usa o ficheiro `dialogs.txt` no moodle, e faz print de todas as linhas do ficheiro.
- Separa cada linha do ficheiro por nome e fala.
- Constroi um dicionário em que os nomes são as *keys* e as falar os *values*.
- Serializa o dicionário num ficheiro JSON – `dialogs.json`
- Faz um executável do programa que leia um qualquer `dialogs.txt` e o transforme num `dialogs.json`.
 - Esse executável deve pedir ao utilizador um input do ID de um NPC existente e imprimir apenas as falas desse NPC

2. Sistema de high scores

- Usa o ficheiro `highscore.json` do moodle, cria um dicionário com os dados e imprime. Faz uma verificação para confirmar que o ficheiro não está vazio
- Usa o sistema de input para pedir ao utilizador um novo nome e score inteiro. Adiciona esses dados aos highscores
- Ordena a lista de highscores por score decrescente (sem usar `sort()` ou `sorted()`)
- Mantém apenas os 3 melhores resultados
- Serializa novamente a lista de scores como `new_highscores.json`
- No final imprime o novo ranking numerado (1º, 2º e 3º)

3. Merge de game logs

- Usa os dois ficheiros que estão no moodle – `day1.log` e `day2.log`
- Abre os ficheiros e lê todas as linhas de cada um deles
- Cria um merged log com todas as linhas dos dois logs, cada linha deve ter a indicação do dia no formato `[DAY1]` ou `[DAY2]`
- Conta o total de linhas de cada um dos dias usando o merged log
- Cria um dicionário estatístico que contenha o número de ações de cada dia e o total de ações de ambos os dias
- Serializa o merged log num `log_stats.json`

Excepções

Exceções

- Existem duas maneiras normais de gerir erros
 - *Error codes*
 - *Exceptions*
- *Error codes* são mais explícitos:

```
ret = some_function_that_can_fail("param1")
if (ret == False):
    print("Some error!")
    exit()
else:
    print("all good")
```

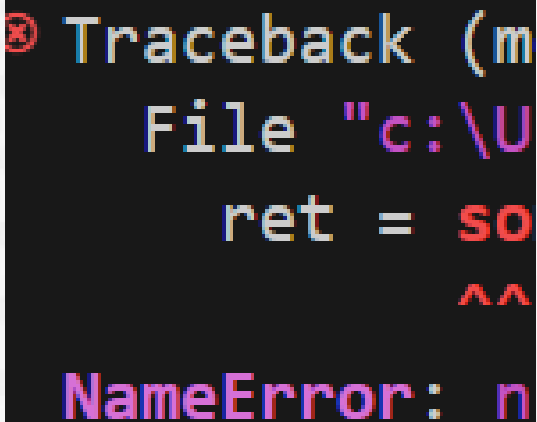
```
Traceback (most recent call last):
  File "c:\Users\filip\OneDrive\Ambiente de Trabalho\Aulas\IC\2526\Aula9_exerc_Python\aula7_fp.py", line 1, in <module>
    ret = some_function_that_can_fail("param1")
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
NameError: name 'some_function_that_can_fail' is not defined
```

Exceções

- Existem duas maneiras normais de gerir erros
 - *Error codes*
 - *Exceptions*
- *Exceptions* são mais estruturadas

```
try:  
    some_function_that_can_fail("param1")  
    print("all good")  
except [Error code]:  
    print("Some error!")  
    exit()
```

```
thon313/python.exe "c:/Users/filip/OneDrive/Ambiente de Trabalho/Aulas/IC  
Some error!
```



```
Traceback (m  
File "c:\U  
ret = so  
^^  
NameError: n
```

Excepções vs. Erros Clássicos

Exceções vs. Erros Clássicos

- Exceções permitem separar o código que gere os erros do código que executa normalmente, o que pode fazer o código ficar mais limpo e fácil de ler

```
try:  
    get_value_by_string_index(array, "10")  
except ValueError:  
    print("Failed because a conversion to int failed!")  
    exit()  
except IndexError:  
    print("Failed because the index was out of range!")  
    exit()  
except Exception:  
    print("Some other error!")  
    exit()
```

● Some other error!

Exceções vs. Erros Clássicos

- Exceções permitem separar o código que gere os erros do código que executa normalmente, o que pode fazer o código ficar mais limpo e fácil de ler

```
array = [1,2,3,4]
```

```
try:  
    print(array[5])  
except ValueError:  
    print("Failed because a conversion to int failed!")  
    exit()  
except IndexError:  
    print("Failed because the index was out of range!")  
    exit()  
except Exception :  
    print("Some other error!")  
    exit()
```

● Failed because the index was out of range!

Exceções vs. Erros Clássicos

- Exceções permitem separar o código que gere os erros do código que executa normalmente, o que pode fazer o código ficar mais limpo e fácil de ler

```
array = [1,2,3,4]

try:
    array.remove(5)
except ValueError:
    print("Failed because a conversion to int failed!")
    exit()
except IndexError:
    print("Failed because the index was out of range!")
    exit()
except Exception :
    print("Some other error!")
    exit()
```

● Failed because a conversion to int failed!

Exceções vs. Erros Clássicos

- Erros clássicos evitam o erro antes de acontecer (`if/else`).
- Exceções lidam com o erro depois de ele ocorrer (`try/except`).
- Se forem tratados, o programa continua normalmente.

```
def division_function(a,b):  
    if(b == 0):  
        print("Can't divide by zero")  
    else:  
        print(a/b)  
  
for i in range(6, 8):  
    for j in range(-1, 2):  
        division_function(i,j)
```

```
-6.0  
Can't divide by zero  
6.0  
-7.0  
Can't divide by zero  
7.0
```

VS.

```
def division_function(a,b):  
    ret = 0  
    try:  
        ret = a/b  
        print(ret)  
    except ZeroDivisionError:  
        print("Can't divide by zero")  
  
    return ret  
  
for i in range(6, 8):  
    for j in range(-1, 2):  
        division_function(i,j)
```

```
• -6.0  
Can't divide by zero  
6.0  
-7.0  
Can't divide by zero  
7.0
```

Exceções vs. Erros Clássicos

- Se não soubermos ou não conseguirmos antecipar o erro específico podemos usar:

```
def division_function(a,b):  
    ret = 0  
    try:  
        ret = a/b  
        print(ret)  
    except Exception:  
        print("something went wrong")  
  
    return ret  
  
for i in range(6,8):  
    for j in range(-1,2):  
        division_function(i,j)
```

```
● -6.0  
  something went wrong  
  6.0  
  -7.0  
  something went wrong  
  7.0
```

Excepções vs. Erros Clássicos

- As excepções podem (e devem) ser tratadas no ponto em que faz mais sentido, mesmo que seja “longe” da função que originou o erro:

O erro está aqui

As excepções são colocadas fora da função

```
def function_that_runs_a_lot_of_code():  
    #...  
    get_value_by_string_index(array, "10")  
    #...  
  
try:  
    function_that_runs_a_lot_of_code()  
except ValueError:  
    print("Failed because a conversion to int failed!")  
    exit()  
except IndexError:  
    print("Failed because the index was out of range!")  
    exit()  
except Exception:  
    print("Some other error!")  
    exit()
```

Exceções em Python - Tratamento

```
try:
    # Code to try to run
except Exception1:
    # Code to run if previous code throws an exception of
    # type Exception 1
except Exception2:
    # Code to run exception of type Exception 2 is thrown
except Exception:
    # Code to run if an exception was thrown, but wasn't
    # of type Exception1 or Exception2
finally:
    # Code to run if any exception was thrown

# Normal code
```


Exceções em Python - Lançamento

- Por vezes queremos ser nós a lançar (**throw** ou **raise**) uma excepção, a ser tratadas por outra parte do código:

```
raise NameError
```

- Neste caso lançamos um excepção de tipo `NameError`.
- Podemos também definir as nossas próprias excepções.
 - Por exemplo no caso da nossa aventura de texto, o player movimenta-se para uma área não definida:

```
class InvalidPosition(Exception):  
    pass
```

Falaremos mais sobre isto em
Programação Orientada a Objectos

```
raise InvalidPosition
```

Exceções em Python - Lançamento

- Exemplos:

```
def raise_an_exception():  
    raise NameError  
  
try:  
    raise_an_exception()  
except NameError:  
    print("There was an error")
```

● There was an error

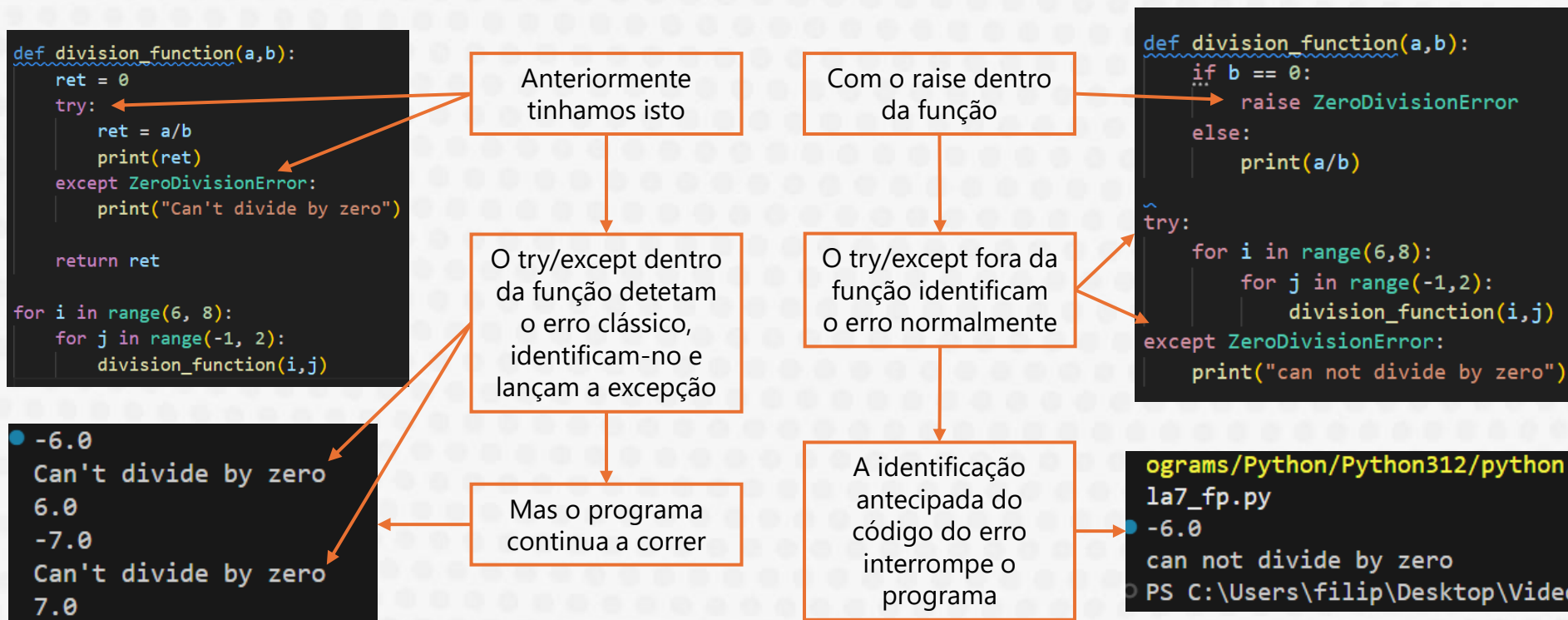
```
def division_function(a,b):  
    if b == 0:  
        raise ZeroDivisionError  
    else:  
        print(a/b)  
  
try:  
    for i in range(6,8):  
        for j in range(-1,2):  
            division_function(i,j)  
except ZeroDivisionError:  
    print("can not divide by zero")
```

```
ograms/Python/Python312/python.6  
la7_fp.py  
-6.0  
can not divide by zero  
PS C:\Users\filip\Desktop\Video
```



Exceções em Python - Lançamento

- O **raise** interrompe imediatamente a execução da função onde ocorre, e passa o controlo para o *except* mais próximo



Exceções em Python - Lançamento

- Às vezes precisamos de lançar exceções que são específicas do programa que estamos a desenvolver.
 - Python só reconhece por defeito erros genéricos (TypeError, ValueError, IndexError, etc.).
 - Mas alguns erros são próprios da lógica do nosso jogo, o interpretador não os reconhece automaticamente.
- No caso da Aventura de texto:
 - Se o player se mover para uma posição que não existe no mapa, isso é um erro lógico do jogo.
 - Não existe nenhuma exceção que represente esse problema, por isso temos de criar a nossa própria exceção e lançá-la:

```
class InvalidPosition(Exception):  
    pass  
  
raise InvalidPosition
```

Exceções em Python - Propagação

- Quando uma exceção é lançada e não é tratada pela função onde ocorreu
 - Ela é automaticamente enviada (propagada) para a função que chamou essa função.
- Este processo continua em cadeia, função acima, depois a seguinte, e assim por diante.....
 - Até encontrar um bloco `try/except` capaz de a apanhar.
- **Se nenhuma função tratar a exceção, ela chega ao interpretador de Python.**
 - Nesse momento, o programa termina imediatamente e o Python escreve a mensagem de erro no ecrã.
- A propagação é útil porque permite que erros sejam tratados ao nível certo do programa.
 - Não obrigatoriamente no local onde ocorreram.

Exceções em Python - Propagação

```
def f3():  
    print("f3: antes do erro")  
    raise ZeroDivisionError  
    print("f3: depois do erro")  
  
def f2():  
    print("f2: a chamar f3")  
    f3()  
    print("f2: depois de f3")  
  
def f1():  
    print("f1: a chamar f2")  
    f2()  
    print("f1: depois de f2")  
  
f1()
```

```
f1: a chamar f2  
f2: a chamar f3  
f3: antes do erro  
Traceback (most recent call last):  
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 147, in <module>  
    f1()  
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 144, in f1  
    f2()  
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 139, in f2  
    f3()  
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 134, in f3  
    raise ZeroDivisionError  
ZeroDivisionError
```

Exceções em Python - Propagação

```
def f3():  
    print("f3: antes do erro")  
    raise ZeroDivisionError  
    print("f3: depois do erro")  
  
def f2():  
    print("f2: a chamar f3")  
    f3()  
    print("f2: depois de f3")  
  
def f1():  
    print("f1: a chamar f2")  
    f2()  
    print("f1: depois de f2")  
  
try:  
    f1()  
except ZeroDivisionError:  
    print("ERRO APANHADO NO TRY EXTERNO!")
```

```
f1: a chamar f2  
f2: a chamar f3  
f3: antes do erro  
ERRO APANHADO NO TRY EXTERNO!
```

Exceções em Python - Propagação

- Pegando no exemplo da exceção que iríamos criar para o nosso jogo


```
class InvalidPosition(Exception):  
    pass  
  
def move_north():  
    # Some calculations were done and we got  
    # an error on the map positions  
    raise InvalidPosition  
  
# Main code for the game  
  
# Player added an input to move north  
move_north()
```

```
Traceback (most recent call last):  
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 167, in <module>  
    move_north()  
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 161, in move_north  
    raise InvalidPosition  
InvalidPosition
```

Exceções em Python - Propagação

- Pegando no exemplo da exceção que iríamos criar para o nosso jogo

```
class InvalidPosition(Exception):  
    pass  
  
def move_north():  
    # Some calculations were done and we got  
    # an error on the map positions  
    try:  
        raise InvalidPosition  
    except InvalidPosition:  
        print("That's an invalid position")  
  
# Main code for the game  
  
# Player added an input to move north  
move_north()
```



```
PS C:\Users\filip\Desktop\Videojogos\  
on312/python.exe c:/Users/filip/Deskt  
That's an invalid position  
PS C:\Users\filip\Desktop\Videojogos\
```



Exceções em Python - Propagação

- Pegando no exemplo da exceção que iríamos criar para o nosso jogo

```
class InvalidPosition(Exception):  
    pass  
  
def move_north():  
    # Some calculations were done and we got  
    # an error on the map positions  
    raise InvalidPosition  
  
# Main code for the game  
  
# Player added an input to move north  
try:  
    move_north()  
except InvalidPosition:  
    print("Invalid Map Position")
```

```
PS C:\Users\Filip\Desktop\Videojogos\2526> python.exe c:/Users/filip/Desktop/V  
Invalid Map Position  
PS C:\Users\Filip\Desktop\Videojogos\2526>
```


Exceções em Python - Parametrização

- Até agora vimos exceções simples, que apenas indicam que aconteceu um erro.
- Em muitos casos, queremos incluir informação adicional dentro da exceção:
 - coordenadas inválidas
 - valores introduzidos pelo jogador
 - o nome do ficheiro em erro
 - o estado do jogo no momento da falha
- Para isso, podemos definir a nossa própria exceção com parâmetros.
- Esta informação pode depois ser lida no bloco *except*, facilitando a resposta ao erro.

Exceções em Python - Parametrização

- Vamos ver melhor a construção de classes na próxima aula, mas a parametrização de uma exceção poderia se qualquer coisa deste género:

```
class InvalidPosition(Exception):  
    def __init__(self, x,y):  
        self.x = x  
        self.y = y  
  
def move_north(x, y):  
    # Some calculations were done and we got  
    # an error on the map positions  
    raise InvalidPosition(x,y)  
  
# Main code for the game  
  
# Player added an input to move north  
try:  
    move_north(15,20)  
except InvalidPosition as exp:  
    print("Invalid Map Position: " + str(exp.x) + "," + str(exp.y))
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas> .\Aulas/Aula7_fp.py  
Invalid Map Position: 15,20  
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
```



Asserções

Asserções

- Asserções são ferramentas importantes de *debugging*
- Um asserção permite parar um programa quando um situação que não deveria acontecer, de facto acontece
- Vamos ver um exemplo:

```
import math

def square_root(x):
    return math.sqrt(x)

print(square_root(9))
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas> python3 /Aulas/Aula7_fp.py
3.0
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
```

Asserções

- Pegando na mesma função

```
import math

def square_root(x):
    return math.sqrt(x)

print(square_root(-16))
```

```
Traceback (most recent call last):
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 181, in <module>
    print(square_root(-16))
    ^^^^^^^^^^^^^^^^^
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 179, in square_root
    return math.sqrt(x)
    ^^^^^^^^^^^^^
ValueError: math domain error
```

- O módulo math lida com números reais, onde não existem raízes de número negativos

Asserções

- Usando uma asserção

```
import math

def square_root(x):
    assert x >= 0
    return math.sqrt(x)

print(square_root(-16))
```

```
Traceback (most recent call last):
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 182, in <module>
    print(square_root(-16))
    ^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 179, in square_root
    assert x >= 0
AssertionError
```

Asserções

- Podemos usar asserções para escrever mensagens:

```
import math

def square_root(x):
    assert x >= 0, "You shouldn't be using math to find square root of a negative number, you should import cmath instead"
    return math.sqrt(x)

print(square_root(-16))
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas> & C:/Users/filip/AppData/Local/Programs/Python/Python312/python.exe c:/
/Aulas/Aula7_fp.py
Traceback (most recent call last):
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 182, in <module>
    print(square_root(-16))
          ^^^^^^^^^^^^^^^^^
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 179, in square_root
    assert x >= 0, "You shouldn't be using math to find square root of a negative number, you should import cmath instead"
AssertionError: You shouldn't be using math to find square root of a negative number, you should import cmath instead
```

Asserções

```
import cmath

def square_root(x):
    return cmath.sqrt(x)

print(square_root(-16))
```

```
PS C:\Users\filip\Desktop\Videojogos\2
/Aulas/Aula7_fp.py
● 4j
○ PS C:\Users\filip\Desktop\Videojogos\2
```

Asserções

- Para que servem:
 - Validar suposições
 - Dar erros mais expressivos acerca de situações
 - Alertar acerca de problemas que previmos que podiam dar problemas, mas que não acreditamos que possam acontecer devido à lógica do programa
- Porque não usar ifs ou exceptions?
 - Porque se tivermos a certeza que o programa agora está bem “comportado”, podemos desligar completamente as asserções, aumentando a performance.



Exercícios

Exercícios

No moodle, ficheiro `exerc_aula7_fp.py`:

- Cada bloco de código contém erros intencionais que fazem o programa reventar.
- O objectivo é implementar tratamento de excepções adequado usando:
 - `try / except`
 - `raise`
 - `assert`
- O objectivo não é corrigir os erros apenas tratá-los para que o código fique funcional
- Não alteres a lógica do programa. Apenas torna-o robusto.