

SPRAWOZDANIE

Imię i nazwisko: Jakub Kęsik

Nr albumu: 163040

Github: https://github.com/KasniQ/PODSTAWY_PROGRAMOWANIA



Python Unit Test

(pythonunittest.py)

Testy jednostkowe w Pythonie są realizowane przy użyciu modułu 'unittest', który pozwala na tworzenie, organizowanie i uruchamianie testów w celu weryfikacji poprawności kodu. Umożliwiają one sprawdzanie poszczególnych funkcji i metod, zapewniając, że działają one zgodnie z oczekiwaniami. Oferuje on różne metody asercji, które pomagają porównywać wyniki działania funkcji z przewidywanymi wartościami.

```
1 # 1. Write a Python unit test program to check if a given number is prime or not.
2
3 import unittest
4
5
6 def isPrime(number):
7     if number < 2:
8         return False
9     for i in range(2, int(number**0.5) + 1):
10         if number % i == 0:
11             return False
12     return True
13
14 class PrimeNumberTestCase(unittest.TestCase):
15     def test_prime_numbers(self):
16         prime = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31]
17         print("Prime numbers:", prime)
18         for number in prime:
19             self.assertTrue(isPrime(number), f"{number} is not a prime number")
20
21     def test_non_prime_numbers(self):
22         nonPrime = [4, 6, 8, 10, 12, 14, 16, 18, 20]
23         print("Non-prime numbers:", nonPrime)
24         for number in nonPrime:
25             self.assertFalse(isPrime(number), f"{number} is not a prime number")
26
27 if __name__ == '__main__':
28     unittest.main()
29
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY
Non-prime numbers: [4, 6, 8, 10, 12, 14, 16, 18, 20]
.Prime numbers: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31]
.

Ran 2 tests in 0.000s

OK
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> []

```

27
28 # 2. Write a Python unit test program to check if a list is sorted in ascending order.
29
30 def ascendingOrderTest(myList):
31     return all(myList[i] <= myList[i+1] for i in range(len(myList)-1))
32
33 class ascendingOrderTester(unittest.TestCase):
34     def testList(self):
35         myList = [1,2,3,4,5,6,7]
36         print("Sorted list: ", myList)
37         self.assertTrue(ascendingOrderTest(myList), "List is sorted in ascending order")
38     def testWrongList(self):
39         myList = [1,5,6,2,4,3]
40         print("Unsorted list: ", myList)
41         self.assertFalse(ascendingOrderTest(myList), "List is unsorted")
42 if __name__ == '__main__':
43     unittest.main()
44

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7
Sorted list: [1, 2, 3, 4, 5, 6, 7]
Unsorted list: [1, 5, 6, 2, 4, 3]
.
-----
Ran 2 tests in 0.000s

OK
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```

45
46 # 3. Write a Python unit test program that checks if two lists are equal.
47
48 def equallists(list1, list2):
49     return list1 == list2
50
51 class testListsEquality(unittest.TestCase):
52     def testEquallists(self):
53         print("\nEqual list test:\n", [5,582,37,238], "\n", [5,582,37,238])
54         self.assertTrue(equallists([5,582,37,238], [5,582,37,238]), "The lists are equal")
55
56     def testUnequalLists(self):
57         print("\nUnequal list test:\n", [1, 2, 3, 4], "\n", [4,5,6,778435])
58         self.assertFalse(equallists([1, 2, 3, 4], [4,5,6,778435]), "The lists are not equal")
59
60 if __name__ == '__main__':
61     unittest.main()
62
63

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7
Equal list test:
[5, 582, 37, 238]
[5, 582, 37, 238]
.
Unequal list test:
[1, 2, 3, 4]
[4, 5, 6, 778435]
.
-----
Ran 2 tests in 0.001s

OK
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PR
Testing non palindrome: hello
.Test palindrome: kajak
.
-----
Ran 2 tests in 0.000s

OK
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>
```

```

1 # 5. Write a Python unit test program to check if a file exists in a specified directory.
2
3 # example.txt
4 # file.txt
5 # plik.txt
6 # pythonasynchronous....
7 # pythonexceptionhand...
8 # pythonunittest.py
9
10 # 5. Write a Python unit test program to check if a file exists in a specified directory.
11
12 import os
13
14 def file_exists(directory, filename):
15     return os.path.isfile(os.path.join(directory, filename))
16
17 class TestFileExists(unittest.TestCase):
18
19     def test_file_exists(self):
20         print("Testing an existing file: ")
21         self.assertTrue(file_exists(".", "example.txt"))
22
23     def test_file_not_exists(self):
24         print("Testing non-existent file: ")
25         self.assertFalse(file_exists(".", "nonexistent.txt"))
26
27 if __name__ == "__main__":
28     unittest.main()
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
1
```

```

108 # 6. Write a Python unit test that checks if a function handles floating-point calculations accurately.
109
110
111 def addFloats(a, b):
112     return a + b
113
114 class testFloatingPoint(unittest.TestCase):
115     def testAddFloats(self):
116         print("Test:", addFloats(0.1, 0.2))
117         self.assertEqual(addFloats(0.1, 0.2), 0.3, places=7)
118         print("Test:", addFloats(1.1, 2.2))
119         self.assertEqual(addFloats(1.1, 2.2), 3.3, places=7)
120 if __name__ == "__main__":
121     unittest.main()
122
123

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7/test_floating_point.py
Test: 0.30000000000000004
Test: 3.3000000000000003
.
-----
Ran 1 test in 0.000s

OK
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```

125 # 7. Write a Python unit test program to check if a function handles multi-threading correctly.
126 import threading
127
128 def task():
129     result = 0
130     for i in range(1, 50):
131         result += i
132     return result
133
134 class multiThreadingTester(unittest.TestCase):
135     def testMultiThreading(self):
136         numberOfThreads = 10
137         threads = []
138         for _ in range(numberOfThreads):
139             t = threading.Thread(target=task)
140             threads.append(t)
141             t.start()
142         for t in threads:
143             t.join()
144         for t in threads:
145             self.assertFalse(t.is_alive())
146 if __name__ == "__main__":
147     unittest.main()
148
149

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7/test_multi_threading.py
.
-----
Ran 1 test in 0.002s

OK
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```

150 # 8. Write a Python unit test program to check if a database connection is successful.
151
152 import sqlite3
153
154 class databaseConnectionTester(unittest.TestCase):
155     def test_database_connection(self):
156         conn = sqlite3.connect(':memory:')
157         cursor = conn.cursor()
158         cursor.execute("SELECT 10")
159         result = cursor.fetchone()
160         cursor.close()
161         conn.close()
162         self.assertEqual(result, (10,))
163
164 if __name__ == '__main__':
165     unittest.main()
166

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\Cwiczenia_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/Cwiczenia_7/test_database_connection.py
.
-----
Ran 1 test in 0.000s

OK
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\Cwiczenia_7>

```

```

167
168 # 9. Write a Python unit test program to check if a database query returns the expected results.
169
170 class testDatabaseQuery(unittest.TestCase):
171     def setUp(self):
172         self.conn = sqlite3.connect(':memory:')
173         self.cursor = self.conn.cursor()
174         self.cursor.execute("CREATE TABLE friends (id INTEGER PRIMARY KEY, name TEXT, surname TEXT)")
175         self.cursor.execute("INSERT INTO friends (name, surname) VALUES ('Marcin', 'Kowalski')")
176         self.cursor.execute("INSERT INTO friends (name, surname) VALUES ('Jan', 'Nowak')")
177         self.conn.commit()
178
179     def tearDown(self):
180         self.cursor.close()
181         self.conn.close()
182
183     def test_database_query(self):
184         self.cursor.execute("SELECT name, surname FROM friends ORDER BY name")
185         results = self.cursor.fetchall()
186
187         expected_results = [('Jan', 'Nowak'), ('Marcin', 'Kowalski')]
188
189         self.assertEqual(results, expected_results)
190
191 if __name__ == '__main__':
192     unittest.main()
193
194

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\Cwiczenia_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/Cwiczenia_7/test_database_query.py
.
-----
Ran 1 test in 0.000s

OK
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\Cwiczenia_7>

```

```

195 # 10. Write a Python unit test program to check if a function correctly parses and validates input data.
196
197 def parseAndValidate(data):
198     if isinstance(data, str) and data.isnumeric():
199         return int(data) > 0
200     return False
201
202 class testInputParsing(unittest.TestCase):
203     def test_valid_input(self):
204         data = "214892147721841289342178"
205         result = parseAndValidate(data)
206         self.assertTrue(result)
207
208     def test_invalid_input(self):
209         data = "Euro 2024"
210         result = parseAndValidate(data)
211         self.assertFalse(result)
212
213 if __name__ == '__main__':
214     unittest.main()
215

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/windows/Documents/GitHub/
..

```

```

-----
Ran 2 tests in 0.000s

```

OK

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

Python Exception Handling

(pythonexceptionhandling.py)

Obsługa wyjątków w Pythonie jest realizowana za pomocą konstrukcji 'try' i 'except', które pozwalają na przechwytywanie i reakcję na błędy podczas wykonywania programu. Dzięki temu możliwe jest zapobieganie awariom aplikacji i zapewnienie jej poprawnego działania w obliczu niespodziewanych sytuacji. Dodatkowo, konstrukcja 'raise' pozwala na wymuszenie wyjątków w celu sygnalizowania błędnych stanów.

```
pythonexceptionhandling.py > ...
1 # 1. Write a Python program to handle a ZeroDivisionError exception when dividing a number by zero.
2 def dividingByZero():
3     try:
4         number1 = int(input("Tell me a number you want to divide: "))
5         number2 = int(input("Tell me a number you want to divide by:"))
6         result = number1/number2
7         print(result)
8     except ZeroDivisionError:
9         print("You can't divide by zero")
10 dividingByZero()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/P
Tell me a number you want to divide: 5
Tell me a number you want to divide by:1
5.0
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/P
Tell me a number you want to divide: 5
Tell me a number you want to divide by:0
You can't divide by zero
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> █
```

```
11
12 # 2. Write a Python program that prompts the user to input
13 # an integer and raises a ValueError exception if the input is not a valid integer.
14 def valueError(prompt="Please enter an integer: "):
15     user_input = input(prompt)
16     try:
17         # Try to convert the input to an integer
18         value = int(user_input)
19         print(f"You entered the integer: {value}")
20         return value
21     except ValueError:
22         # Raise a ValueError if the input is not a valid integer
23         raise ValueError(f"Invalid input: '{user_input}' is not a valid integer")
24
25 if __name__ == "__main__":
26     try:
27         valueError()
28     except ValueError as e:
29         print(e)
30
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/D
Please enter an integer: z
Invalid input: 'z' is not a valid integer
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> █
```



```

20
21 # 3. Write a Python program that opens a file and handles a FileNotFoundError exception if the file does not exist.
22 def fileFinder(name):
23     try:
24         file = open(name, 'r')
25         contents = file.read()
26         print("Content:")
27         print(contents)
28         file.close()
29     except FileNotFoundError:
30         print("File does not exist")
31 fileFinder("folder")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_F
File does not exist
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```

33 # 4. Write a Python program that prompts the user to input
34 # two numbers and raises a TypeError exception if the inputs are not numerical.
35 def getInput(prompt):
36     user_input = input(prompt)
37     try:
38         value = float(user_input)
39         return value
40     except ValueError:
41         raise TypeError(f'{user_input} is not a number')
42
43 if __name__ == "__main__":
44     try:
45         num1 = getInput("First number:: ")
46         num2 = getInput("Second number: ")
47     except TypeError as e:
48         print(e)
49

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Win
First number:: 5
Second number: z
z is not a number
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```

60
61 # 5. Write a Python program that opens a file and handles a PermissionError exception if there is a permission issue.
62
63 def fileOpener(file_name):
64     try:
65         with open(file_name, 'r') as file:
66             content = file.read()
67             print(content)
68     except PermissionError:
69         print(f'Permission denied')
70         raise
71 fileOpener("plik.txt")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWAN
kijdxkdjxkd
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```

72
73 # 6. Write a Python program that executes an operation on a list and handles an IndexError exception if the index is out of range.
74 def rangeError():
75     myList = [1,2,3,4,5,6,7]
76     index = int(input("Number:"))
77     try:
78         number = myList[index]
79     except IndexError:
80         print("Number not in range")
81 rangeError()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7/rangeError.py
Number:10
Number not in range
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```

82
83 # 7. Write a Python program that prompts the user to input a number and handles a KeyboardInterrupt exception if the user cancels the input.
84 def userCancel():
85     try:
86         number = int(input("Give me a number:"))
87         print(number)
88     except KeyboardInterrupt:
89         print("Input has been cancelled by user")
90 userCancel()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7/userCancel.py
Give me a number:Input has been cancelled by user
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7/userCancel.py
Give me a number:5
5
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```

91
92 # 8. Write a Python program that executes division and handles an ArithmeticError exception if there is an arithmetic error.
93 def arithmeticError(number1, number2):
94     try:
95         result = number1/number2
96         print(f'{number1} divided by {number2} gives: {result}')
97     except ArithmeticError:
98         print("Arithmetic error occurred!")
99     arithmeticError(50,10)
100

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7/arithmeticError.py
Arithmetic error occurred!
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7/arithmeticError.py
50 divided by 10 gives: 5.0
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```

100
101 # 9. Write a Python program that opens a file and handles a UnicodeDecodeError exception if there is an encoding issue.
102 def read_file(file_name, encoding='utf-8'):
103     try:
104         with open(file_name, 'r', encoding=encoding) as file:
105             content = file.read()
106             print(content)
107     except UnicodeDecodeError:
108         print(f"An UnicodeDecodeError appeared!")
109 read_file(["example.txt"])
110

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7/read_file.py
An UnicodeDecodeError appeared!
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```
104 # 10. Write a Python program that executes a list operation and handles an AttributeError exception if the attribute does not exist.
105 def attributeError():
106     numbers = [1,2,3,4,5,6,7,8,9,0,100,20,3]
107     try:
108         r = len(numbers)
109         print(f'Length of the list: {r}')
110     except AttributeError:
111         print("An attribute error occurred")
112 attributeError()
113 def attributeError2():
114     numbers = [1,2,3,4,5,6,7,8,9,0,100,20,3]
115     try:
116         r = numbers.length
117         print(f'Length of the list: {r}')
118     except AttributeError:
119         print("An attribute error occurred")
120 attributeError2()
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7/
Length of the list: 13
An attribute error occurred
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> |
```

Python Asynchronous

([pythonasynchronous.py](#))

Programowanie asynchroniczne w Pythonie jest realizowane za pomocą modułu 'asyncio', który umożliwia pisanie współbieżnego kodu w prosty i czytelny sposób. Kluczowe są słowa 'async' i 'await', które kolejno pozwalają na deklarowanie funkcji asynchronicznych oraz wstrzymywanie ich wykonania. Dzięki temu można efektywnie zarządzać zasobami i poprawiać wydajność aplikacji w przypadku operacji sieciowych czy odczytu/zapisu danych.

```
pythonasynchronous.py > ...
1 # 1. Write a Python program that creates an asynchronous function to print "Python Exercises!" with a two second delay.
2 import asyncio
3 async def printSentence():
4     await asyncio.sleep(2)
5     print("Python Exercises!")
6 asyncio.run(printSentence())
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Users/Windows/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7/pythonasynchronous.py
Python Exercises!
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> |
```

```

7
8 # 2. Write a Python program that creates three asynchronous functions and
9 # displays their respective names with different delays (1 second, 2 seconds, and 3 seconds).
10
11 async def functionOne():
12     await asyncio.sleep(1)
13     print("functionOne")
14
15 async def functionTwo():
16     await asyncio.sleep(2)
17     print("functionTwo")
18
19 async def functionThree():
20     await asyncio.sleep(3)
21     print("functionThree")
22
23 asyncio.run(functionOne())
24 asyncio.run(functionTwo())
25 asyncio.run(functionThree())

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Users/Windows/AppData/Local/Programs/Python/Py
functionOne
functionTwo
functionThree
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> 

```

```

26
27 # 3. Write a Python program that creates an asyncio event loop and runs
28 # a coroutine that prints numbers from 1 to 7 with a delay of 1 second each.
29
30 async def numbers():
31     for i in range(1,8):
32         print(i)
33         await asyncio.sleep(1)
34 asyncio.run(numbers())

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Users/Windows/AppData/Local/Pr
1
2
3
4
5
6
7
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> 

```

```

36 # 4. Write a Python program that implements a coroutine to
37 # fetch data from two different URLs simultaneously using the "aiohttp" library.
38
39 import aiohttp
40
41 async def fetch_url(url):
42     async with aiohttp.ClientSession() as session:
43         async with session.get(url) as response:
44             return await response.text()
45
46 async def main():
47     urls = [
48         'https://www.youtube.com/',
49         'https://github.com/KasniQ/PODSTAWY_PROGRAMOWANIA/tree/main/CWICZENIA_7'
50     ]
51     url1 = asyncio.create_task(fetch_url(urls[0]))
52     url2 = asyncio.create_task(fetch_url(urls[1]))
53     data1 = await url1
54     data2 = await url2
55     print("Data from ", urls[0], len(data1), "bytes")
56     print("Data from ", urls[1], len(data2), "bytes")
57
58 asyncio.run(main())

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/
Data from https://www.youtube.com/ 540326 bytes
Data from https://github.com/KasniQ/PODSTAWY_PROGRAMOWANIA/tree/main/CWICZENIA_7 242138 bytes
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```

57
58 # 5. Write a Python program that runs multiple asynchronous tasks
59 # concurrently using asyncio.gather() and measures the time taken.
60 import time
61
62 async def taskOne():
63     print("Task 1")
64     await asyncio.sleep(5)
65     print("Task 1 Finished")
66
67 async def taskTwo():
68     print("Task 2")
69     await asyncio.sleep(2)
70     print("Task 2 Finished")
71
72 async def taskThree():
73     print("Task 3")
74     await asyncio.sleep(2)
75     print("Task 3 Finished")
76
77 async def main():
78     start = time.time()
79     await asyncio.gather(taskOne(), taskTwo(), taskThree())
80     finish = time.time()
81     timeTaken = finish - start
82     print("Time taken: ", timeTaken)
83
84 asyncio.run(main())
85
86

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PR
Task 1
Task 2
Task 3
Task 2 Finished
Task 3 Finished
Task 1 Finished
Time taken: 5.0086541175842285
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```

84
85 # 6. Write a Python program to create a coroutine that simulates
86 # a time-consuming task and use asyncio.CancelledError to handle task cancellation.
87 async def taskOne():
88     print("Task started")
89     try:
90         for i in range(10):
91             print(f"Thinking... {i+1}/10")
92             await asyncio.sleep(1)
93     except asyncio.CancelledError:
94         print("Task cancelled")
95         raise
96 async def main():
97     task = asyncio.create_task(taskOne())
98     await asyncio.sleep(5)
99     task.cancel()
100     try:
101         await task
102     except asyncio.CancelledError:
103         print("The task has been cancelled due to environmental problems.")
104     asyncio.run(main())
105

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7/task6.py
Task started
Thinking... 1/10
Thinking... 2/10
Thinking... 3/10
Thinking... 4/10
Thinking... 5/10
Task cancelled
The task has been cancelled due to environmental problems.
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```

106 # 7. Write a Python program that implements a timeout for an asynchronous operation using asyncio.wait_for().
107 async def anotherTask(seconds):
108     print(f'Starting task for {seconds} seconds')
109     await asyncio.sleep(seconds)
110     return f'Task has been completed in {seconds} seconds'
111 async def main():
112     timeout = 8
113     try:
114         result = await asyncio.wait_for(anotherTask(9), timeout)
115         print(result)
116     except asyncio.TimeoutError:
117         print(f'Timeout error occurred after {timeout} seconds')
118     asyncio.run(main())
119

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7> & C:/Python311/python.exe c:/Users/Windows/Documents/GitHub/PODSTAWY_PROGRAMOWANIA/CWICZENIA_7/task7.py
Starting task for 9 seconds
Timeout error occurred after 8 seconds
PS C:\Users\Windows\Documents\GitHub\PODSTAWY_PROGRAMOWANIA\CWICZENIA_7>

```

```

119
120 # 8. Write a Python program that uses asyncio queues to simulate
121 # a producer-consumer scenario with multiple producers and a single consumer.
122
123 async def producer(queue, id):
124     for i in range(5):
125         await asyncio.sleep(2)
126         item = f'Item: {id}--{i}'
127         await queue.put(item)
128         print(f'Producer {id} has produced -> {item}')
129 async def consumer(queue):
130     while True:
131         item = await queue.get()
132         if item is None:
133             break
134         print(f'Consumer took {item}')
135         await asyncio.sleep(1)
136         queue.task_done()
137 async def main():
138     queue = asyncio.Queue()
139     producers = [asyncio.create_task(producer(queue, i)) for i in range(5)]
140     consumer_task = asyncio.create_task(consumer(queue))
141     await asyncio.gather(*producers)
142     await queue.join()
143     await queue.put(None)
144     await consumer_task
145     asyncio.run(main())

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

Producer 1 has produced -> Item: 1--4
Producer 4 has produced -> Item: 4--4
Producer 2 has produced -> Item: 2--4
Consumer took Item: 1--1
Consumer took Item: 0--2
Consumer took Item: 1--2
Consumer took Item: 4--2
Consumer took Item: 3--2
Consumer took Item: 2--2
Consumer took Item: 0--3
Consumer took Item: 4--3
Consumer took Item: 2--3
Consumer took Item: 1--3
Consumer took Item: 3--3
Consumer took Item: 0--4
Consumer took Item: 3--4
Consumer took Item: 1--4
Consumer took Item: 4--4
Consumer took Item: 2--4

```

PS C:\Users\Windows\Documents\GitHub\PODSTAWY PROGRAMOWANIA\CWICZENIA > █

Spis treści

SPRAWOZDANIE	1
Python Unit Test.....	2
Python Exception Handling	8
Python Asynchronous.....	11