Formularz samooceny do projektu z języków skryptowych

Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max

1	UI	JEST	`menu = ConsoleMenu('Meteolizer', 'Analizator danych meteorologicznych', screen=scr)`
			
			`show_params = FunctionItem("Pokaż parametry programu", show_params_scr, [scr, args])`
			`mod_params = FunctionItem('Nadpisz parametry programu', modify_params_scr, [scr, args])`
			`draw_graph = FunctionItem('Narysuj graf na podstawie zaimportowanych danych', draw_graph_scr, [scr, args])`
			`import_data = FunctionItem('Zaimportuj dane CSV z pliku', import_data_scr, [scr, args])`
			`gen_data = FunctionItem('Wygeneruj dane do pliku CSV', gen_data_scr, [scr, args])`
			`modify_data = FunctionItem('Nadpisz dane zaimportowane z pliku CSV', modify_data_scr, [scr, args])`
			`show_data = FunctionItem('Pokaż zaimportowane dane CSV', show_data_scr, [scr, args])`
			
			`menu.append_item(show_params)`
			`menu.append_item(show_data)`
			`menu.append_item(mod_params)`
			`menu.append_item(draw_graph)`
			`menu.append_item(import_data)`
			`menu.append_item(gen_data)`
			`menu.append_item(modify_data)`
			
			`menu.show()`
				[redacted]		

		Wprowadzanie danych	`def modify_data_scr(scr, args):`
			`global record`
			
			`def insert_num(scr, args):`
			`global record`
			
			`data = record.get_data()`
			
			`vmax = len(data)`
				[redacted]		2

		Wyświetlanie danych	`def show_data_scr(scr, args):`
			`global record`
			
			`data = record.get_data()`
			
			`scr.println('Dane CSV')`
			`scr.println()`
			
			`for item in data:`
			`scr.println(f"| {item['pomiar']} | {item['temperatura']} | {item['wilgotnosc']} | {item['predkosc_wiatru']} |")`
			
			`scr.input()`
				[redacted]		2

		Zmiana danych	`def modify_data_scr(scr, args):`
			`global record`
			
			`def insert_num(scr, args):`
			`global record`
			
			`data = record.get_data()`
			
			`vmax = len(data)`
				[redacted]		2

		Wyszukiwanie danych	`def import_data_scr(scr, args):`
			`global record`
			
			`try:`
			`data = datagen.data_reader(args.csvfile)`
			
			`except FileNotFoundError:`
			`scr.println(f'Nie znaleziono pliku {args.csvfile} w katalogu {args.csvdir}!')`
			`scr.input()`
				[redacted]		2

			except (TypeError, OSError, ValueError):
			scr.println(f'Błąd odczytu pliku {args.csvfile} w katalogu {args.csvdir}!')
			else:
			record.set_data(data)
			
			scr.println(f'Udało się zaimportować dane z pliku {args.csvfile}!')
			scr.input()
			def gen_data_scr(scr, args):
			data = datagen.file_data_generator(args.n,
			
			(args.tmin, args.tmax),
			
			(args.hmin, args.hmax),
			
			(args.wmin, args.wmax))
			
			try:
			
			datagen.data_writer(args.csvdir + '/' + args.csvfile, data)
			except OSError:
			scr.println(f'Wystąpił problem z zapisaniem danych do pliku {args.csvfile}')
			except:
			scr.println(f'Wystąpił wyjątek')
			else:
			scr.println(f'Udało się zapisać dane do pliku {args.csvfile} w katalogu {args.csvdir}!')
			scr.input()			

		Przedstawienie wyników	def print_params(scr, args):
			scr.println('Parametry programu:')
			scr.println(f'Ilość pomiarów: {args.n}')
			scr.println(f'Minimalna temperatura: {args.tmin}')
			scr.println(f'Maksymalna temperatura: {args.tmax}')
			scr.println(f'Minimalna wilgotność: {args.hmin}')
			scr.println(f'Maksymalna wilgotność: {args.hmax}')
			scr.println(f'Maksymalna prędkość wiatru: {args.wmin}')
			scr.println(f'Maksymalna prędkość wiatru: {args.wmax}')
			scr.println(f'Domyślna ścieżka katalogu eksportu: {args.csvdir}')
			scr.println(f'Domyślna nazwa pliku eksportu: {args.csvfile}')
			def show_params_scr(scr, args):
			print_params(scr, args)
			scr.input('Kliknij Enter, aby wyjść')
				●		2

2	Podstawy	Zmienne	def main():
			# Definiujemy zakresy dla temperatury, wilgotności i wiatru
			temp_range = (20, 30)
			humidity_range = (30, 90) # wilgotność od 30% do 90%
			wind_range = (0, 20) # siła wiatru od 0 do 20 m/s
			
			data = file_data_generator(10, temp_range, humidity_range, wind_range)
			data_writer('dane.csv', data)
			
			if __name__ == '__main__':
			main()
				●		2

		typy danych	parser.add_argument('-n', type=int, help='Liczba pomiarów', default=30) parser.add_argument('--tmin', type=int, help='Minimalna wartość temperatury (w stopniach Celsjusza)', default=-40) parser.add_argument('--tmax', type=int, help='Maksymalna wartość temperatury (w stopniach Celsjusza)', default=50) parser.add_argument('--hmin', type=int, help='Minimalna wartość wilgotności (w procentach)', default=0) parser.add_argument('--hmax', type=int, help='Maksymalna wartość	●		2

			wilgotności (w procentach)', default=100) parser.add_argument('--wmin', type=int, help='Minimalna prędkość wiatru (m/s)', default=30) parser.add_argument('--wmax', type=int, help='Maksymalna prędkość wiatru (m/s)', default=300) parser.add_argument('--csv-file', type=str, help='Nazwa pliku do eksportu danych CSV', default='dane.csv') parser.add_argument('--csvdir', type=str, help='Ścieżka pliku do eksportu danych CSV', default=os.path.dirname(__file__)) args = parser.parse_args()			

		komentarze	def main():
			# Definiujemy zakresy dla temperatury, wilgotności i wiatru
			temp_range = (20, 30)
			humidity_range = (30, 90) # wilgotność od 30% do 90%
			wind_range = (0, 20) # siła wiatru od 0 do 20 m/s
			
			data = file_data_generator(10, temp_range, humidity_range, wind_range)
			data_writer('dane.csv', data)
			
			if __name__ == '__main__':
			main()
				[redacted]		1

		operatory	def get_standard_deviation(data):
			"""
			Oblicz odchylenie standardowe
			"""
			
			deviation = 0
			res = 0
			length = len(data)
			
			avg = get_arithmetic_average(data)
			
			for i in range(length):
			deviation += ((i - avg) ** 2)
			
			try:
			deviation /= length-1
			res = math.sqrt(deviation)
			except ZeroDivisionError:
			print(f'Wykryto dzielenie przez zero')
			except:
			print(f'Wykryto wyjątek!')
			finally:
			return res
				[redacted]		1,5

		Instrukcje warunkowe (if, elif, else)	if key == 'pomiar' or key == 'predkosc_wiatru':
			if new_value < 0:
			raise ValueError(f'Niepoprawna wartość: {new_value})')
			scr.input()
				[redacted]		3

		Instrukcje iteracyjne	num = [nm['pomiar'] for nm in data]	[redacted]		
		for	num = [nm['pomiar'] for nm in data]	[redacted]		2

		while	# Algorytm quicksort do sortowania wyników
			def partition(array, low, high):
			pivot = array[high].get('wartosc')
			
			i = low - 1
			
			j = low
			while j < high:
			if array[j].get('wartosc') <= pivot:
			i = i + 1
			(array[i], array[j]) = (array[j], array[i])
			j += 1
			
			(array[i + 1], array[high]) = (array[high], array[i + 1])
			return i + 1
				[redacted]		2

		Operacje wejścia (input)	def show_data_scr(scr, args): global record data = record.get_data()	[redacted]		1,5

			`scr.println('Dane CSV')`
			`scr.println()`
			
			`for item in data:`
			`scr.println(f"|{item['pomiar']} | {item['temperatura']} | {item['wilgotnosc']} |{item['predkosc_wiatru']} |")`
			
			`scr.input()`			

		Operacje wyjścia (print)	`def show_data_scr(scr, args):`
			`global record`
			
			`data = record.get_data()`
			
			`scr.println('Dane CSV')`
			`scr.println()`
			
			`for item in data:`
			`scr.println(f"|{item['pomiar']} | {item['temperatura']} | {item['wilgotnosc']} |{item['predkosc_wiatru']} |")`
			
			`scr.input()`
				●		1,5

		Funkcje z parametrami i wartościami zwracanymi	`def data_reader(filepath):`
			`"""`
			`Odczytaj plik CSV i zwróć jego zawartość w liście`
			`"""`
			`file_content = []`
			
			`with open(filepath, 'r') as f:`
			`reader = csv.DictReader(f)`
			
			`for line in reader:`
			`line['pomiar'] = int(line['pomiar'])`
			`line['temperatura'] = int(line['temperatura'])`
			`line['wilgotnosc'] = int(line['wilgotnosc'])`
			`line['predkosc_wiatru'] = int(line['predkosc_wiatru'])`
			`file_content.append(line)`
			
			`return file_content`
				●		2

		Funkcje rekurencyjne	`def quick_sort(array, low, high):`
			`if low < high:`
			`pi = partition(array, low, high)`
			`quick_sort(array, low, pi - 1)`
			`quick_sort(array, pi + 1, high)`
				●		3

		Funkcje przyjmujące inne funkcje jako argumenty	`menu = ConsoleMenu('Meteorolizer', 'Analizator danych meteorologicznych', screen=scr)`
			
			`show_params = FunctionItem("Pokaż parametry programu", show_params_scr, [scr, args])`
			`mod_params = FunctionItem('Nadpisz parametry programu', modify_params_scr, [scr, args])`
			`draw_graph = FunctionItem('Narysuj graf na podstawie zaimportowanych danych', draw_graph_scr, [scr, args])`
			`import_data = FunctionItem('Zaimportuj dane CSV z pliku', import_data_scr, [scr, args])`
			`gen_data = FunctionItem('Wygeneruj dane do pliku CSV', gen_data_scr, [scr, args])`
			`modify_data = FunctionItem('Nadpisz dane zaimportowane z pliku CSV', modify_data_scr, [scr, args])`
			`show_data = FunctionItem('Pokaż zaimportowane dane CSV', show_data_scr, [scr, args])`
			
			`menu.append_item(show_params)`
			`menu.append_item(show_data)`
			`menu.append_item(mod_params)`
			`menu.append_item(draw_graph)`
			`menu.append_item(import_data)`
			`menu.append_item(gen_data)`
			`menu.append_item(modify_data)`
			
			`menu.show()`
				●		3

		Dekoratory	`@profile`
			`def test_get_celsius():`
			`assert asys.get_celsius(1.0) == 33.8`
				●		1,5

Formularz samooceny do projektu z języków skryptowych

			`@profile`
			`def test_get_fahrenheit():`
			`  assert int(asys.get_fahrenheit(1.0)) == -17`			

3	Kontenery	Użycie listy	record = models.Record(list())	☒		2

		Użycie słownika	`def quick_sort_perf_test():`
			`  start = timeit.default_timer()`
			`  for _ in range(10):`
			`  data = [{ 'pomiar': 0, 'wartosc': -20.0 },{ 'pomiar': 1, 'wartosc': -10.0 },`
			`  { 'pomiar': 2, 'wartosc': -81.0 },{ 'pomiar': 3, 'wartosc': 24.0 },`
			`  { 'pomiar': 4, 'wartosc': 2.0 },{ 'pomiar': 5, 'wartosc': 0.0 }]`
			`  asys.quick_sort(data, 0, len(data)-1)`
			`  stop = timeit.default_timer()`
			`  print(asys.quick_sort.__name__ + ': Czas wykonania: ', stop - start)`
				☒		2

		Użycie zbioru		☐		1,5

		Użycie krotki	`def main():`
			`  # Definiujemy zakresy dla temperatury, wilgotności i wiatru`
			`  temp_range = (20, 30)`
			`  humidity_range = (30, 90)  # wilgotność od 30% do 90%`
			`  wind_range = (0, 20)  # siła wiatru od 0 do 20 m/s`
			
			`  data = file_data_generator(10, temp_range, humidity_range, wind_range)`
			`  data_writer('dane.csv', data)`
			
			`if __name__ == '__main__':`
			`  main()`
				☒		1,5

4	Przestrzenie nazw	Zastosowano zmienne lokalne	`def main():`
			`  # Definiujemy zakresy dla temperatury, wilgotności i wiatru`
			`  temp_range = (20, 30)`
			`  humidity_range = (30, 90)  # wilgotność od 30% do 90%`
			`  wind_range = (0, 20)  # siła wiatru od 0 do 20 m/s`
			
			`  data = file_data_generator(10, temp_range, humidity_range, wind_range)`
			`  data_writer('dane.csv', data)`
			
			`if __name__ == '__main__':`
			`  main()`
				☒		1,5

		Zastosowano zmienne globalne	`global record`	☒		1,5

		Zastosowano zakresy funkcji	`  def modify_params_scr(scr, args):`
			`  def modify_n(scr, args):`
			`  try:`
			`  new_n = int(scr.input('Podaj nową wartość liczby pomiarów: '))`
			
			`  if new_n <= 0:`
			`  raise ValueError(f'Zbyt mała liczba pomiarów: {new_n}')`
			`  except (TypeError, ValueError) as e:`
			`  print(e)`
			`  else:`
			`  args.n = new_n`
			
			`scr.println(f'Nadpisano wartość liczby pomiarów: {args.n}')`
			
			`scr.input('Kliknij Enter, aby wyjść')`
				☒		1,5

		Zastosowano zakresy klas	`# Nazwy wielkości`
			`class Record:`
			`  '''Pomiar'''`
			`  class Station:`
			`  '''Stacja badawcza'''`
			`  # name: nazwa stacji`
			`  # city: nazwa miasta`
			`  # quantity: mierzona wielkość fizyczna (temperatura)`
			`  def __init__(self):`
			`  pass`
				☒		1,5

			`def set_name(self, new_name):`
			`  self.name = new_name`
			`def set_city(self, new_city):`
			`  self.city = new_city`
			`def set_quantity(self, new_quantity):`
			`  self.quantity = new_quantity`
			`def get_name(self):`
			`  return self.name`
			`def get_city(self):`
			`  return self.city`
			`def get_quantity(self):`
			`  return self.quantity`
			`def __init__(self, data):`
			`  self.data = data.copy()`
			`  self.station = self.Station()`
			`def set_station_name(self, new_name):`
			`self.station.set_name(new_name)`
			`def set_station_city(self, new_city):`
			`self.station.set_city(new_city)`
			`def set_station_quantity(self, new_quantity):`
			`self.station.set_quantity(new_quantity)`
			`def get_station_name(self):`
			`  return self.station.get_name() if hasattr(self.station, 'name') else None`
			`def get_station_city(self):`
			`  return self.station.get_city() if hasattr(self.station, 'city') else None`
			`def get_station_quantity(self):`
			`  return self.station.get_quantity() if hasattr(self.station, 'quantity') else None`
			`def set_data(self, new_data):`
			`  self.data = new_data.copy()`
			`def get_data(self):`
			`  return self.data.copy()`			

5	Moduły i pakiety	Projekt podzielony na moduły (import, __init__)	`import utils.data_generator as datagen`
			`import utils.analysis as asys`
			`import utils.models as models`
				●		2

Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max

		Własne pakiety/funkcje pomocnicze w osobnych plikach .py	`import utils.data_generator as datagen`
			`import utils.analysis as asys`
			`import utils.models as models`
				●		2

6	Obsługa błędów	Obsługa wyjątków (try, except, finally)	`def modify_wmin(scr, args):`
			`  try:`
			`  new_wmin = int(scr.input('Podaj nową najmniejszą prędkość wiatru (w m/s): '))`
			`  if new_wmin < 0:`
			`scr.println('Niepoprawna prędkość wiatru {new_wmin} m/s')`
			`  scr.input()`
			`  raise ValueError('Niepoprawa prędkość wiatru: {new_wmin} m/s')`
			`  except (TypeError, ValueError) as e:`
			`  print(e)`
			`  else:`
			`  args.wmin = new_wmin`
			`scr.println(f'Nadpisano najmniejszą prędkość wiatru: {args.wmin} m/s')`
				●		2

			`scr.input('Kliknij Enter, aby wyjść')`			

		Użycie assert do testów i walidacji	`def test_get_arithmetic_average():`
			`  random_data = []`
			
			`  def gen_random_data():`
			`  nonlocal random_data`
			
			`  for i in range(10):`
			`  random_data.append(ran-dom.randint(-100, 100))`
			
			`  gen_random_data()`
			`  tmp_data = random_data.copy()`
			
			`  avg1 = asys.get_arithmetic_aver-age(random_data)`
			`  avg2 = np.average(np.array(tmp_data))`
			
			`  assert avg1 == avg2`
				☐		1,5

7	Łańcuchy znaków	Operacje na stringach (m.in. formatowanie, dzielenie, wyszukiwanie)	`  def gen_data_scr(scr, args):`
			`  data = datagen.file_data_generator(args.n,`
			
			`(args.tmin, args.tmax),`
			
			`(args.hmin, args.hmax),`
			
			`(args.wmin, args.wmax))`
			
			`  try:`
			
			`datagen.data_writer(args.csvdir + '/' + args.csvfile, data)`
			`  except OSError:`
			`  scr.println(f'Wys-tąpił problem z zapisaniem danych do pliku {args.csvfile}')`
			`  except:`
			`  scr.println(f'Wys-tąpił wyjątek')`
			`  else:`
			`  scr.println(f'Udało się zapisać dane do pliku {args.csv-file} w katalogu {args.csvdir}!')`
			`  scr.input()`
				☐		2

8	Obsługa plików	Odczyt plików .txt, .csv, .json, .xml (min. 1)	`def data_reader(filepath):`
			`  """`
			`  Odczytaj plik CSV i zwróć jego zawartość w liście`
			`  """`
			`  file_content = []`
			
			`  with open(filepath, 'r') as f:`
			`  reader = csv.DictReader(f)`
			
			`  for line in reader:`
			`  line['pomiar'] = int(line['pomiar'])`
			`  line['temperatura'] = int(line['temperatura'])`
			`  line['wilgotnosc'] = int(line['wilgotnosc'])`
			`  line['predkosc_wiatru'] = int(line['predkosc_wiatru'])`
			`  file_content.append(line)`
			
			`  return file_content`
				☐		2

		Zapis do plików .txt .csv, .json, .xml (min. 1)	`def data_writer(filepath, data):`
			`  """`
			`  Utwórz plik CSV i wpisz do niego dane meteorologiczne`
			`  """`
			`  with open(filepath, 'w', newline='') as f:`
			`  writer = csv.writer(f)`
			`  writer.writerow(['pomiar', 'temperatura', 'wilgotnosc', 'pred-kosc_wiatru'])`
			`  writer.writerows(data)`
				☐		2

9	OOP	Klasy	`# Nazwy wielkości`
			`class Record:`
			`  '''Pomiar'''`
			`  class Station:`
			`  '''Stacja badawcza'''`
			`  # name: nazwa stacji`
			`  # city: nazwa miasta`
			`  # quantity: mierzona wielkość fizyczna (temperatura)`
			`  def __init__(self):`
			`  pass`
			`  def set_name(self, new_name):`
			`  self.name = new_name`
			`  def set_city(self, new_city):`
			`  self.city = new_city`
			`  def set_quantity(self, new_quantity):`
			`  self.quantity = new_quan-tity`
				☐		2

Formularz samooceny do projektu z języków skryptowych

			def get_name(self):
			return self.name
			def get_city(self):
			return self.city
			def get_quantity(self):
			return self.quantity
			def __init__(self, data):
			self.data = data.copy()
			self.station = self.Station()
			def set_station_name(self, new_name):
			self.station.set_name(new_name)
			def set_station_city(self, new_city):
			self.station.set_city(new_city)
			def set_station_quantity(self, new_quantity):
			self.station.set_quantity(new_quantity)
			def get_station_name(self):
			return self.station.get_name() if hasattr(self.station, 'name') else None
			def get_station_city(self):
			return self.station.get_city() if hasattr(self.station, 'city') else None
			def get_station_quantity(self):
			return self.station.get_quantity() if hasattr(self.station, 'quantity') else None
			def set_data(self, new_data):
			self.data = new_data.copy()
			def get_data(self):
			return self.data.copy()			

		Metody	def set_name(self, new_name):
			self.name = new_name
			def set_city(self, new_city):
			self.city = new_city
			def set_quantity(self, new_quantity):
			self.quantity = new_quantity
			def get_name(self):
			return self.name
			def get_city(self):
			return self.city
			def get_quantity(self):
			return self.quantity
				☐		2

		Konstruktory	def __init__(self):
			pass
			def __init__(self, data):
			self.data = data.copy()
				☐		2

		Dziedziczenie	class test_model(unittest.TestCase):
			
			def setUp(self):
			self.record = models.Record(list())
			
			self.record.set_station_name('Stacja')
			
			self.record.set_station_city('Kielce')
			self.record.set_station_quantity('Temperatura')
			def test_get_station_name(self):
			self.assertEqual(self.record.get_station_name(), 'Stacja')
			def test_get_station_city(self):
			self.assertEqual(self.record.get_station_city(), 'Kielce')
			def test_get_station_quantity(self):
			
			self.assertEqual(self.record.get_station_quantity(), 'Temperatura')
				☐		2

10	Programowa-nie funkcyjne	map		☐		1,5
		filter		☐		1,5
		lambda	dat_sum = reduce(lambda x, y: x +	☐		1,5

Formularz samooceny do projektu z języków skryptowych

			`y, [ x for x in data ])`			
		reduce	`dat_sum = reduce(lambda x, y: x + y, [ x for x in data ])`	●		1,5

11	Wizualizacja danych	Wygenerowano wykres (np. Matplotlib, seaborn)	`def draw_graph_scr(scr, args):`
			`global record`
			
			`try:`
			`data = record.get_data()`
			
			`if not data:`
			`scr.println('Nie zaimportowano danych!')`
			`scr.input()`
			
			`raise ValueError('Nie zaimportowano danych!')`
			`except ValueError as e:`
			`print(e)`
			`else:`
			`num = [ nm['pomiar'] for nm in data ]`
			`temp = [ tp['temperatura'] for tp in data ]`
			`hum = [ hm['wilgotnosc'] for hm in data ]`
			`wind = [ wn['predkosc_wiatru'] for wn in data ]`
			
			`plt.plot(num, temp, color='r', label='Temperatura [°C]')`
			`plt.plot(num, hum, color='g', label='Wilgotność [%]')`
			`plt.plot(num, wind, color='b', label='Prędkość wiatru [m/s]')`
			`plt.legend()`
			`plt.xlabel('Numer pomiaru')`
			
			`plt.ylabel('Wielkości')`
			`plt.title('Wykres danych meteorologicznych')`
			
			`plt.show()`
				●		2

		Zapisano wykres do pliku graficznego (.png lub .jpg)	`def draw_graph_scr(scr, args):`
			`global record`
			
			`try:`
			`data = record.get_data()`
			
			`if not data:`
			`scr.println('Nie zaimportowano danych!')`
			`scr.input()`
			
			`raise ValueError('Nie zaimportowano danych!')`
			`except ValueError as e:`
			`print(e)`
			`else:`
			`num = [ nm['pomiar'] for nm in data ]`
			`temp = [ tp['temperatura'] for tp in data ]`
			`hum = [ hm['wilgotnosc'] for hm in data ]`
			`wind = [ wn['predkosc_wiatru'] for wn in data ]`
			
			`plt.plot(num, temp, color='r', label='Temperatura [°C]')`
			`plt.plot(num, hum, color='g', label='Wilgotność [%]')`
			`plt.plot(num, wind, color='b', label='Prędkość wiatru [m/s]')`
			`plt.legend()`
			`plt.xlabel('Numer pomiaru')`
			
			`plt.ylabel('Wielkości')`
			`plt.title('Wykres danych meteorologicznych')`
			
			`plt.show()`
				●		1,5

T12	Testowanie	Testy (assert, unittest, pytest)	`class test_model(unittest.TestCase):`
			`def setUp(self):`
			`self.record = models.Record(list())`
			
			`self.record.set_station_name('Stacja')`
			
			`self.record.set_station_city('Kielce'`
				●		1,5

			)
			self.record.set_station_quantity('Temperatura')
			def test_get_station_name(self):
			self.assertEqual(self.record.get_station_name(), 'Stacja')
			def test_get_station_city(self):
			self.assertEqual(self.record.get_station_city(), 'Kielce')
			def test_get_station_quantity(self):
			self.assertEqual(self.record.get_station_quantity(), 'Temperatura')			

		Testy funkcjonalne	def test_get_arithmetic_average():
			random_data = []
			
			def gen_random_data():
			nonlocal random_data
			
			for i in range(10):
			random_data.append(random.randint(-100, 100))
			
			gen_random_data()
			tmp_data = random_data.copy()
			
			avg1 = asys.get_arithmetic_average(random_data)
			avg2 = np.average(np.array(tmp_data))
			
			assert avg1 == avg2
				☒		1,5

		Testy Integracyjne	class test_model(unittest.TestCase):
			def setUp(self):
			self.record = models.Record(list())
			
			self.record.set_station_name('Stacja')
			
			self.record.set_station_city('Kielce')
			self.record.set_station_quantity('Temperatura')
			def test_get_station_name(self):
			self.assertEqual(self.record.get_station_name(), 'Stacja')
			def test_get_station_city(self):
			self.assertEqual(self.record.get_station_city(), 'Kielce')
			def test_get_station_quantity(self):
			self.assertEqual(self.record.get_station_quantity(), 'Temperatura')
				☒		1,5

		Testy graniczne / błędne dane		☐		1,5

		Testy wydajności (np. czas wykonania timeit)	@profile
			def get_arthm_average_perf_test():
			start = timeit.default_timer()
			for _ in range(10):
			asys.get_arithmetic_average([10, 20, 30, 40, 50, 60, 70, 80, 90])
			stop = timeit.default_timer()
			print(asys.get_arithmetic_average.__name__ + ': Czas wykonania: ', stop - start)
				☒		1,5

		Testy pamięci memory_profiler	@profile
			def test_get_celsius():
			assert asys.get_celsius(1.0) == 33.8
			
			@profile
			def test_get_fahrenheit():
			assert int(asys.get_fahrenheit(1.0)) == -17
				☒		1,5

		Test jakości kodu (flake8, pylint)	Plik setup.cfg biblioteki flake8 :	☒		1,5

			[flake8] exclude = .git,.gitignore,*.pyc,*. png,*.jpg,*.csv max-line-length = 119			
13	Wersjono- wanie	Repozytorium GIT	Git init	●		1
		Historia commitów	commit b20bbf23b914cc2996d64f5b39b- bc3e1fc584f83 (HEAD -> main, origin/ main) Author: TKundera <s096148@student.- tu.kielce.pl> Date: Sun Jun 29 09:01:06 2025 +0200 Dodanie przeglądania danych CSV. Aktualizacja README	●		1
Nr	Obszar	Wymaganie	KOD		Przyzna- ne pkt	Pkt max
		Link do GitHub	https://orkan.- tu.kielce.pl/git- lab/TKundera/js- projekt/-/tree/ main? ref_type=heads	●		1
		Opis commitów	commit b20bbf23b914c- c2996d64f5b39bbc3e1fc584f83 (HEAD -> main, origin/main) Author: TKundera <s096148@s- tudent.tu.kielce.pl> Date: Sun Jun 29 09:01:06 2025 +0200 Dodanie przeglądania danych CSV. Aktualizacja README	●		1
14	Dokumenta- cja	Plik README.md (cel, autorzy, uruchamianie)	# Projekt z przedmiotu Języki Skryptowe ## Temat nr 25: Analizator danych meteorologicznych ### Wykonali - Wojciech Lis - Tomasz Kundera ### Zależności - `console-menu` - `matplotlib` - `memory_profiler` - `numpy` ### Krótka instrukcja użytkowania ``` usage: Meteolizer [options] Analizator danych meteoro- logicznych options: -h, --help show this help message and exit -n N Liczba pomiarów --tmin TMIN Mini- malna wartość temperatury (w stopniach Celsjusza) --tmax TMAX Maksy- malna wartość temperatury (w stopniach Celsjusza) --hmin HMIN Mini- malna wartość wilgotności (w procentach) --hmax HMAX Maksy- malna wartość wilgotności (w procentach) --wmin WMIN Mini- malna prędkość wiatru (m/s) --wmax WMAX Maksy- malna prędkość wiatru (m/s) --csvfile CSVFILE Nazwa pliku do eksportu danych CSV --csvdir CSVDIR Ścieżka pliku do eksportu danych CSV	●		1,5

Formularz samooceny do projektu z języków skryptowych

			Wykonany przez: Wojciech Lis, Tomasz Kundera ```			
		Przykładowe dane wejściowe i wyjściowe	TAK	☑		2
		Diagram klas lub struktura modułów	### Struktura projektu ``` . ├── CHANGELOG.md ├── misc │ ├── input_data.csv │ ├── output_graph.png │ └── output_graph_sort-ed.png ├── README.md ├── setup.cfg └── src ├── dane.csv ├── meteolizer.py ├── test.csv ├── tests │ ├── test_analysis.py │ ├── test_data_gener-ator.py │ └── test_models.py └── utils ├── analysis.py ├── data_genera-tor.py ├── __init__.py └── models.py ```	☑		2
			SUMA			