In [30]:
```
from gamspy import (
    Container,Set,Alias,Parameter,Variable,Equation,Model,Problem,Sense,Opti
    Domain,Number,Sum,Product,Smax,Smin,Ord,Card,ModelStatus,SpecialValues
)
```

In [35]:
```
import numpy as np
from gamspy import (
    Container, Set, Parameter, Equation
)

# Initialize GAMS container
m = Container()

# Set of months from January (1) to December (12)
items = Set(m, 'items', records=['Living', 'Rent', 'Car', 'Mobile', 'Muggle'
months = Set(m, 'months', records=[str(i + 1) for i in range(12)])
# Parameters for monthly expenses

allowance = 150
dentistry = 1900

leisure = Variable(m, 'leisure', domain=[months], description='Monthly amour

# Parameters for enjoyment and bills
billpaid = Parameter(
    m,
    name="billpaid",
    domain=[items],
    description="bills to be paid",
    domain_forwarding=True,
    records=[["living", 550], ["rent", 650], ["car", 340], ["mobile", 135],
)

enjoy = Parameter(m, 'enjoy', domain=[months], records = [[1, 12],[2, 22],[3
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
Cell In[35], line 20
     17 leisure = Variable(m, 'leisure', domain=[months], description='Month
ly amount spend on leisure')
     19 # Parameters for enjoyment and bills
---> 20 billpaid = Parameter(
     21     m,
     22     name="billpaid",
     23     domain=[items],
     24     description="bills to be paid",
     25     domain_forwarding=True,
     26     records=[["living", 550], ["rent", 650], ["car", 340], ["mobil
e", 135], ["muggle", 100], ["gas_electric", 850]]
     27 )
     29 enjoy = Parameter(m, 'enjoy', domain=[months], records = [[1, 12],
[2, 22],[3, 30],[4, 36],[5, 40],[6, 42],[7, 42],[8, 40],[9, 36],[10, 30],[1
1, 22],[12, 12]])

File ~/CS524/venv/lib/python3.10/site-packages/gamspy/_symbols/parameter.py:
245, in Parameter.__init__(self, container, name, domain, records, domain_fo
rwarding, description, uels_on_axes, is_miro_input, is_miro_output, is_miro_
table)
    242 self.container._add_statement(self)
    244 if records is not None:
--> 245     self.setRecords(records, uels_on_axes=uels_on_axes)
    246 else:
    247     self.container._synch_with_gams()

File ~/CS524/venv/lib/python3.10/site-packages/gamspy/_symbols/parameter.py:
429, in Parameter.setRecords(self, records, uels_on_axes)
    405 """
    406 Main convenience method to set standard pandas.DataFrame formatted
    407 records. If uels_on_axes=True setRecords will assume that all domain
   (...)
    425
    426 """
    427 super().setRecords(records, uels_on_axes)
--> 429 self.container._synch_with_gams()
    430 self._winner = "python"

File ~/CS524/venv/lib/python3.10/site-packages/gamspy/_container.py:532, in
Container._synch_with_gams(self)
    530 def _synch_with_gams(self) -> DataFrame | None:
    531     runner = backend_factory(self, self._options)
--> 532     summary = runner.run()
    534     if self._options and self._options.seed is not None:
    535         # Required for correct seeding. Seed can only be set in the
first run.
    536         self._options.seed = None

File ~/CS524/venv/lib/python3.10/site-packages/gamspy/_backend/local.py:88,
in Local.run(self)
     80     self.options._set_solver_options(
     81         working_directory=self.container.working_directory,
     82         solver=self.solver,
```

```
     83         problem=self.model.problem,
     84         solver_options=self.solver_options,
     85     )
     87 # Generate gams string and write modified symbols to gdx
---> 88 gams_string = self.preprocess(self.container._gdx_in)
     90 # Run the model
     91 self.execute_gams(gams_string)

File ~/CS524/venv/lib/python3.10/site-packages/gamspy/_backend/backend.py:149, in Backend.preprocess(self, gdx_in)
    146 modified_names = self.container._get_touched_symbol_names()
    148 if len(modified_names) != 0:
--> 149     self.container.write(
    150         self.container._gdx_in, modified_names, eps_to_zero=False
    151     )
    153 gams_string = self.container._generate_gams_string(
    154     gdx_in, modified_names
    155 )
    157 self.update_modified_state(modified_names)

File ~/CS524/venv/lib/python3.10/site-packages/gamspy/_container.py:688, in Container.write(self, write_to, symbol_names, compress, mode, eps_to_zero)
    660 def write(
    661     self,
    662     write_to: str,
   (...)
    666     eps_to_zero: bool = True,
    667 ) -> None:
    668     """
    669     Writes specified symbols to the GDX file. If symbol_names are
    670     not provided, it writes all symbols to the GDX file.
   (...)
    686
    687     """
--> 688     super().write(
    689         write_to,
    690         symbol_names,
    691         compress,
    692         mode=mode,
    693         eps_to_zero=eps_to_zero,
    694     )

File ~/CS524/venv/lib/python3.10/site-packages/gams/transfer/containers/_container.py:1782, in Container.write(self, write_to, symbols, compress, uel_priority, merge_symbols, mode, eps_to_zero)
   1774     raise TypeError(
   1775         "Argument 'write_to' expects "
   1776         "type str or Pathlike object (i.e., a path to a GDX file) "
   1777         "or a valid gmdHandle (or GamsDatabase instance) "
   1778         f"User passed: '{type(write_to)}'."
   1779     )
   1781 if dest is DestinationType.GDX:
-> 1782     self._gdx_write(
   1783         write_to, symbols, uel_priority, compress, mode, eps_to_zero
   1784     )
   1786 if dest is DestinationType.GMD:
```

```
   1787     self._gmd_write(
   1788         write_to,
   1789         symbols,
   (...)
   1793         eps_to_zero,
   1794     )

File ~/CS524/venv/lib/python3.10/site-packages/gams/transfer/containers/_container.py:1603, in Container._gdx_write(self, write_to, symbols, uel_priority, compress, mode, eps_to_zero)
   1602 def _gdx_write(self, write_to, symbols, uel_priority, compress, mode, eps_to_zero):
-> 1603     return io.gdx.container_write(
   1604         self, write_to, symbols, uel_priority, compress, mode, eps_to_zero
   1605     )

File ~/CS524/venv/lib/python3.10/site-packages/gams/transfer/containers/_io/gdx.py:798, in container_write(container, write_to, symbols, uel_priority, compress, mode, eps_to_zero)
    795         os.remove(write_to)
    797     # raise error
--> 798     raise err
    800 finally:
    801     # restore domains in the Container
    802     for _, properties in was_relaxed.items():

File ~/CS524/venv/lib/python3.10/site-packages/gams/transfer/containers/_io/gdx.py:777, in container_write(container, write_to, symbols, uel_priority, compress, mode, eps_to_zero)
    774         current_error_count = gdx.gdxDataErrorCount(gdxHandle)
    776         if current_error_count != 0:
--> 777             raise Exception(
    778                 f"Encountered data errors with symbol `{symname}`. "
    779                 "Possible causes are from duplicate records and/or domain violations. \n\n"
    780                 "Use 'hasDuplicateRecords', 'findDuplicateRecords', 'dropDuplicateRecords', "
    781                 "and/or 'countDuplicateRecords' to find/resolve duplicate records. \n"
    782                 "Use 'hasDomainViolations', 'findDomainViolations', 'dropDomainViolations', "
    783                 "and/or 'countDomainViolations' to find/resolve domain violations. \n\n"
    784                 "GDX file was not created successfully."
    785             )
    787 except Exception as err:
    788     # close file
    789     gdx.gdxClose(gdxHandle)

Exception: Encountered data errors with symbol `items`. Possible causes are from duplicate records and/or domain violations.

Use 'hasDuplicateRecords', 'findDuplicateRecords', 'dropDuplicateRecords', and/or 'countDuplicateRecords' to find/resolve duplicate records.
Use 'hasDomainViolations', 'findDomainViolations', 'dropDomainViolations', a
```

```
nd/or 'countDomainViolations' to find/resolve domain violations.

GDX file was not created successfully.
```

In [28]:

```python
from gamspy import (
    Container, Set, Parameter, Variable, Equation, Model, Problem, Sense, Su
)

# Initialize GAMS container
m = Container()

# Set of months from January (1) to December (12)
months = Set(m, 'months', records=[str(i + 1) for i in range(12)])

# Parameters for monthly expenses
allowance = 150  # Monthly child allowance
dentistry = 1900  # Monthly dentistry income
total_income = allowance + dentistry  # Total monthly income

# Leisure expenses variable (must be at least £165 per month)
leisure = Variable(m, 'leisure', 'positive', [months], description='Monthly

# Parameters for fixed bills (monthly values)
fixed_expenses = Parameter(
    m,
    'fixed_expenses',
    domain=[months],
    records=[
        [str(i + 1), 550 + 650 + 340] for i in range(12)
    ],  # Living expense, rent, and car
    description="Fixed monthly expenses"
)

# Additional expenses (occur periodically)
mobile_plan_cost = Parameter(m, 'mobile_plan_cost', domain=[months], records
    [str(i + 1), 135 if (i + 1) % 2 == 0 else 0] for i in range(12)
])
gas_electricity_cost = Parameter(m, 'gas_electricity_cost', domain=[months],
    [str(i + 1), 850 if (i + 1) % 6 == 0 else 0] for i in range(12)
])
muggle_tax_cost = Parameter(m, 'muggle_tax_cost', domain=[months], records=[
    [str(i + 1), 100 if (i + 1) % 4 == 0 else 0] for i in range(12)
])

# Parameter for enjoyment based on month index and leisure spending
enjoy = Parameter(m, 'enjoy', domain=[months], records = [[1, 12],[2, 22],[3


# Equation to ensure the total expenses do not exceed the total monthly inco
budget_constraint = Equation(m, 'budget_constraint', domain=[months])
budget_constraint[months] = (
    leisure[months] + fixed_expenses[months] +
    mobile_plan_cost[months] + gas_electricity_cost[months] +
    muggle_tax_cost[months] <= total_income
)
```

```
# Ensure that leisure expenses are at least £165 per month
leisure_minimum = Equation(m, 'leisure_minimum', domain=[months])
leisure_minimum[months] = leisure[months] >= 165

# Model setup to maximize total enjoyment
budget_model = Model(
    m,
    name='budget_model',
    equations=[budget_constraint, leisure_minimum],
    problem=Problem.LP,
    sense=Sense.MAX,
    objective=Sum([months], enjoy[months])  # Maximize the total enjoyment
)

# Solve the model
budget_model.solve()

# Output the results
```

Out[28]:

	Solver Status	Model Status	Objective	Num of Equations	Num of Variables	Model Type	Solver
0	Normal	InfeasibleNoSolution	NA	25	13	LP	CPLEX

In [ ]: