

Universidad Americana

*Facultad de Ingeniería y Arquitectura*



Algoritmos y Estructuras de Datos, Grupo 4

**Manejo de Funciones, Clases y Paquetes**

Realizado por:

*Joaquín Alberto Pérez Zúñiga*

*Nicolás Rafael Laguna Vallejos*

Docente:

*MSc. César Marín López*

## 1. main.py:

```
> nu      main.py U X  _init_.py U  manejo_gastos.py U  gasto.py U
semana-4 > prueba-1 > main.py > ...
24  from locale import LC_TIME, setlocale
25  from os import system
26
27  from control_gastos import (
28      ...agregar_gasto,
29      ...encontrar_total,
30      ...Gasto,
31      ...mostrar_gastos,
32      ...promediar_categoria,
33  )
34
35
36  def menu_principal(gastos: list[Gasto]) -> None:
37      """
38      ...Loop principal del programa que le muestra un menú de opciones al usuario.
39      ..."""
40
41      ...opcion = ""
42      ...while opcion != "7":
43          ...system("cls || clear")
44          ...print("\nGestión de Gastos")
45          ...print("-----\n")
46          ...print("1. Agregar gasto")
47          ...print("2. Registrar nueva categoría de gasto")
48          ...print("3. Mostrar gastos de un mes")
49          ...print("4. Mostrar gastos de una categoría")
50          ...print("5. Calcular totales")
51          ...print("6. Promediar gastos de categoría")
52          ...print("7. Salir")
53
54          ...opcion = input("\n-> ")
55
```

> nu

main.py U X

\_init\_.py U

manejo\_gastos.py U

gasto.py U

semana-4 > prueba-1 > main.py > ...

```
36 def menu_principal(gastos: list[Gasto]) -> None:
56     .... match opcion:
57     .... case "1":
58     ....     system("cls || clear")
59     ....     agregar_gasto(gastos)
60     .... case "2":
61     ....     system("cls || clear")
62     ....     print("\nCategorías Registradas:")
63     ....     print("*****\n")
64
65     .... for i, cat in enumerate(Gasto.categorias()):
66     ....     print(f"{i + 1}. {cat}")
67
68     .... print("\nIngrese la nueva categoría:")
69     .... input(f"{len(Gasto.categorias())}. ")
70     .... case "3":
71     ....     system("cls || clear")
72     ....     try:
73     ....         mes = int(
74     ....             input(
75     ....                 "\nIngrese el número del mes de gastos que desea visualizar: "
76     ....             )
77     ....         )
78     ....         if mes <= 0 or mes > 12:
79     ....             raise ValueError
80
81     ....         mostrar_gastos(gastos, mes)
82     ....     except (TypeError, ValueError):
83     ....         input("¡Error! Debe ingresar un número de mes válido (1-12).")
84     ....         continue
85     .... case "4":
86     ....     system("cls || clear")
87     ....     cat = input("\nIngrese la categoría de gastos que desea visualizar: ")
88     ....     if not Gasto.validar_categoria(cat):
89     ....         input("¡Debe ingresar una categoría registrada!")
90     ....     continue
```

>- nu

main.py U X

\_init\_.py U

manejo\_gastos.py U

gasto.py U

semana-4 > prueba-1 > main.py > ...

```
36 def menu_principal(gastos: list[Gasto]) -> None:
85     ..... case "4":
86     .....     system("cls || clear")
87     .....     cat = input("\nIngrese la categoría de gastos que desea visualizar: ")
88     .....     if not Gasto.validar_categoria(cat):
89     .....         input("¡Debe ingresar una categoría registrada!")
90     .....         continue
91
92     .....     mostrar_gastos(gastos, cat)
93     ..... case "5":
94     .....     system("cls || clear")
95     .....     tipo = input(
96     .....         "\n¿Desea encontrar el total de gastos de un mes (1)"
97     .....         + " o de una categoría (2)? (1/2)"
98     .....     )
99
100    .....     if tipo == "1":
101    .....         try:
102    .....             mes = int(
103    .....                 input(
104    .....                     "\nIngrese el número del mes de gastos que desea visualizar: "
105    .....                 )
106    .....             )
107    .....             if mes <= 0 or mes > 12:
108    .....                 raise ValueError
109
110    .....             encontrar_total(gastos, mes)
111    .....         except (TypeError, ValueError):
112    .....             input("¡Error! Debe ingresar un número de mes válido (1-12).")
113    .....             continue
114    .....     elif tipo == "2":
115    .....         cat = input("\nIngrese la categoría de gastos que desea visualizar: ")
116    .....         if not Gasto.validar_categoria(cat):
117    .....             input("¡Debe ingresar una categoría registrada!")
118    .....             continue
119
```

> nu

main.py U X

\_init\_.py U

manejo\_gastos.py U

gasto.py U

semana-4 > prueba-1 > main.py > ...

```
36 def menu_principal(gastos: list[Gasto]) -> None:
114     ... elif tipo == "2":
115     ...     cat = input("\nIngrese la categoría de gastos que desea visualizar: ")
116     ...     if not Gasto.validar_categoria(cat):
117     ...         input("¡Debe ingresar una categoría registrada!")
118     ...         continue
119
120     ...     encontrar_total(gastos, cat)
121     ...     else:
122     ...         input("\n¡Error! Debe ingresar una respuesta inválida.")
123     ...     case "6":
124     ...         system("cls || clear")
125     ...         cat = input("\nIngrese la categoría de gastos que desea visualizar: ")
126     ...         if not Gasto.validar_categoria(cat):
127     ...             input("¡Debe ingresar una categoría registrada!")
128     ...             continue
129
130     ...         promediar_categoria(gastos, cat)
131     ...     case "7":
132     ...         print("\nSaliendo del programa...\n")
133     ...     case _:
134     ...         input("\n¡Opción inválida, intente nuevamente!")
135
136
137 def main() -> None:
138     ... """
139     ... Ejecución del programa.
140     ... """
141
142     ... gastos: list[Gasto] = []
143     ... menu_principal(gastos)
144
145
146 if __name__ == "__main__":
147     ... setlocale(LC_TIME, "es_ES")
148     ... main()
```

## 2. `__init__.py`:

```
> nu      main.py U  _init_.py U X  manejo_gastos.py U  gasto.py U
semana-4 > prueba-1 > control_gastos > _init_.py > ...
1  """
2  Aplicación que permite a un usuario, principalmente estudiantes,
3  registrar sus gastos para llevar mejor control sobre ellos.
4  """
5
6  from .gasto import Gasto
7  from .manejo_gastos import (
8      ...agregar_gasto,
9      ...encontrar_total,
10     ...mostrar_gastos,
11     ...promediar_categoria
12 )
13
14
15 __all__ = [
16     ..."Gasto",
17     ..."agregar_gasto",
18     ..."encontrar_total",
19     ..."mostrar_gastos",
20     ..."promediar_categoria"
21 ]
22
```

### 3. gasto.py:

```
semana-4 > prueba-1 > control_gastos > gasto.py > ...
1  """
2  Implementación de la clase Gasto.
3  """
4
5  from datetime import date
6
7
8  class Gasto:
9      """
10     Representa un gasto monetario mensual.
11     """
12
13     _categorias = [
14         "alimentos",
15         "renta",
16         "transporte",
17     ]
18
19     def __init__(
20         self,
21         codigo: int,
22         fecha: date,
23         categoria: str,
24         monto: float,
25         descripcion: str
26     ):
27         self._codigo = codigo
28         self._fecha = fecha
29         self._categoria = categoria
30         self._monto = monto
31         self._descripcion = descripcion
32
```

> nu

main.py U

\_\_init\_\_.py U

manejo\_gastos.py U

gasto.py U X

semana-4 > prueba-1 > control\_gastos > gasto.py > ...

```
8 class Gasto:
33     @property
34     def fecha(self) -> date:
35         """
36         Getter method para acceder al atributo privado _fecha.
37         """
38
39         return self._fecha
40
41     @property
42     def categoria(self) -> str:
43         """
44         Getter method para acceder al atributo privado _categoria.
45         """
46
47         return self._categoria
48
49     @property
50     def monto(self) -> float:
51         """
52         Getter method para acceder al atributo privado _monto.
53         """
54
55         return self._monto
56
57     @property
58     def descripcion(self) -> str:
59         """
60         Getter method para acceder al atributo privado _descripcion.
61         """
62
63         return self._descripcion
64
```



> nu X main.py U \_init\_.py U manejo\_gastos.py U gasto.py U X

semana-4 > prueba-1 > control\_gastos > gasto.py > ...

```
8 class Gasto:
65     @staticmethod
66     def categorias() -> list[str]:
67         """
68         Getter method para acceder al atributo privado _categorias.
69         """
70
71         return Gasto._categorias
72
73     @classmethod
74     def agregar_categoria(cls, cat_nueva: str) -> None:
75         """
76         Agrega la categoría dada a la lista de categorías válidas.
77         """
78
79         cls._categorias.append(cat_nueva)
80
81     @classmethod
82     def validar_categoria(cls, cat: str) -> bool:
83         """
84         Valida si un string es una categoría de gasto válida.
85         """
86
87         return cat in cls._categorias
88
```

#### 4. manejo\_gastos.py:

```
>- nu  main.py U  _init_.py U  manejo_gastos.py U X  gasto.py U
semana-4 > prueba-1 > control_gastos > manejo_gastos.py > ...
1  """
2  Implementaciones de funciones que permiten manejar
3  los gastos registrados en la aplicación.
4  """
5
6  from datetime import date
7  from os import system
8  from typing import Union
9
10 from . import Gasto
11
12
13 def agregar_gasto(gastos: list[Gasto]) -> None:
14     """
15     ... Pide los datos de un producto nuevo y lo agrega a productos.
16     ... """
17
18     ... print("\nAgregar Gasto Nuevo:")
19     ... print("*****")
20
21     ... print("\nIngresa los detalles del gasto:")
22     ... try:
23     ...     fecha = date.fromisoformat(
24     ...         input("Fecha (en formato yyyy-mm-dd, o vacío para fecha actual): ")
25     ...     )
26
27     ...     cat = input("Categoría: ")
28     ...     if not Gasto.validar_categoria(cat):
29     ...         raise ValueError("¡Debe ingresar una categoría registrada!")
30
31     ...     monto = float(input("Monto: C$"))
32     ...     if monto <= 0:
33     ...         raise ValueError("¡El precio debe ser un número real positivo!")
34
35     ...     desc = input("Descripción: ")
36
```

> nu

main.py U

\_init\_.py U

manejo\_gastos.py U X

gasto.py U

semana-4 > prueba-1 > control\_gastos > manejo\_gastos.py > ...

```
13 def agregar_gasto(gastos: list[Gasto]) -> None:
37     except (TypeError, ValueError) as e:
38         if "isoformat" not in str(e) or isinstance(e, TypeError):
39             input(f"\n{e}")
40         else:
41             input("\n¡Error! La fecha debe ser ingresada en el formato yyyy-dd-mm.")
42
43     system("cls || clear")
44     return agregar_gasto(gastos)
45
46     gastos.append(Gasto(len(gastos) - 1, fecha, cat, monto, desc))
47     input("\n¡Gasto agregado exitosamente!")
48     return None
49
50
51 def mostrar_gastos(gastos: list[Gasto], filtro: Union[int, str, None] = None) -> None:
52     """
53     Muestra los gastos que coinciden con el filtro de
54     mes o categoría (None para mostrar todos los gastos).
55     """
56
57     if filtro is None:
58         tipo = "Registrados"
59         gastos_filtrados = gastos
60     elif isinstance(filtro, int):
61         if filtro <= 0 or filtro > 12:
62             input("¡Error! Debe ingresar un número de mes válido (1-12).")
63             return
64
65         tipo = f"de {date(2025, filtro, 1).strftime("%B")}"
66         gastos_filtrados = [g for g in gastos if g.fecha.month == filtro]
67     else:
68         if not Gasto.validar_categoria(filtro):
69             input("¡Debe ingresar una categoría registrada!")
70             return
71
```

> nu

main.py U

\_init\_.py U

manejo\_gastos.py U X

gasto.py U

semana-4 > prueba-1 > control\_gastos > manejo\_gastos.py > ...

```
51 def mostrar_gastos(gastos: list[Gasto], filtro: Union[int, str, None] = None) -> None:
52     tipo = f"de {filtro}"
53     gastos_filtrados = [g for g in gastos if g.categoria == filtro]
54
55     print(f"\nGastos {tipo}:")
56     print("*****")
57
58     for i, gasto in enumerate(gastos_filtrados):
59         print(f"\nGasto #{i + 1}:")
60         print("-----")
61         print(f"Fecha: {gasto.fecha}")
62         print(f"Categoría: {gasto.categoria}")
63         print(f"Monto: C${gasto.monto}")
64         print(f"Descripción: {gasto.descripcion}")
65         print("-----")
66
67     input("\nPresione 'Enter' para regresar al menú principal...")
68
69 def encontrar_total(gastos: list[Gasto], filtro: Union[int, str, None] = None) -> None:
70     """
71     Encuentra el total de los gastos filtrados:
72     -> si filtro es un int, encuentra el total de gastos del mes que corresponde al int
73     -> si filtro es un str, encuentra el total de gastos de la categoría que corresponde
74     -> si filtro es None, encuentra el total de todos los gastos registrados.
75     """
76
77     if filtro is None:
78         tipo_gasto = "todos los gastos registrados"
79         gastos_filtrados = gastos
80     elif isinstance(filtro, int):
81         tipo_gasto = f"todos los gastos de de {date(2025, filtro, 1).strftime("%B")}"
82         gastos_filtrados = [g for g in gastos if g.fecha.month == filtro]
83     else:
84         tipo_gasto = f"de todos los gastos de {filtro}"
85         gastos_filtrados = [g for g in gastos if g.categoria == filtro]
```

> nu

main.py U

\_init\_.py U

manejo\_gastos.py U X

gasto.py U

semana-4 > prueba-1 > control\_gastos > manejo\_gastos.py > ...

```
90 def encontrar_total(gastos: list[Gasto], filtro: Union[int, str, None] = None) -> None:
91     """
92     Encuentra el total de los gastos filtrados:
93     -> si filtro es un int, encuentra el total de gastos del mes que corresponde al int
94     -> si filtro es un str, encuentra el total de gastos de la categoría que corresponde
95     -> si filtro es None, encuentra el total de todos los gastos registrados.
96     """
97
98     if filtro is None:
99         tipo_gasto = "todos los gastos registrados"
100         gastos_filtrados = gastos
101     elif isinstance(filtro, int):
102         tipo_gasto = f"todos los gastos de de {date(2025, filtro, 1).strftime('%B')}}"
103         gastos_filtrados = [g for g in gastos if g.fecha.month == filtro]
104     else:
105         tipo_gasto = f"de todos los gastos de {filtro}"
106         gastos_filtrados = [g for g in gastos if g.categoria == filtro]
107
108     total = sum(g.monto for g in gastos_filtrados)
109     print(f"\nTotal de {tipo_gasto}: C$ {total}")
110     input("\nPresione 'Enter' para regresar al menú principal...")
111
112
113 def promediar_categoria(gastos: list[Gasto], cat: str) -> None:
114     """
115     Encuentra el promedio de los gastos registrados en la categoría dada.
116     """
117
118     promedio = sum(g.monto for g in gastos if g.categoria == cat) / len(gastos)
119     print(f"\nPromedio de todos los gastos de {cat}: C$ {promedio}")
120     input("\nPresione 'Enter' para regresar al menú principal...")
121
```

## Output:

```
>- python  X
○
Gestión de Gastos
-----

1. Agregar gasto
2. Registrar nueva categoría de gasto
3. Mostrar gastos de un mes
4. Mostrar gastos de una categoría
5. Calcular totales
6. Promediar gastos de categoría
7. Salir

-> |
```

```
>- python  X
○
Categorías Registradas:
*****

1. alimentos
2. renta
3. transporte

Ingrese la nueva categoría:
4. internet|
```

```
>- python  X
○
Agregar Gasto Nuevo:
*****

Ingrese los detalles del gasto:
Fecha (en formato yyyy-mm-dd, o vacío para fecha actual): 2025-02-04
Categoría: transporte
Monto: C$50
Descripción: Bus para ir a clase

¡Gasto agregado exitosamente!|
```

> python X

○  
Agregar Gasto Nuevo:

\*\*\*\*\*

Ingrese los detalles del gasto:

Fecha (en formato yyyy-mm-dd, o vacío para fecha actual):

Categoría: alimentos

Monto: C\$200

Descripción: Almuerzo

¡Gasto agregado exitosamente! |

> python X

○  
Ingrese el número del mes de gastos que desea visualizar: 2

Gastos de febrero:

\*\*\*\*\*

Gasto #1:

-----

Fecha: 2025-02-04

Categoría: transporte

Monto: C\$50.0

Descripción: Bus para ir a clase

-----

Presione 'Enter' para regresar al menú principal... |

> python X

○  
Ingrese la categoría de gastos que desea visualizar: transporte

Gastos de transporte:

\*\*\*\*\*

Gasto #1:

-----

Fecha: 2025-02-04

Categoría: transporte

Monto: C\$50.0

Descripción: Bus para ir a clase

-----

Presione 'Enter' para regresar al menú principal... |