

---

```
1 # Represents a student with multiple variables
2
3 name = input("Name: ")
4 house = input("House: ")
5 print(f"{name} from {house}")
```

```
1 # Modularizes getting student's name and house
2
3
4 def main():
5     name = get_name()
6     house = get_house()
7     print(f"{name} from {house}")
8
9
10 def get_name():
11     return input("Name: ")
12
13
14 def get_house():
15     return input("House: ")
16
17
18 if __name__ == "__main__":
19     main()
```

```
1 # Returns student as tuple, unpacking it
2
3
4 def main():
5     name, house = get_student()
6     print(f"{name} from {house}")
7
8
9 def get_student():
10    name = input("Name: ")
11    house = input("House: ")
12    return name, house
13
14
15 if __name__ == "__main__":
16    main()
```

```
1 # Returns student as tuple, without unpacking it
2
3
4 def main():
5     student = get_student()
6     print(f"{student[0]} from {student[1]}")
7
8
9 def get_student():
10    name = input("Name: ")
11    house = input("House: ")
12    return (name, house)
13
14
15 if __name__ == "__main__":
16    main()
```

```
1 # Demonstrates immutability of tuples, removes parentheses
2 # https://scifi.stackexchange.com/q/105992
3
4
5 def main():
6     student = get_student()
7     if student[0] == "Padma":
8         student[1] = "Ravenclaw"
9     print(f"{student[0]} from {student[1]}")
10
11
12 def get_student():
13     name = input("Name: ")
14     house = input("House: ")
15     return name, house
16
17
18 if __name__ == "__main__":
19     main()
```

```
1 # Stores student as (mutable) list
2
3
4 def main():
5     student = get_student()
6     if student[0] == "Padma":
7         student[1] = "Ravenclaw"
8     print(f"{student[0]} from {student[1]}")
9
10
11 def get_student():
12     name = input("Name: ")
13     house = input("House: ")
14     return [name, house]
15
16
17 if __name__ == "__main__":
18     main()
```

```
1 # Stores student as dict
2
3
4 def main():
5     student = get_student()
6     print(f"{student['name']} from {student['house']}")
7
8
9 def get_student():
10    student = {}
11    student["name"] = input("Name: ")
12    student["house"] = input("House: ")
13    return student
14
15
16 if __name__ == "__main__":
17    main()
```

```
1 # Eliminates unneeded variable
2
3
4 def main():
5     student = get_student()
6     print(f"{student['name']} from {student['house']}")
7
8
9 def get_student():
10    name = input("Name: ")
11    house = input("House: ")
12    return {"name": name, "house": house}
13
14
15 if __name__ == "__main__":
16    main()
```



```
1 # Demonstrates mutability of dicts
2
3
4 def main():
5     student = get_student()
6     if student["name"] == "Padma":
7         student["house"] = "Ravenclaw"
8     print(f"{student['name']} from {student['house']}")
9
10
11 def get_student():
12     name = input("Name: ")
13     house = input("House: ")
14     return {"name": name, "house": house}
15
16
17 if __name__ == "__main__":
18     main()
```

```
1 # Defines class for a student
2
3
4 class Student:
5     ...
6
7
8 def main():
9     student = get_student()
10    print(f"{student.name} from {student.house}")
11
12
13 def get_student():
14    student = Student()
15    student.name = input("Name: ")
16    student.house = input("House: ")
17    return student
18
19
20 if __name__ == "__main__":
21    main()
```

```
1 # Adds __init__
2
3
4 class Student:
5     def __init__(self, name, house):
6         self.name = name
7         self.house = house
8
9
10 def main():
11     student = get_student()
12     print(f"{student.name} from {student.house}")
13
14
15 def get_student():
16     name = input("Name: ")
17     house = input("House: ")
18     student = Student(name, house)
19     return student
20
21
22 if __name__ == "__main__":
23     main()
```

```
1 # Eliminates unneeded variable
2
3
4 class Student:
5     def __init__(self, name, house):
6         self.name = name
7         self.house = house
8
9
10 def main():
11     student = get_student()
12     print(f"{student.name} from {student.house}")
13
14
15 def get_student():
16     name = input("Name: ")
17     house = input("House: ")
18     return Student(name, house)
19
20
21 if __name__ == "__main__":
22     main()
```

```
1 # Adds validation in __init__ using raise
2
3
4 class Student:
5     def __init__(self, name, house):
6         if not name:
7             raise ValueError("Missing name")
8         if house not in ["Gryffindor", "Hufflepuff", "Ravenclaw", "Slytherin"]:
9             raise ValueError("Invalid house")
10        self.name = name
11        self.house = house
12
13
14 def main():
15     student = get_student()
16     print(f"{student.name} from {student.house}")
17
18
19 def get_student():
20     name = input("Name: ")
21     house = input("House: ")
22     return Student(name, house)
23
24
25 if __name__ == "__main__":
26     main()
```

```
1 # Prints student without __str__
2
3
4 class Student:
5     def __init__(self, name, house):
6         if not name:
7             raise ValueError("Missing name")
8         if house not in ["Gryffindor", "Hufflepuff", "Ravenclaw", "Slytherin"]:
9             raise ValueError("Invalid house")
10        self.name = name
11        self.house = house
12
13
14 def main():
15     student = get_student()
16     print(student)
17
18
19 def get_student():
20     name = input("Name: ")
21     house = input("House: ")
22     return Student(name, house)
23
24
25 if __name__ == "__main__":
26     main()
```

```
1 # Adds __str__
2
3
4 class Student:
5     def __init__(self, name, house):
6         if not name:
7             raise ValueError("Missing name")
8         if house not in ["Gryffindor", "Hufflepuff", "Ravenclaw", "Slytherin"]:
9             raise ValueError("Invalid house")
10        self.name = name
11        self.house = house
12
13        def __str__(self):
14            return f"{self.name} from {self.house}"
15
16
17 def main():
18     student = get_student()
19     print(student)
20
21
22 def get_student():
23     name = input("Name: ")
24     house = input("House: ")
25     return Student(name, house)
26
27
28 if __name__ == "__main__":
29     main()
```

```
1 # Prompts for patronus too, but doesn't display yet
2
3
4 class Student:
5     def __init__(self, name, house, patronus):
6         if not name:
7             raise ValueError("Missing name")
8         if house not in ["Gryffindor", "Hufflepuff", "Ravenclaw", "Slytherin"]:
9             raise ValueError("Invalid house")
10        self.name = name
11        self.house = house
12        self.patronus = patronus
13
14        def __str__(self):
15            return f"{self.name} from {self.house}"
16
17
18 def main():
19     student = get_student()
20     print(student)
21
22
23 def get_student():
24     name = input("Name: ")
25     house = input("House: ")
26     patronus = input("Patronus: ")
27     return Student(name, house, patronus)
28
29
30 if __name__ == "__main__":
31     main()
```



```
1 # Adds charm method to cast a charm
2
3
4 class Student:
5     def __init__(self, name, house, patronus=None):
6         if not name:
7             raise ValueError("Missing name")
8         if house not in ["Gryffindor", "Hufflepuff", "Ravenclaw", "Slytherin"]:
9             raise ValueError("Invalid house")
10        if patronus and patronus not in ["Stag", "Otter", "Jack Russell terrier"]:
11            raise ValueError("Invalid patronus")
12        self.name = name
13        self.house = house
14        self.patronus = patronus
15
16    def __str__(self):
17        return f"{self.name} from {self.house}"
18
19    def charm(self):
20        match self.patronus:
21            case "Stag":
22                return "🐎"
23            case "Otter":
24                return "🦦"
25            case "Jack Russell terrier":
26                return "🐕"
27            case _:
28                return "🪄"
29
30
31 def main():
32     student = get_student()
33     print("Expecto Patronum!")
34     print(student.charm())
35
36
37 def get_student():
38     name = input("Name: ")
39     house = input("House: ")
40     patronus = input("Patronus: ") or None
41     return Student(name, house, patronus)
42
```

---

```
43  
44 if __name__ == "__main__":  
45     main()
```

```
1 # Removes patronus for simplicity, circumvents error-checking by setting attribute
2
3
4 class Student:
5     def __init__(self, name, house):
6         if not name:
7             raise ValueError("Invalid name")
8         if house not in ["Gryffindor", "Hufflepuff", "Ravenclaw", "Slytherin"]:
9             raise ValueError("Invalid house")
10        self.name = name
11        self.house = house
12
13        def __str__(self):
14            return f"{self.name} from {self.house}"
15
16
17 def main():
18     student = get_student()
19     student.house = "Number Four, Privet Drive"
20     print(student)
21
22
23 def get_student():
24     name = input("Name: ")
25     house = input("House: ")
26     return Student(name, house)
27
28
29 if __name__ == "__main__":
30     main()
```

```
1 # Adds @property for house
2
3
4 class Student:
5     def __init__(self, name, house):
6         if not name:
7             raise ValueError("Invalid name")
8         self.name = name
9         self.house = house
10
11     def __str__(self):
12         return f"{self.name} from {self.house}"
13
14     @property
15     def house(self):
16         return self._house
17
18     @house.setter
19     def house(self, house):
20         if house not in ["Gryffindor", "Hufflepuff", "Ravenclaw", "Slytherin"]:
21             raise ValueError("Invalid house")
22         self._house = house
23
24
25 def main():
26     student = get_student()
27     print(student)
28
29
30 def get_student():
31     name = input("Name: ")
32     house = input("House: ")
33     return Student(name, house)
34
35
36 if __name__ == "__main__":
37     main()
```

```
1 # Adds @property for name
2
3
4 class Student:
5     def __init__(self, name, house):
6         self.name = name
7         self.house = house
8
9     def __str__(self):
10        return f"{self.name} from {self.house}"
11
12    @property
13    def name(self):
14        return self._name
15
16    @name.setter
17    def name(self, name):
18        if not name:
19            raise ValueError("Invalid name")
20        self._name = name
21
22    @property
23    def house(self):
24        return self._house
25
26    @house.setter
27    def house(self, house):
28        if house not in ["Gryffindor", "Hufflepuff", "Ravenclaw", "Slytherin"]:
29            raise ValueError("Invalid house")
30        self._house = house
31
32
33 def main():
34     student = get_student()
35     print(student)
36
37
38 def get_student():
39     name = input("Name: ")
40     house = input("House: ")
41     return Student(name, house)
42
```

---

```
43  
44 if __name__ == "__main__":  
45     main()
```

---

```
1 # Prints the type of an integer
2
3 print(type(50))
```

---

```
1 # Prints the type of a string
2
3 print(type("hello, world"))
```



---

```
1 # Prints the type of a list
2
3 print(type([]))
```

---

```
1 # Prints the type of a list
2
3 print(type(list()))
```

---

```
1 # Prints the type of a dictionary
2
3 print(type({}))
```

---

```
1 # Prints the type of a dictionary
2
3 print(type(dict()))
```

```
1 # Implements sort() with an instance method
2
3 import random
4
5
6 class Hat:
7     def __init__(self):
8         self.houses = ["Gryffindor", "Hufflepuff", "Ravenclaw", "Slytherin"]
9
10    def sort(self, name):
11        print(name, "is in", random.choice(self.houses))
12
13
14 hat = Hat()
15 hat.sort("Harry")
```

```
1 # Implements sort() with a class method
2
3 import random
4
5
6 class Hat:
7
8     houses = ["Gryffindor", "Hufflepuff", "Ravenclaw", "Slytherin"]
9
10    @classmethod
11    def sort(cls, name):
12        print(name, "is in", random.choice(cls.houses))
13
14
15 Hat.sort("Harry")
```

```
1 # Moves get_student into Student class
2
3
4 class Student:
5     def __init__(self, name, house):
6         self.name = name
7         self.house = house
8
9     def __str__(self):
10        return f"{self.name} from {self.house}"
11
12    @classmethod
13    def get(cls):
14        name = input("Name: ")
15        house = input("House: ")
16        return cls(name, house)
17
18
19 def main():
20     student = Student.get()
21     print(student)
22
23
24 if __name__ == "__main__":
25     main()
```

```
1 # Demonstrates inheritance [maybe don't add `if` error-checking]
2
3
4 class Wizard:
5     def __init__(self, name):
6         if not name:
7             raise ValueError("Missing name")
8         self.name = name
9
10    ...
11
12
13 class Student(Wizard):
14     def __init__(self, name, house):
15         super().__init__(name)
16         self.house = house
17
18    ...
19
20
21 class Professor(Wizard):
22     def __init__(self, name, subject):
23         super().__init__(name)
24         self.subject = subject
25
26    ...
27
28
29 wizard = Wizard("Albus")
30 student = Student("Harry", "Gryffindor")
31 professor = Professor("Severus", "Defense Against the Dark Arts")
32 ...
```



```
1 # Adds vaults, storing total in new vault
2
3
4 class Vault:
5     def __init__(self, galleons=0, sickles=0, knuts=0):
6         self.galleons = galleons
7         self.sickles = sickles
8         self.knuts = knuts
9
10    def __str__(self):
11        return f"{self.galleons} Galleons, {self.sickles} Sickles, {self.knuts} Knuts"
12
13
14 potter = Vault(100, 50, 25)
15 print(potter)
16
17 weasley = Vault(25, 50, 100)
18 print(weasley)
19
20 galleons = potter.galleons + weasley.galleons
21 sickles = potter.sickles + weasley.sickles
22 knuts = potter.knuts + weasley.knuts
23
24 total = Vault(galleons, sickles, knuts)
25 print(total)
```

```
1 # Adds vaults via operator overloading
2
3
4 class Vault:
5     def __init__(self, galleons=0, sickles=0, knuts=0):
6         self.galleons = galleons
7         self.sickles = sickles
8         self.knuts = knuts
9
10    def __str__(self):
11        return f"{self.galleons} Galleons, {self.sickles} Sickles, {self.knuts} Knuts"
12
13    def __add__(self, other):
14        galleons = self.galleons + other.galleons
15        sickles = self.sickles + other.sickles
16        knuts = self.knuts + other.knuts
17        return Vault(galleons, sickles, knuts)
18
19
20 potter = Vault(100, 50, 25)
21 print(potter)
22
23 weasley = Vault(25, 50, 100)
24 print(weasley)
25
26 total = potter + weasley
27 print(total)
```