BASIC PYTHON

INTRODUCTION PYTHON
សេចក្តីណែនាំ PYTHON

Python គឺជា ភាសាកូដ high-level មួយ ដែលត្រូវបានគេស្គាល់ថា ជាភាសាកូដ ដែលងាយស្រួល, សាមញ្ញ និង មិនពិបាកក្នុងការស្វែងយល់។ Python ត្រូវបានគេប្រើក្នុងការ develop លើ Web, អនុវត្តទៅលើ Data Science, Data Analysis ហើយនិងអនុវត្តលើផ្នែក AI ផងដែរ។

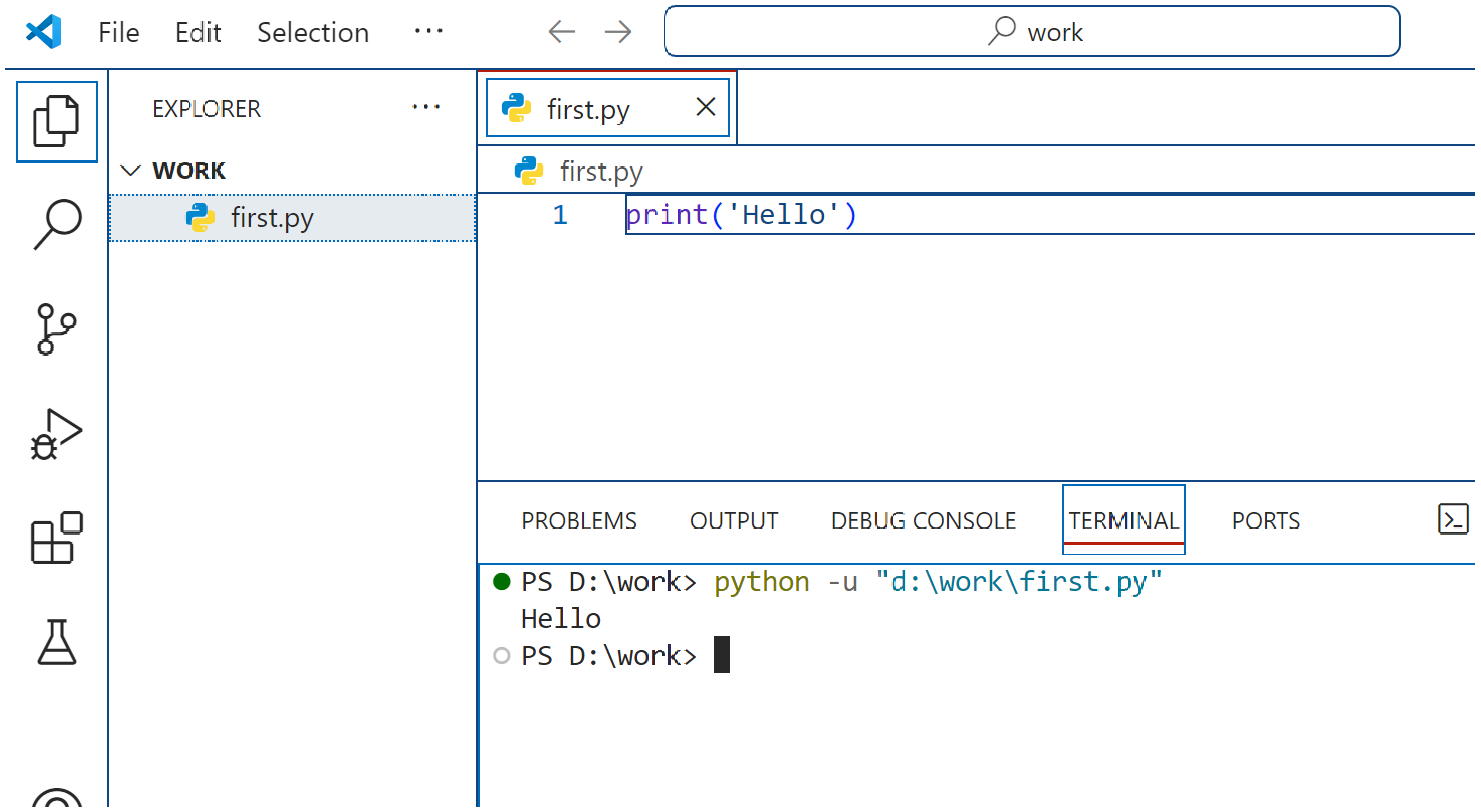
Python ត្រូវបានបង្កើតឡើងដោយ Guido van Rossum ហើយត្រូវបានគេដាក់អោយប្រើជាសាធារណៈ នៅឆ្នាំ 1991 ។

SETTING UP PYTHON
ការដំឡើង PYTHON

Link សម្រាប់ដំឡើង Vs code:
https://code.visualstudio.com/Download

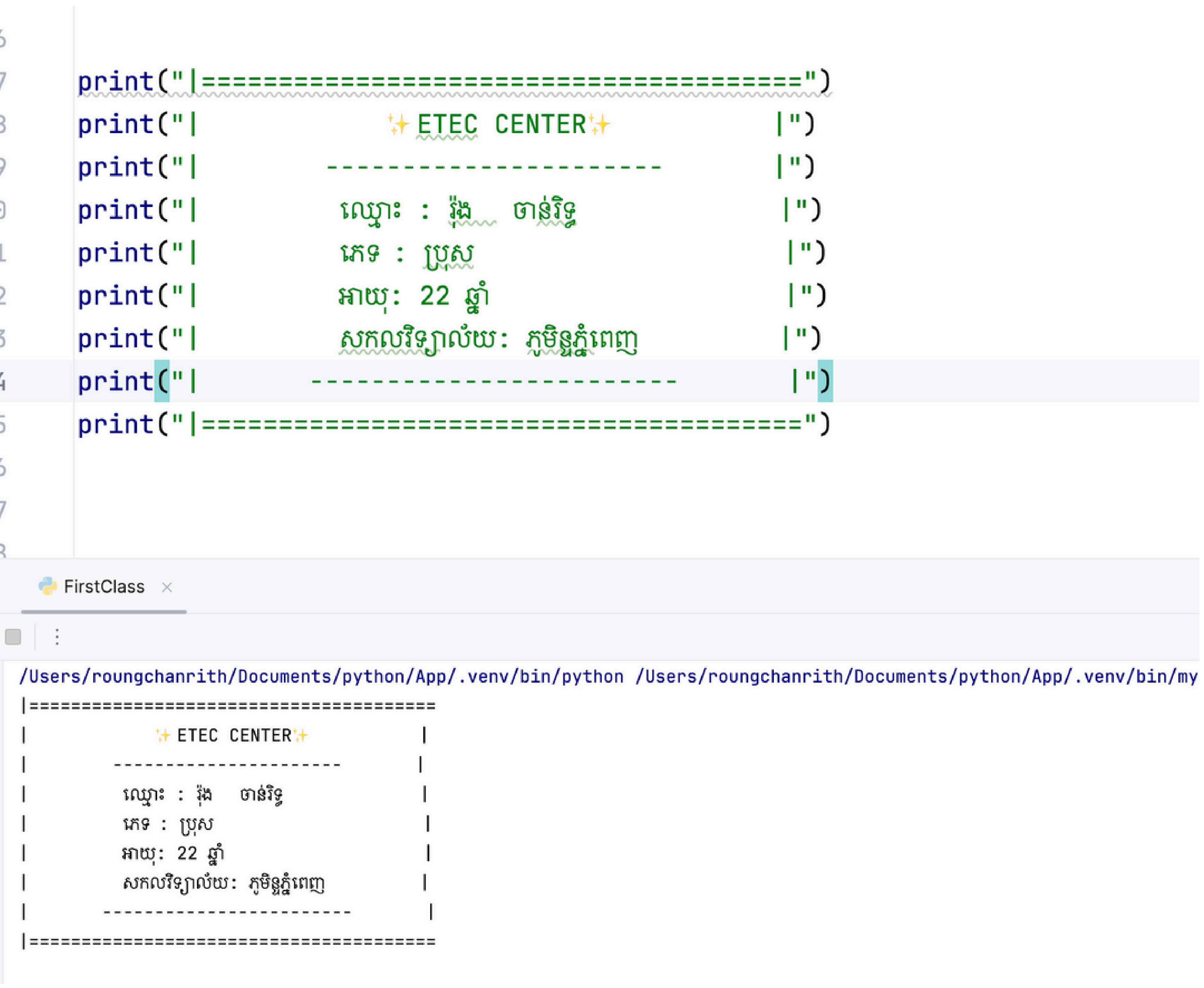
ធ្វើការ plug-in នូវ Extensions មួយចំនួនសម្រាប់
Run Python នៅក្នុង Vs Code ឬក៏ ដំណើរការ
ជាមួយនឹង local Environment setting up

PYTHON គ្មានទម្រង់គ្រឹះក្នុងការសរសេរកូដ



```python
print('Hello')
```

```
PS D:\work> python -u "d:\work\first.py"
Hello
PS D:\work>
```

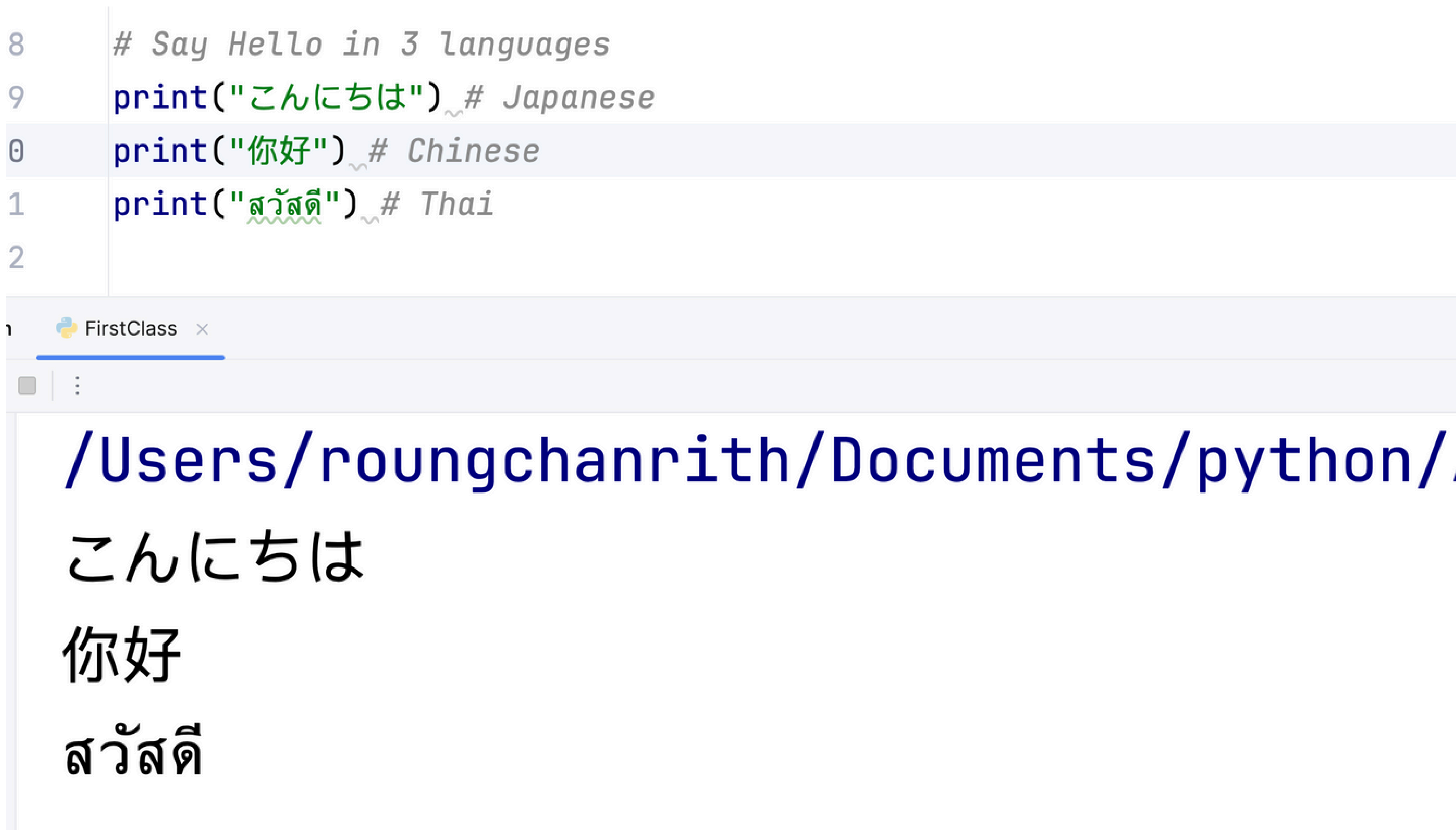


```python
print("|=====================================")
print("|             ✨ETEC CENTER✨            |")
print("|       -----------------------        |")
print("|         ឈ្មោះ : រ៉ុង  ចាន់រិទ្ធ            |")
print("|         ភេទ : ប្រុស                    |")
print("|         អាយុ: 22 ឆ្នាំ                  |")
print("|         សកលវិទ្យាល័យ: ភូមិន្ទភ្នំពេញ         |")
print("|      -------------------------        |")
print("|=====================================")
```

```
/Users/roungchanrith/Documents/python/App/.venv/bin/python /Users/roungchanrith/Documents/python/App/.venv/bin/my
|=====================================
|             ✨ETEC CENTER✨           |
|       -----------------------        |
|         ឈ្មោះ : រ៉ុង  ចាន់រិទ្ធ            |
|         ភេទ : ប្រុស                    |
|         អាយុ: 22 ឆ្នាំ                  |
|         សកលវិទ្យាល័យ: ភូមិន្ទភ្នំពេញ         |
|      -------------------------       |
|=====================================
```

EXAMPLE 02

ភាសា Python ជាភាសារដែល Support Multiple languages



```python
# Say Hello in 3 languages
print("こんにちは") # Japanese
print("你好") # Chinese
print("สวัสดี") # Thai
```

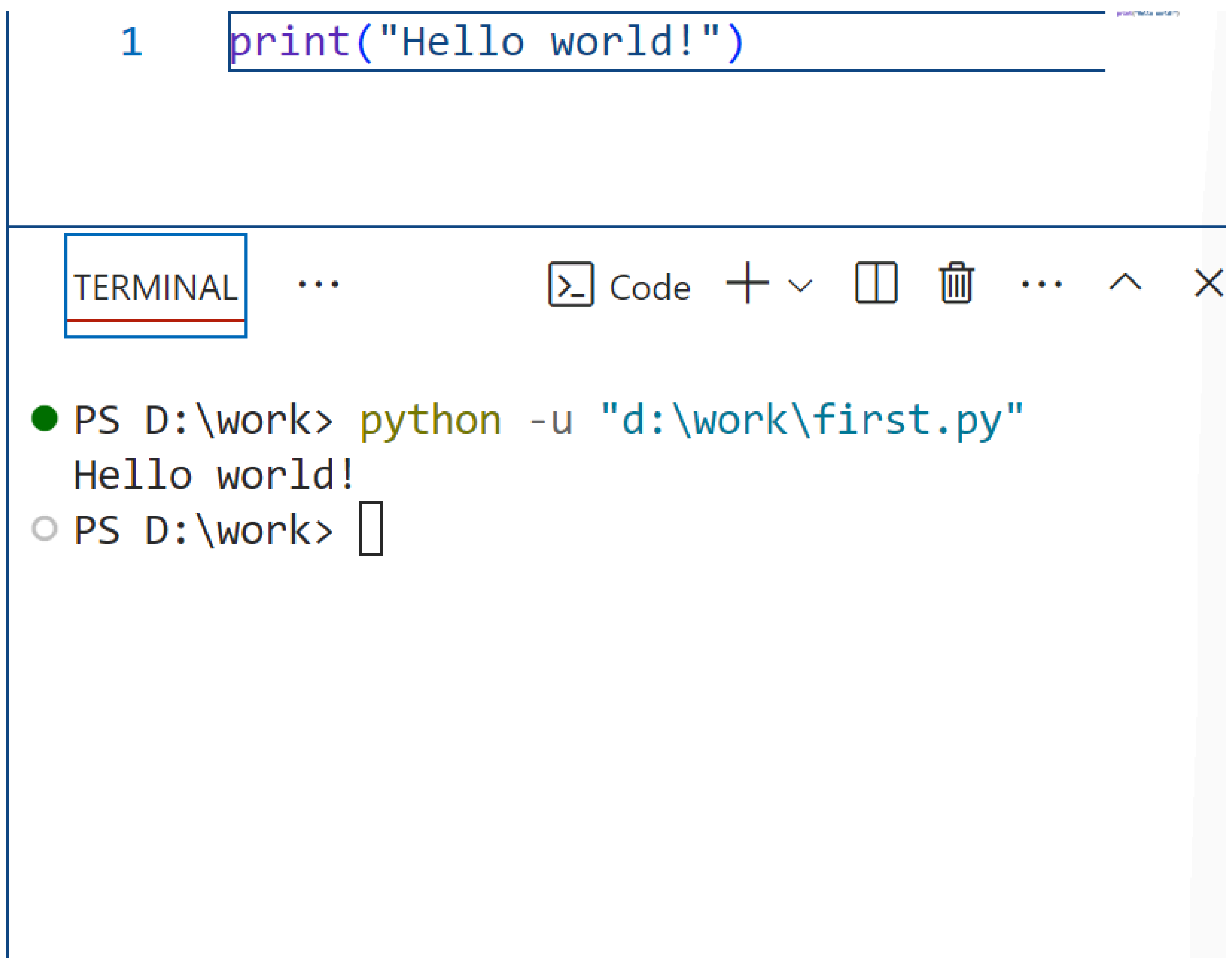
```
/Users/roungchanrith/Documents/python/
こんにちは
你好
สวัสดี
```

លំហាត់អនុវត្ត

ចូរបង្ហាញទិន្ន័យរបស់អ្នកដោយយលគំរូតាមការបង្ហាញខាងក្រោម

```
==========[ ETEC CENTER  ]=========
 ID: E008
 Name:  Roung Chanrith
 Gender: Male
Place of Birth: Battambang
 Phone number: 060 535 771

=================================
```

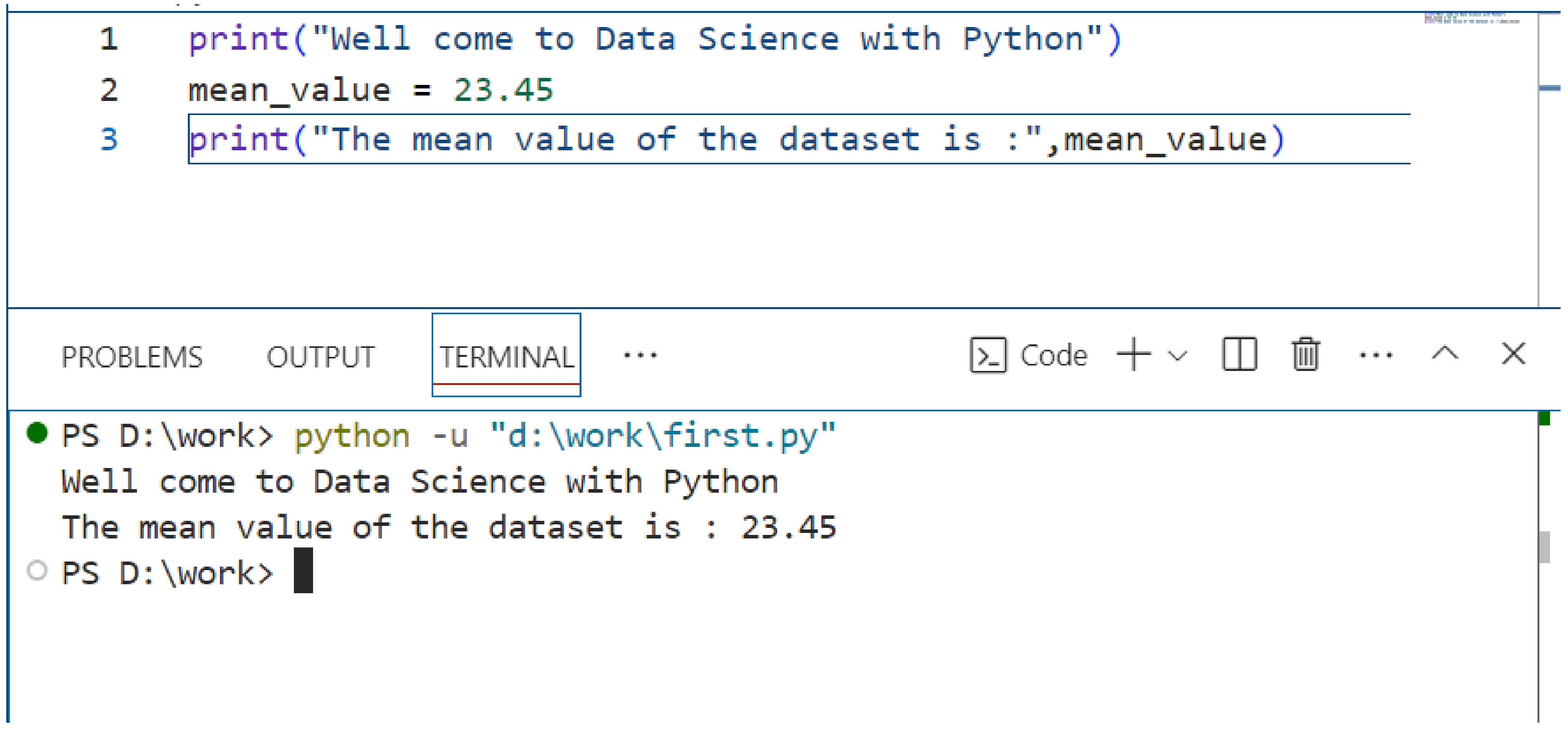
EXAMPLE 01



```
print("Hello world!")
```

TERMINAL ··· Code

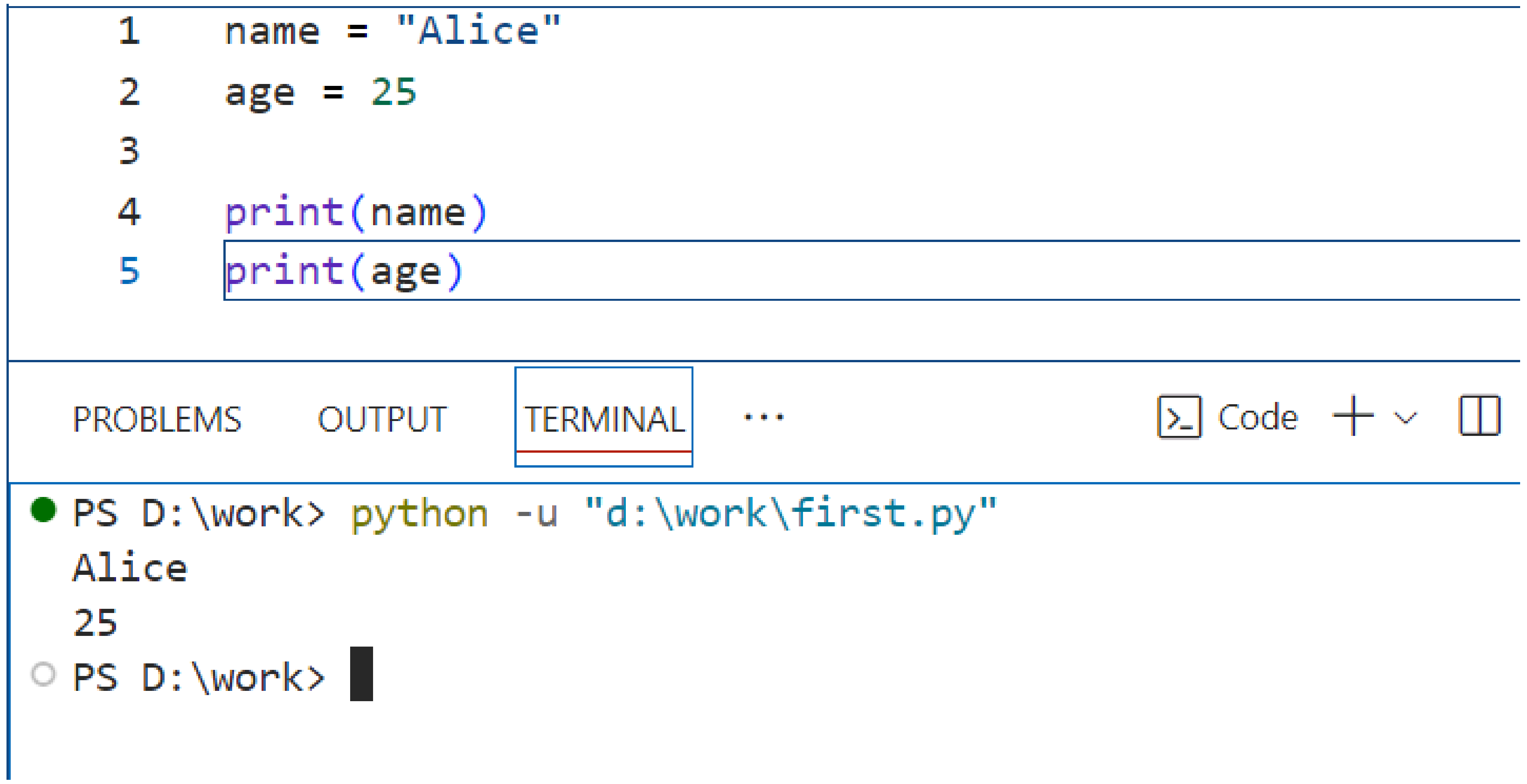
```
PS D:\work> python -u "d:\work\first.py"
Hello world!
PS D:\work>
```



```
print("Well come to Data Science with Python")
mean_value = 23.45
print("The mean value of the dataset is :",mean_value)
```

PROBLEMS OUTPUT TERMINAL ··· Code

```
PS D:\work> python -u "d:\work\first.py"
Well come to Data Science with Python
The mean value of the dataset is : 23.45
PS D:\work>
```

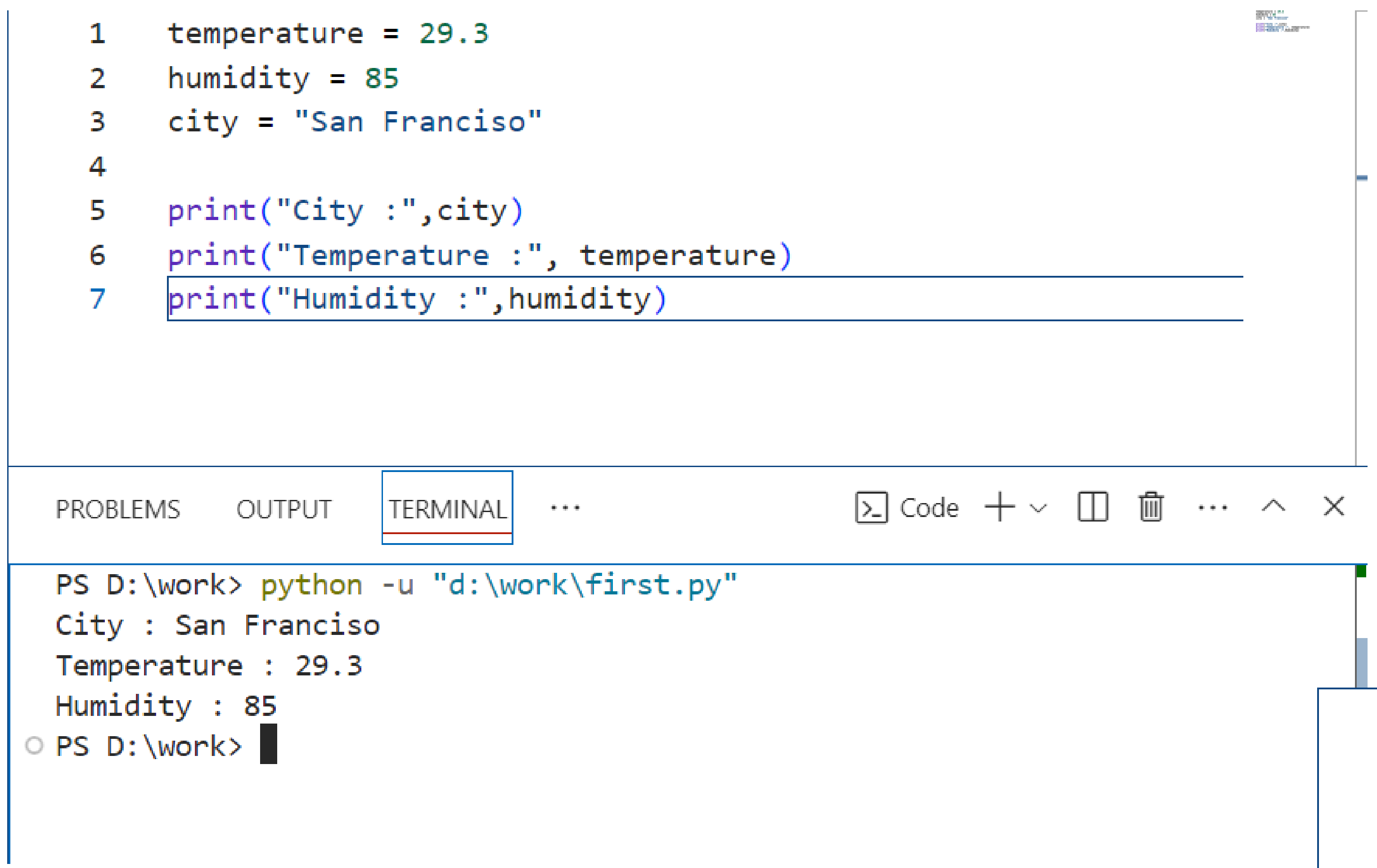


```
name = "Alice"
age = 25

print(name)
print(age)
```

PROBLEMS OUTPUT TERMINAL ··· Code

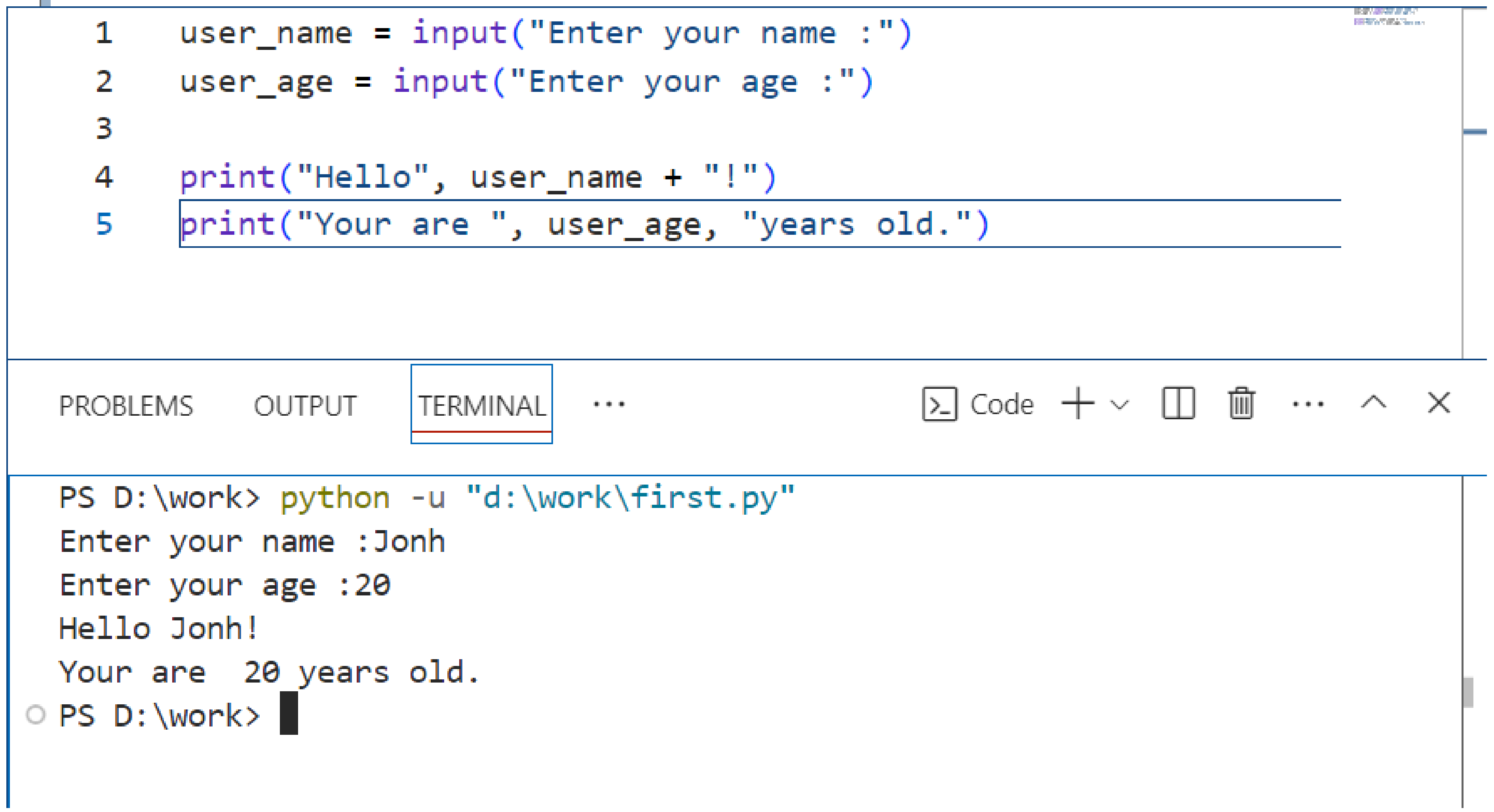
```
PS D:\work> python -u "d:\work\first.py"
Alice
25
PS D:\work>
```

EXAMPLE 03



```python
temperature = 29.3
humidity = 85
city = "San Franciso"

print("City :",city)
print("Temperature :", temperature)
print("Humidity :",humidity)
```

```
PS D:\work> python -u "d:\work\first.py"
City : San Franciso
Temperature : 29.3
Humidity : 85
PS D:\work>
```



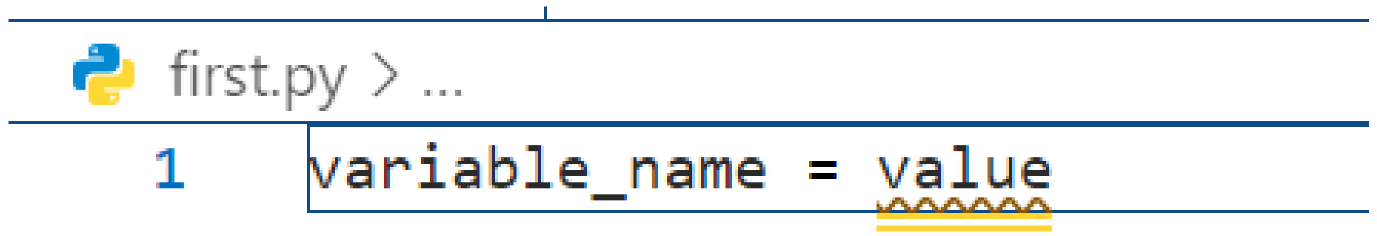
```python
user_name = input("Enter your name :")
user_age = input("Enter your age :")

print("Hello", user_name + "!")
print("Your are ", user_age, "years old.")
```

```
PS D:\work> python -u "d:\work\first.py"
Enter your name :Jonh
Enter your age :20
Hello Jonh!
Your are  20 years old.
PS D:\work>
```

VARIABLE :

Variable គឺជា ការតាង ឈ្មោះ ហើយផ្ទុកទិន្នន័យក្នុងឈ្មោះនោះ ។ Variable ក្នុង Python មានលក្ខណៈ ជា dynamically typed បានន័យថាយើងមិនចាំបាច់ប្រកាស Type របស់វាច្បាស់លាស់នោះទេ។ ជំនួសមកវិញ Type ត្រូវបានសន្និដ្ឋាន ឬសម្រេចចេញពីតម្លៃដែលយើងផ្តល់អោយទៅ variable ។

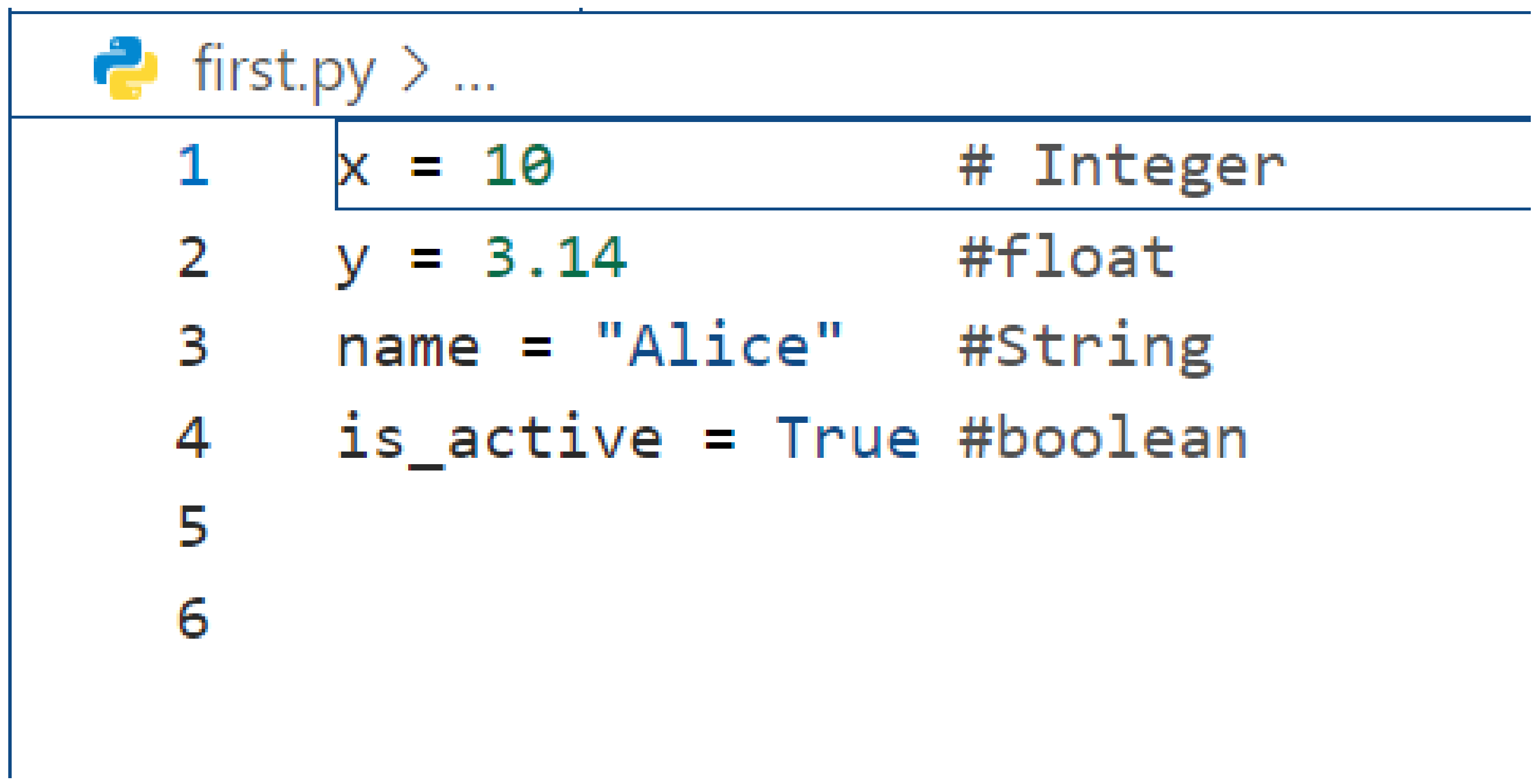
Syntax :



first.py > ...

```
variable_name = value
```

Example :



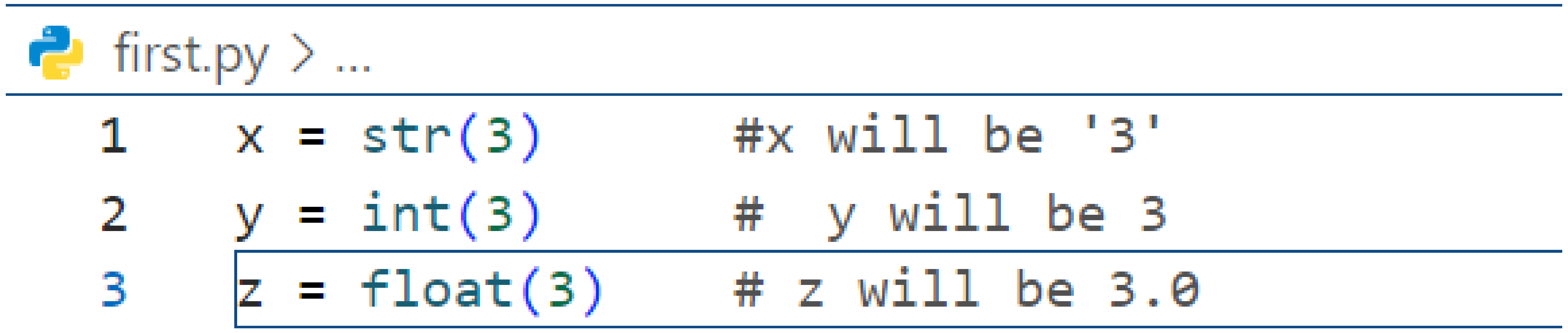
first.py > ...

```
x = 10                # Integer
y = 3.14              #float
name = "Alice"        #String
is_active = True #boolean
```

CASTING:

ប្រសិនបើអ្នកចង់បញ្ជាក់ប្រភេទទិន្នន័យនៃអថេរ នេះអាចត្រូវបានធ្វើដោយCasting។

Example :

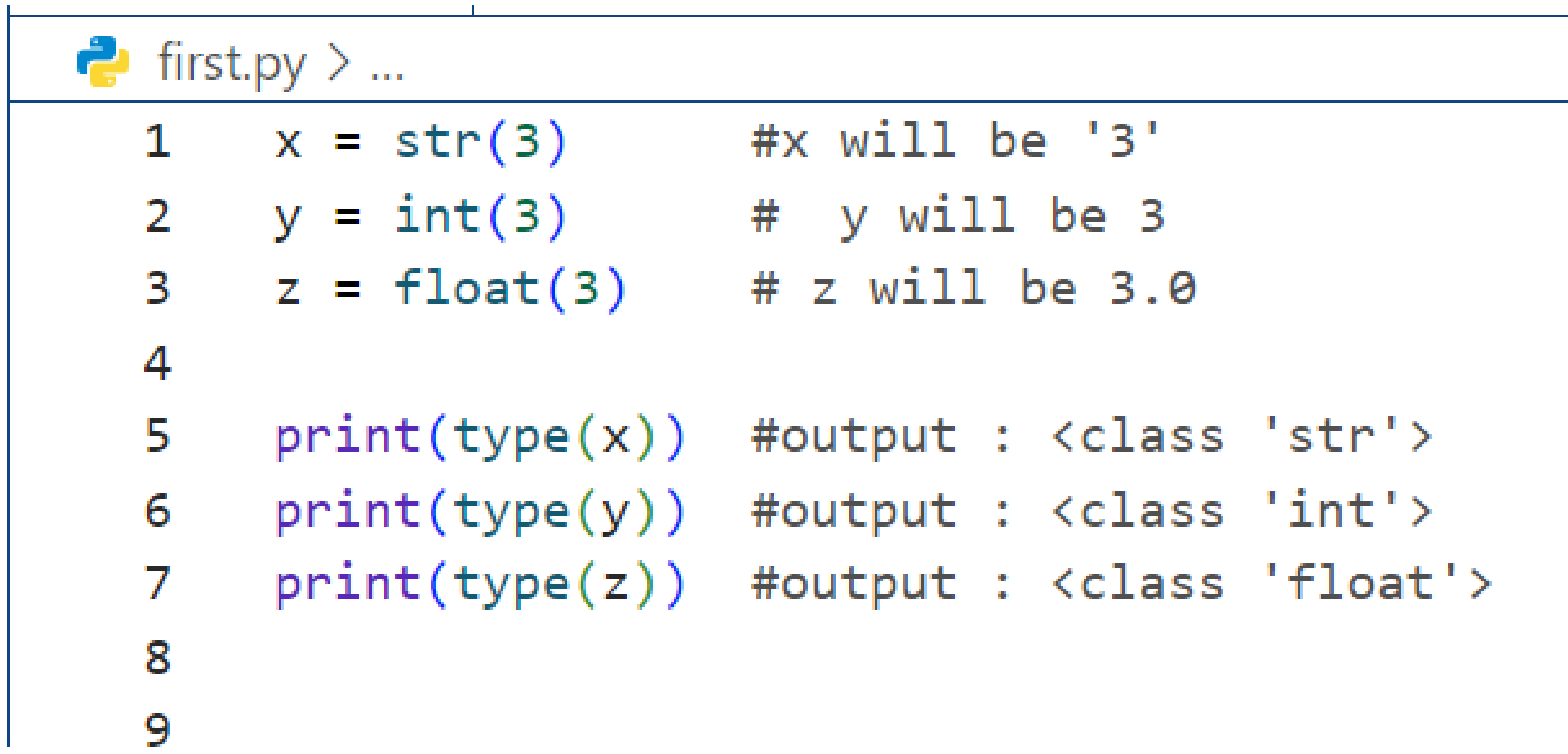


first.py > ...

```python
x = str(3)        #x will be '3'
y = int(3)        #  y will be 3
z = float(3)      # z will be 3.0
```

ប្រសិនបើអ្នកចង់ដឹងពី Data Type នៃ Variable គឺ យើងប្រើនូវ function{ type() } ។

Example :



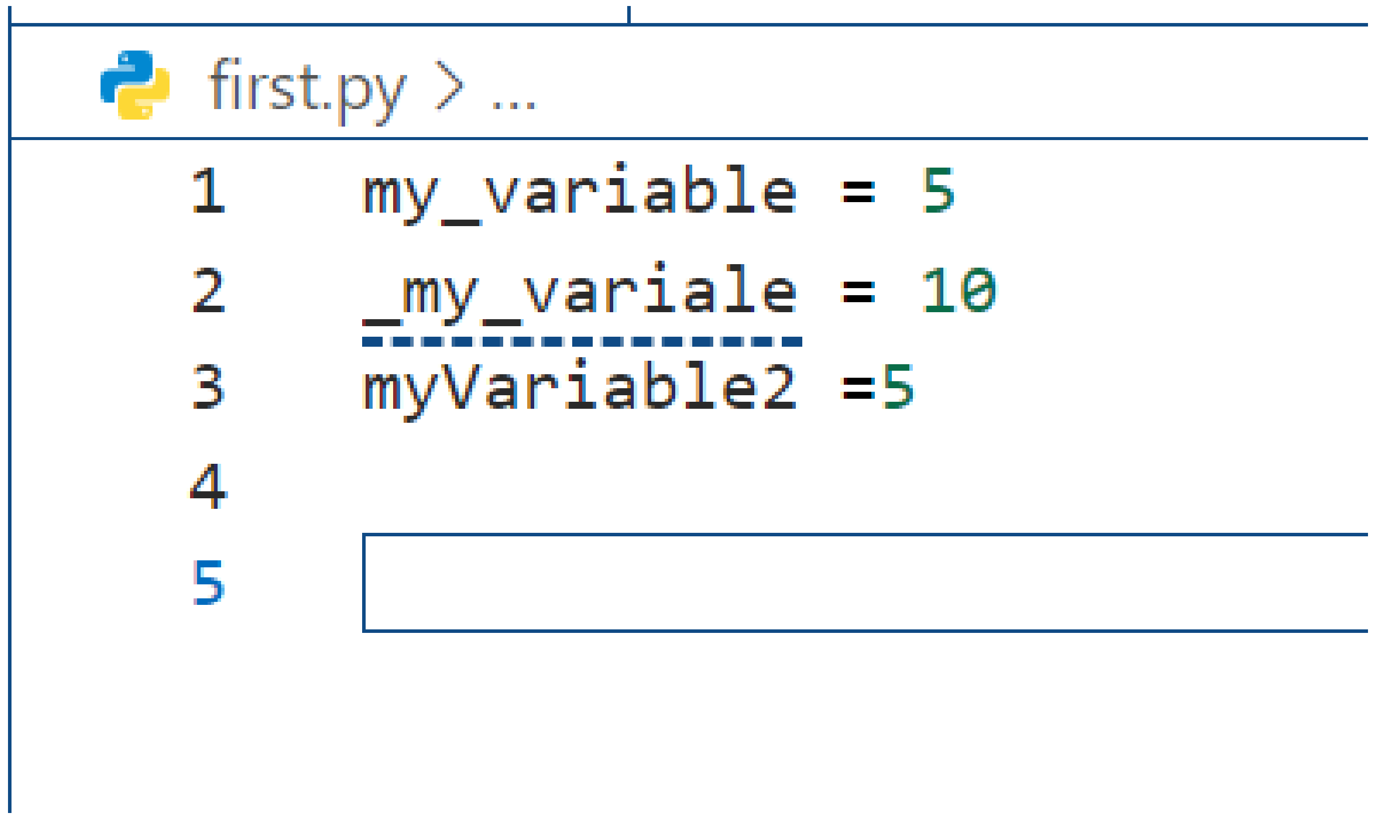
first.py > ...

```python
x = str(3)        #x will be '3'
y = int(3)        #  y will be 3
z = float(3)      # z will be 3.0

print(type(x))    #output : <class 'str'>
print(type(y))    #output : <class 'int'>
print(type(z))    #output : <class 'float'>


```

IDENTIFIER (ច្បាប់នៃការដាក់ឈ្មោះអោយ VARIABLE)

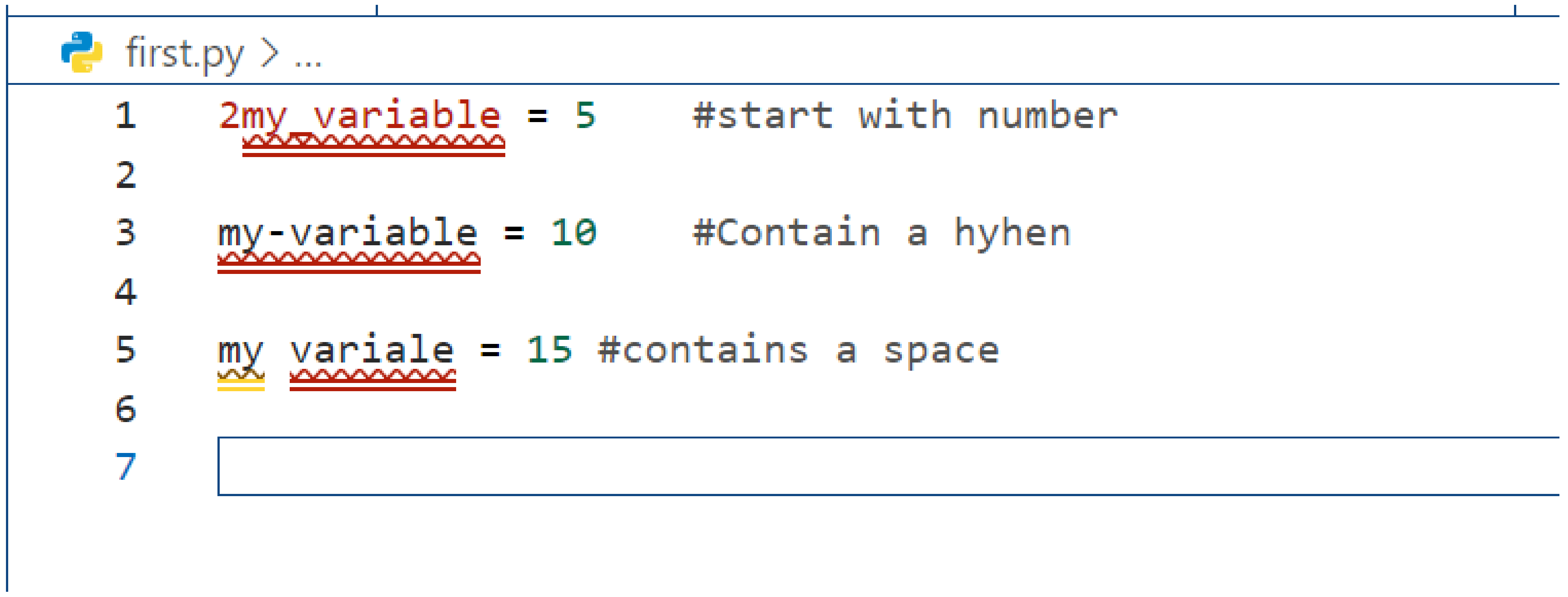
ការតាងឈ្មោះ Variable ត្រូវតែចាប់ផ្តើមដោយអក្សរ (a-z, A-Z) ឬសញ្ញាគូសក្រោម Underscore (_) ។ ឈ្មោះដែលនៅសល់ពីក្រោយអាចមានអក្សរ លេខ ឬសញ្ញាគូសក្រោម ។ ឈ្មោះ Variable គឺប្រកាន់អក្សរតូចធំ (“Age” និង “age” គឺជាអថេរផ្សេងគ្នា) ។

ការប្រកាស Variable ដែលត្រឹមត្រូវ ៖


first.py > ...

```python
my_variable = 5
_my_variale = 10
myVariable2 =5

```

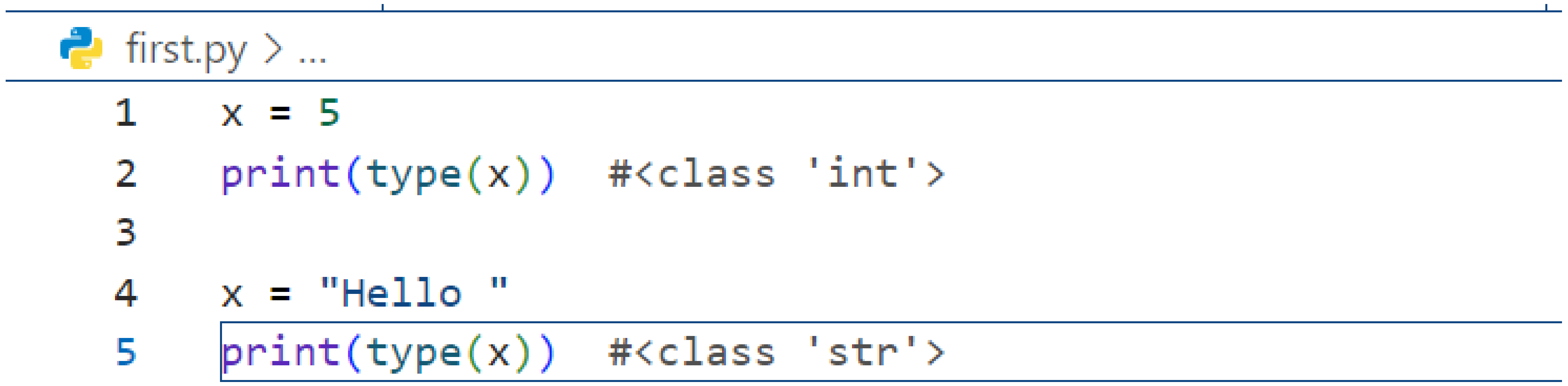
ការប្រកាស Variable ដែលមិនត្រឹមត្រូវ ៖


first.py > ...

```python
2my_variable = 5     #start with number

my-variable = 10     #Contain a hyhen

my variale = 15 #contains a space

```

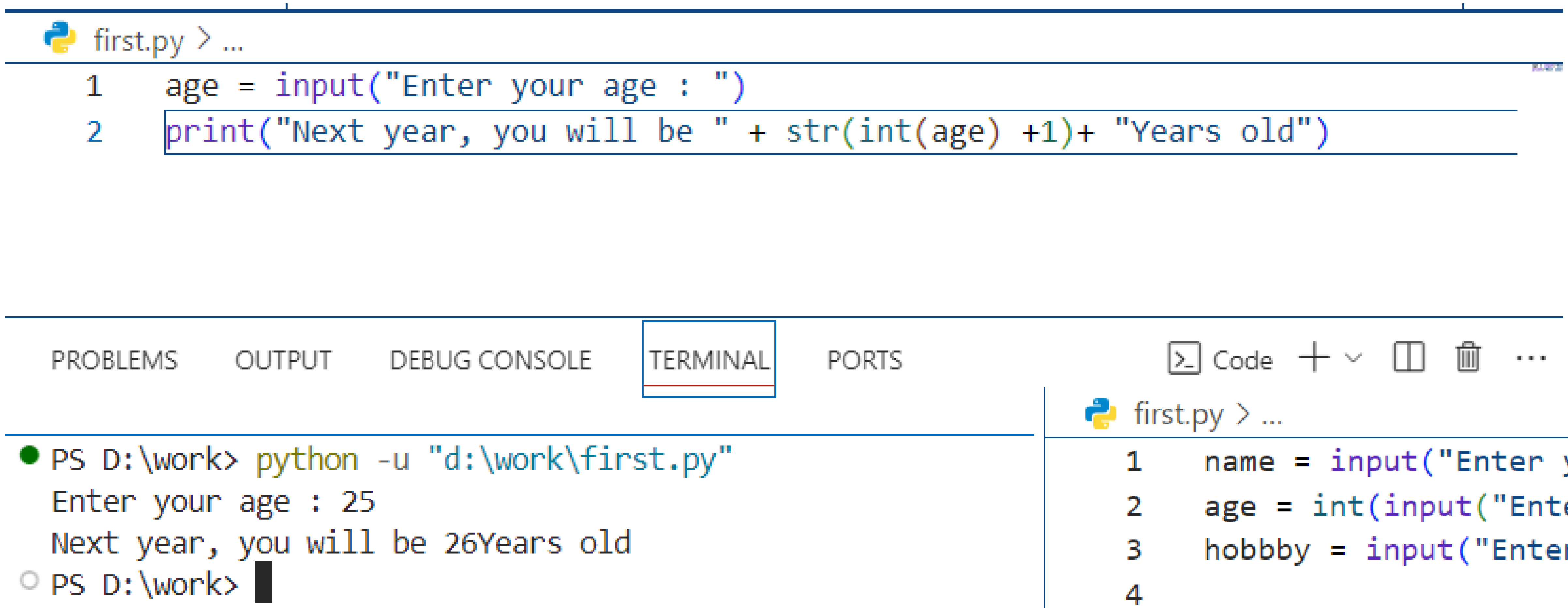
DYNAMIC TYPING :

Variable អាចផ្លាស់ប្តូរ Type បានបន្ទាប់ពីការកំណត់ឬអោយតម្លៃ ដោយសារ Dynamic Typing ៖



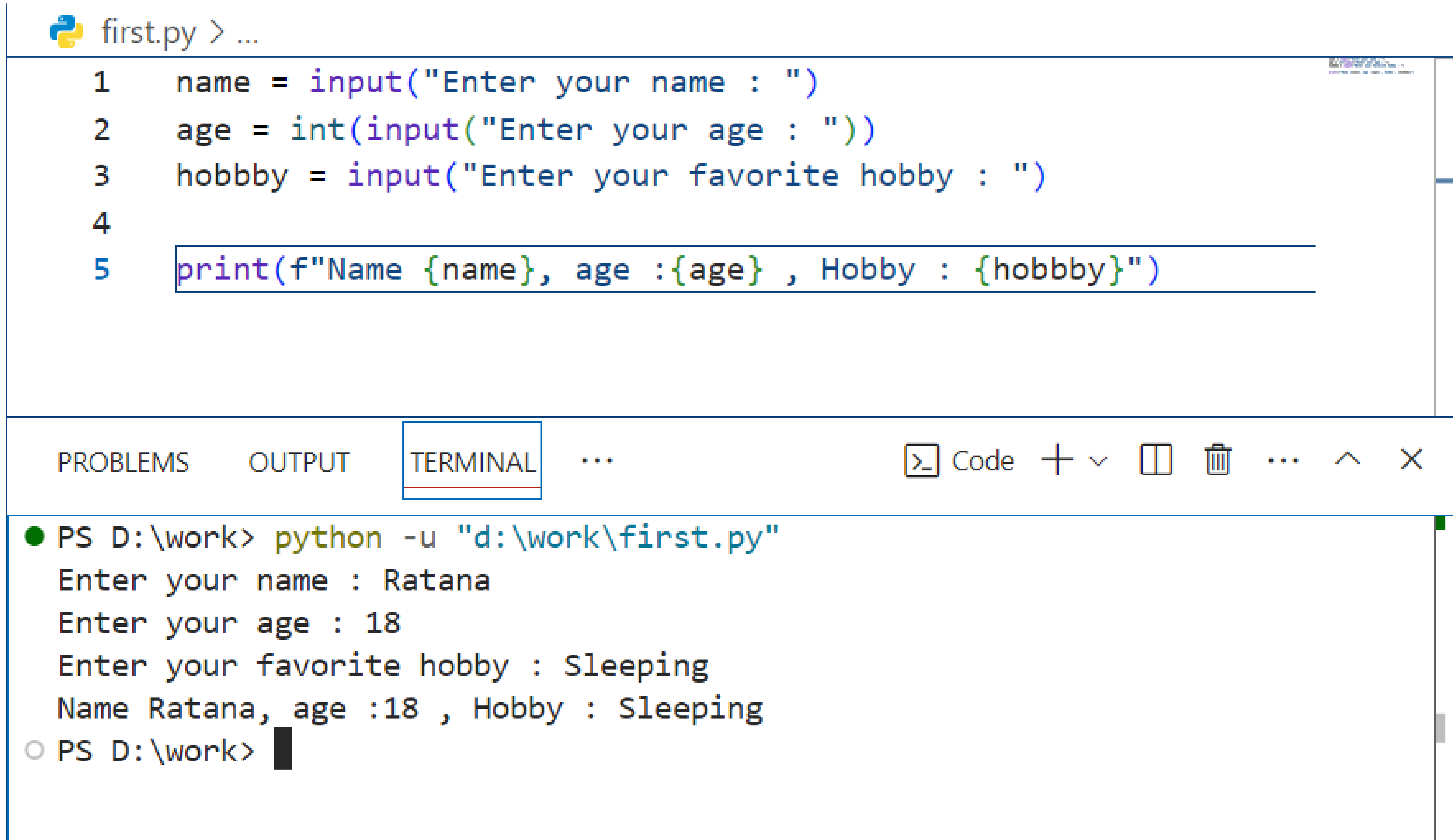
```python
x = 5
print(type(x))  #<class 'int'>

x = "Hello "
print(type(x))  #<class 'str'>
```

EXAMPLE 01



```python
age = input("Enter your age : ")
print("Next year, you will be " + str(int(age) +1)+ "Years old")
```

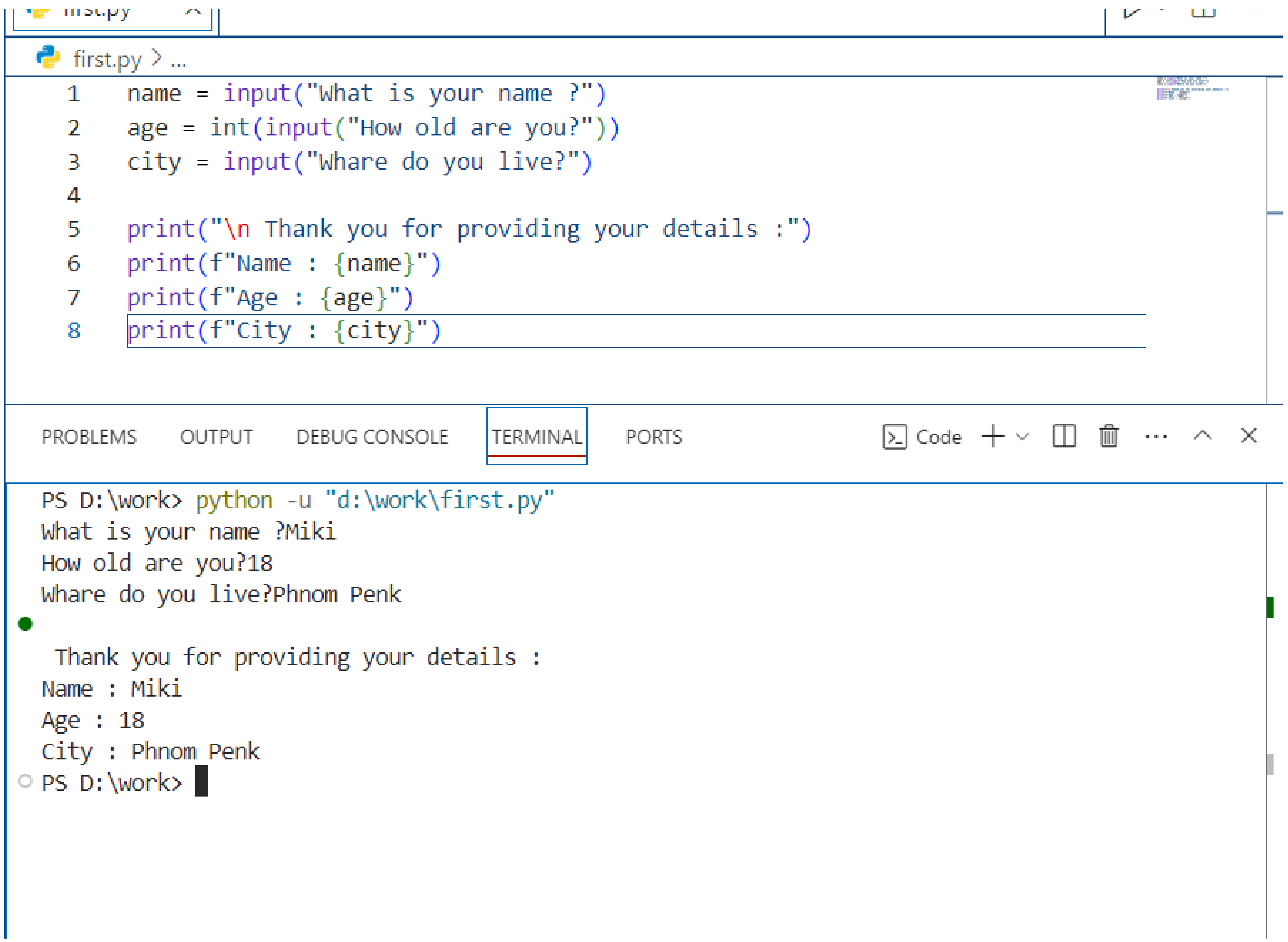
```
PS D:\work> python -u "d:\work\first.py"
Enter your age : 25
Next year, you will be 26Years old
PS D:\work>
```



```python
name = input("Enter your name : ")
age = int(input("Enter your age : "))
hobbby = input("Enter your favorite hobby : ")

print(f"Name {name}, age :{age} , Hobby : {hobbby}")
```

```
PS D:\work> python -u "d:\work\first.py"
Enter your name : Ratana
Enter your age : 18
Enter your favorite hobby : Sleeping
Name Ratana, age :18 , Hobby : Sleeping
PS D:\work>
```

EXAMPLE 02



```python
name = input("What is your name ?")
age = int(input("How old are you?"))
city = input("Whare do you live?")

print("\n Thank you for providing your details :")
print(f"Name : {name}")
print(f"Age : {age}")
print(f"City : {city}")
```

```
PS D:\work> python -u "d:\work\first.py"
What is your name ?Miki
How old are you?18
Whare do you live?Phnom Penk

 Thank you for providing your details :
Name : Miki
Age : 18
City : Phnom Penk
PS D:\work>
```

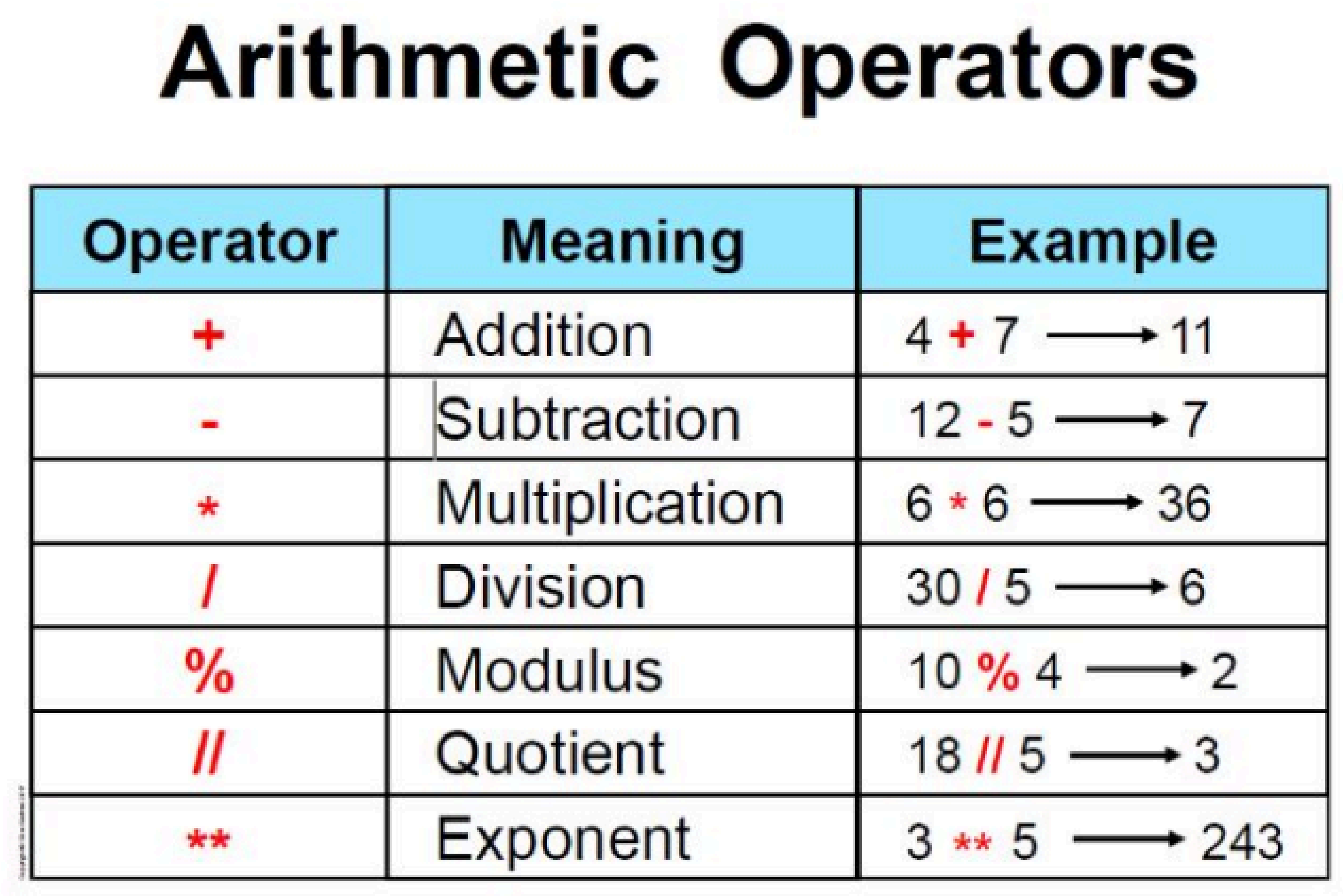
OPERATOR :

Operator ប្រើប្រាស់ដើម្បីធ្វើប្រមាណវិធីផ្សេងៗ ទៅលើ តម្លៃ ឬ Variable ។

Operatorត្រូវបានបែងចែកជាក្រុមដូចខាងក្រោម ៖

- Arithmetic operators/ សញ្ញាគណនា
- Assignment operators/ សញ្ញាកំណើននិង តំហយ
- Comparison operators/ សញ្ញាប្រៀបធៀប
- Logical operators/ សញ្ញាតក្ក

Arithmetic Operators ត្រូវបានប្រើជាមួយតម្លៃជាលេខដើម្បីធ្វើប្រមាណវិធីគណិតវិទ្យាទូទៅ ។



Arithmetic Operators

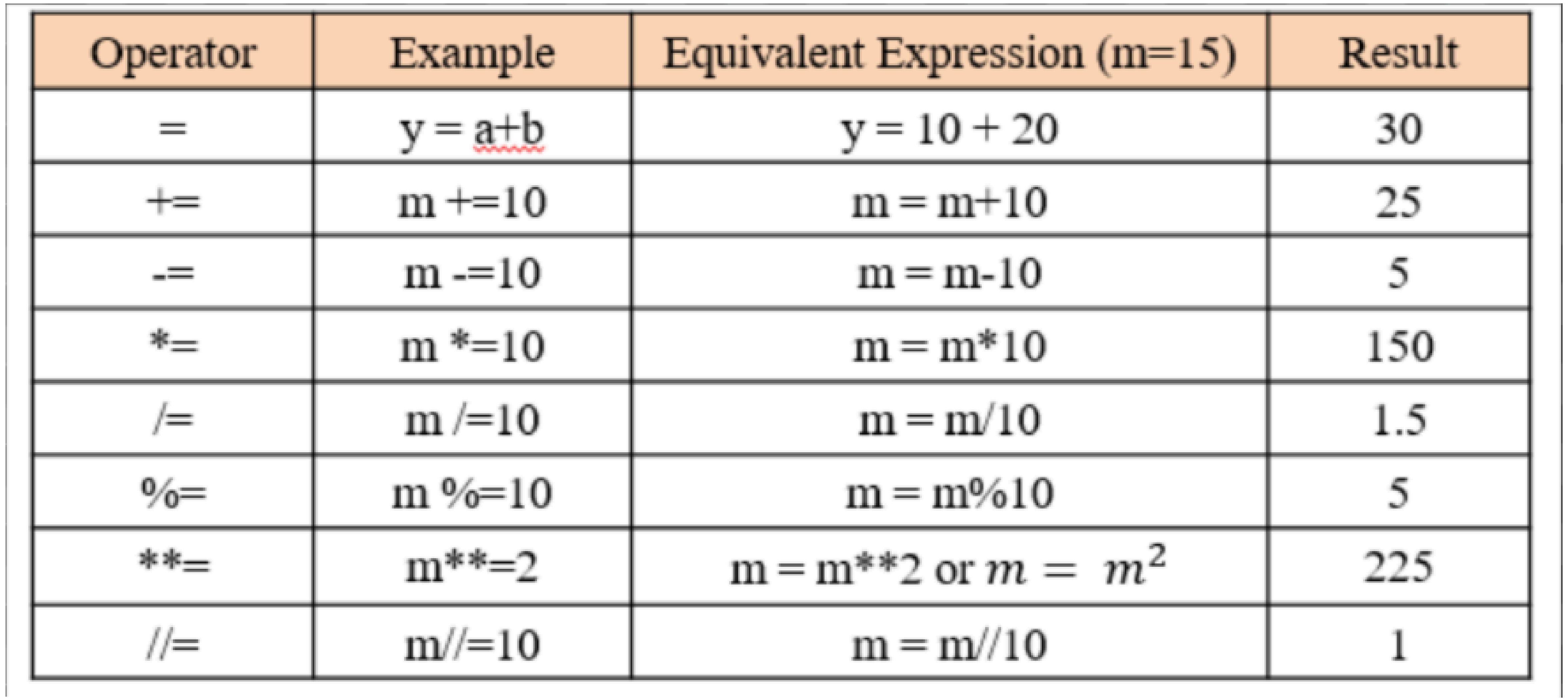
Operator	Meaning	Example
+	Addition	$4 + 7 \longrightarrow 11$
-	Subtraction	$12 - 5 \longrightarrow 7$
*	Multiplication	$6 * 6 \longrightarrow 36$
/	Division	$30 / 5 \longrightarrow 6$
%	Modulus	$10 \% 4 \longrightarrow 2$
//	Quotient	$18 // 5 \longrightarrow 3$
**	Exponent	$3 ** 5 \longrightarrow 243$

```
result = 2 + 3    # result is 5
```

```
result = 5 - 2    # result is 3
```

```
result = 2 * 3    # result is 6
result = 6 / 3    # result is 2
result = 7 // 3    # result is 2
```

Assignment Operators ត្រូវបានប្រើដើម្បីកំណត់តម្លៃទៅអថេរ ។

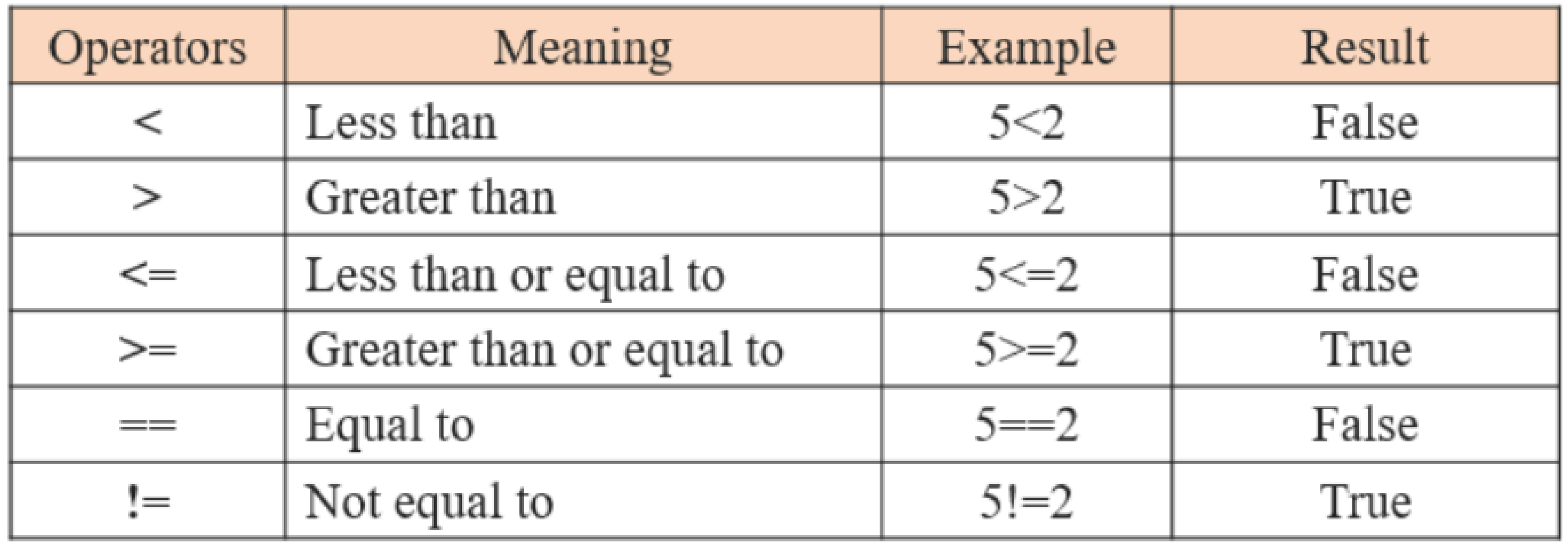


Operator	Example	Equivalent Expression (m=15)	Result
=	y = a+b	y = 10 + 20	30
+=	m +=10	m = m+10	25
-=	m -=10	m = m-10	5
*=	m *=10	m = m*10	150
/=	m /=10	m = m/10	1.5
%=	m %=10	m = m%10	5
**=	m**=2	m = m**2 or $m = m^2$	225
//=	m//=10	m = m//10	1

```
x  =5
x  +=  3   # x  =   x  +  3
x  -=3     # x  = x  -3
```

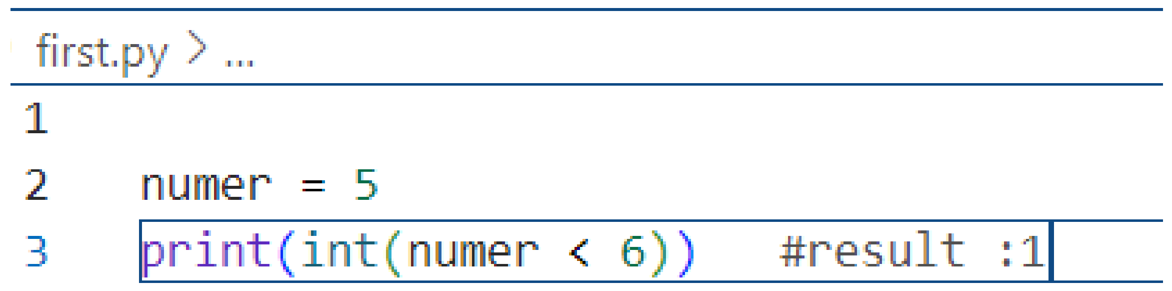
Comparison operators ត្រូវបានប្រើដើម្បីប្រៀបធៀបតម្លៃពីរ ។

ចំណាំ : វានឹងផ្តល់តម្លៃត្រឡប់មកវិញតែ ពីរប៉ុណ្ណោះគឺ True (ពិត) និង False (មិនពិត)។



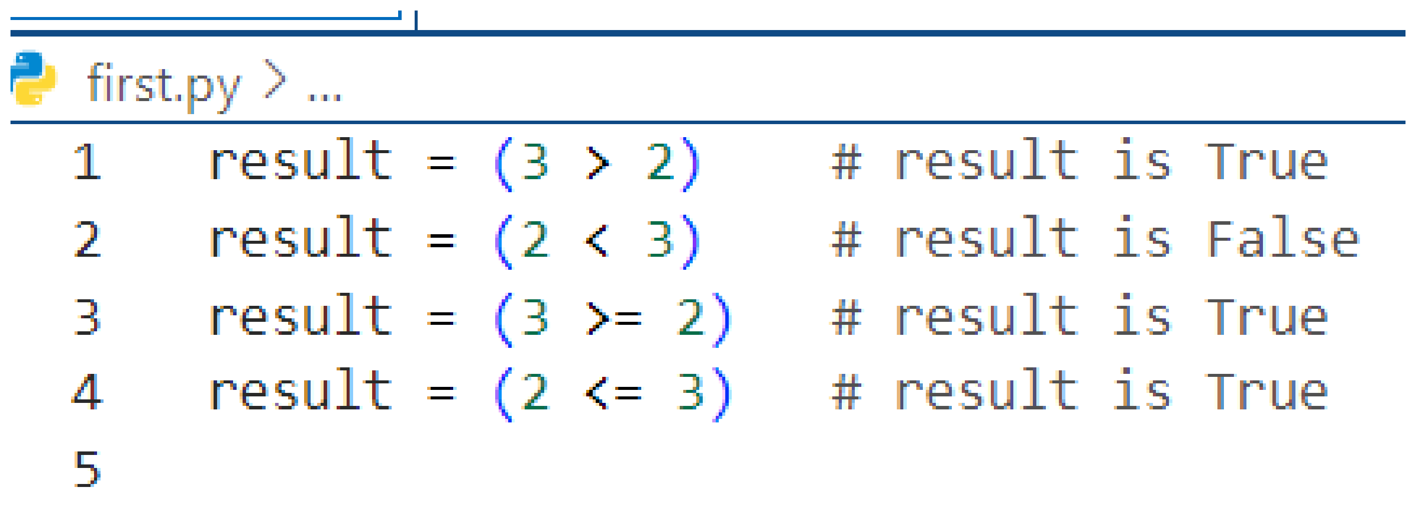
Operators	Meaning	Example	Result
<	Less than	5<2	False
>	Greater than	5>2	True
<=	Less than or equal to	5<=2	False
>=	Greater than or equal to	5>=2	True
==	Equal to	5==2	False
!=	Not equal to	5!=2	True

បើសិនជាយើងចង់បានតម្លៃជាលេខយើង ត្រូវ បញ្ជាក់វាជាមួយប្រភេទទិន្នន័យ int ។



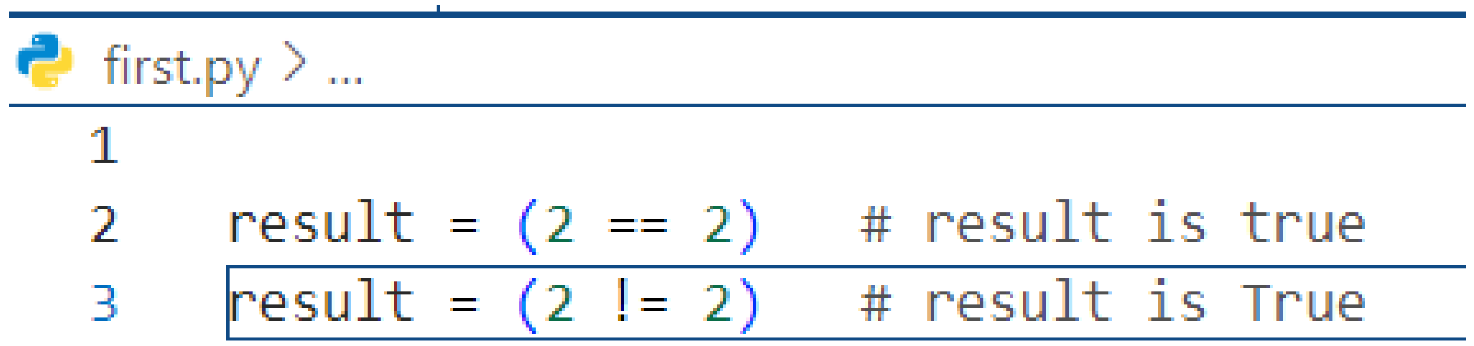
first.py > ...

```
numer = 5
print(int(numer < 6))    #result :1
```



first.py > ...

```
result = (3 > 2)     # result is True
result = (2 < 3)     # result is False
result = (3 >= 2)    # result is True
result = (2 <= 3)    # result is True
```



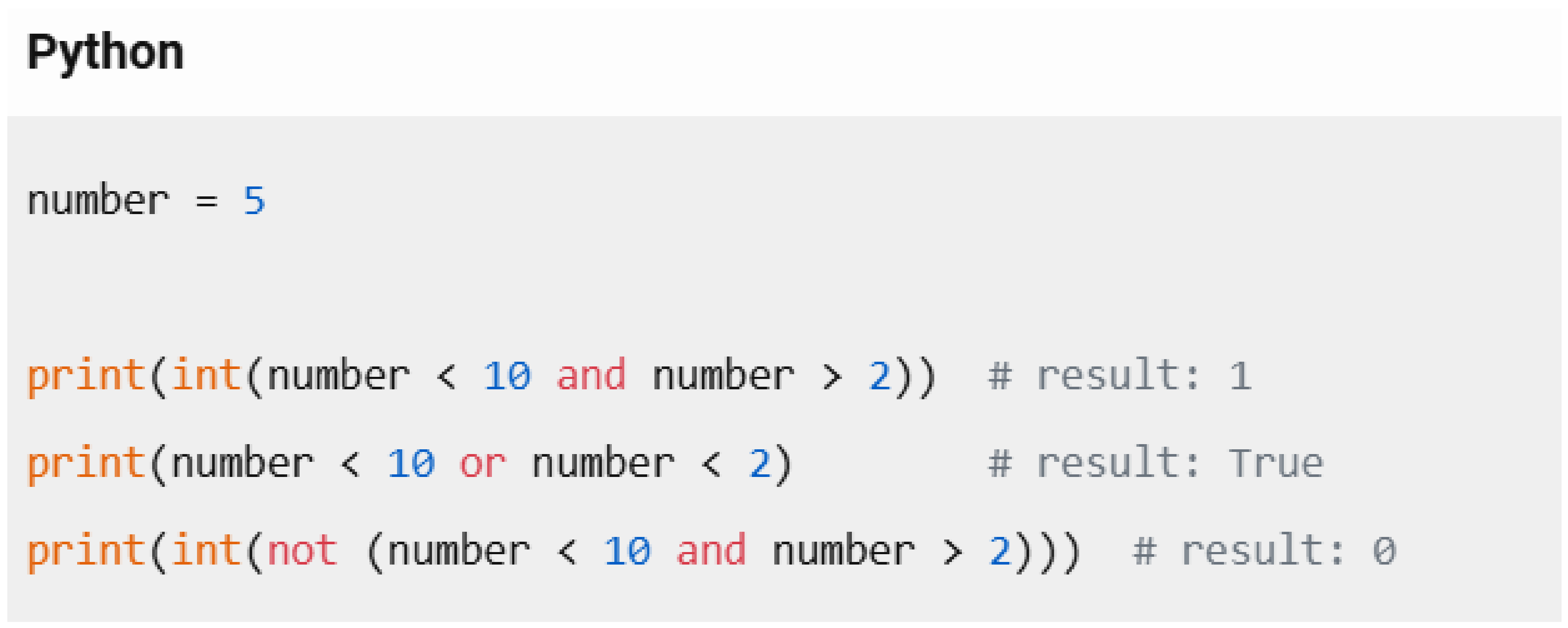
first.py > ...

```
result = (2 == 2)    # result is true
result = (2 != 2)    # result is True
```

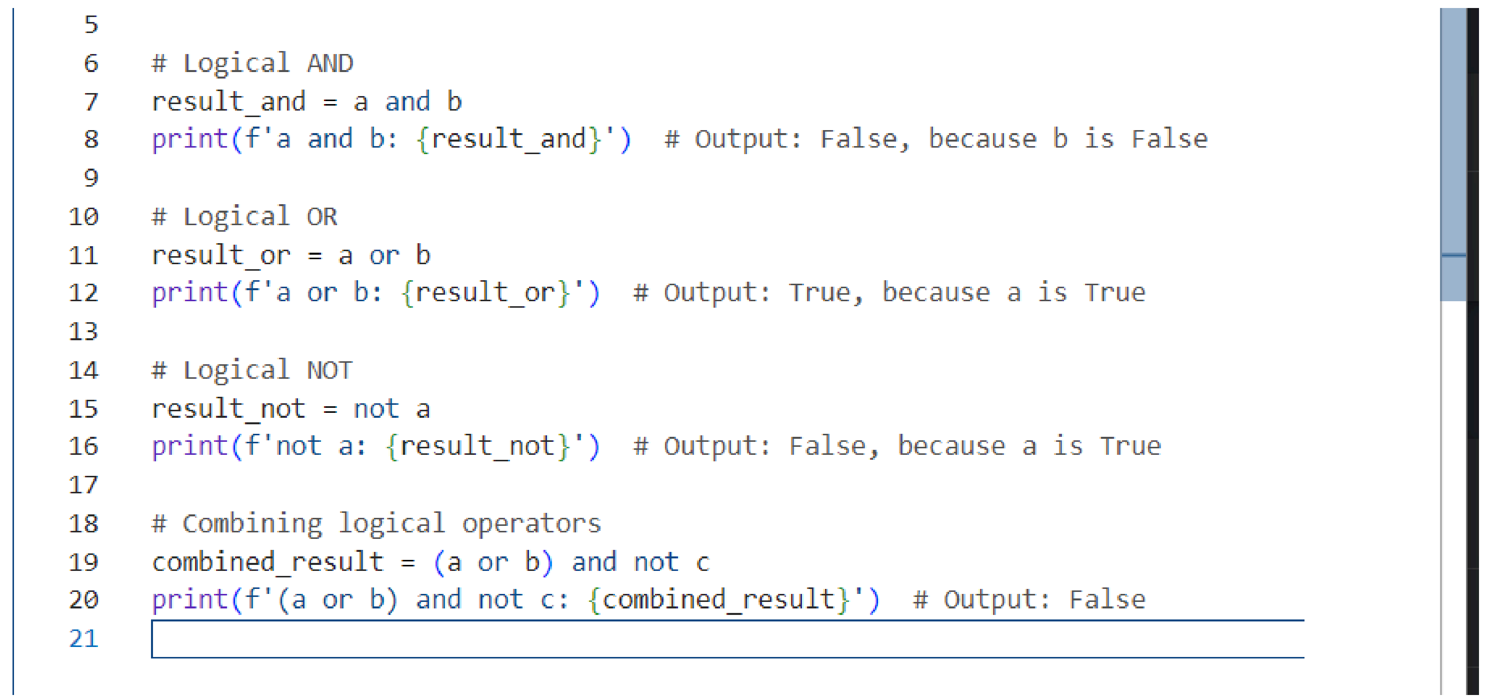
Logical Operators ប្រើប្រាស់ដើម្បី combine conditional statements

ចំណាំ : វានឹងផ្តល់តម្លៃត្រឡប់មកវិញតែ ពីរប៉ុណ្ណោះគឺ
True (ពិត) និង False (មិនពិត)។

បើសិនជាយើងចង់បានតម្លៃជាលេខយើងត្រូវ បញ្ជាក់វាជាមួយ
ប្រភេទទិន្នន័យ int ។


Python

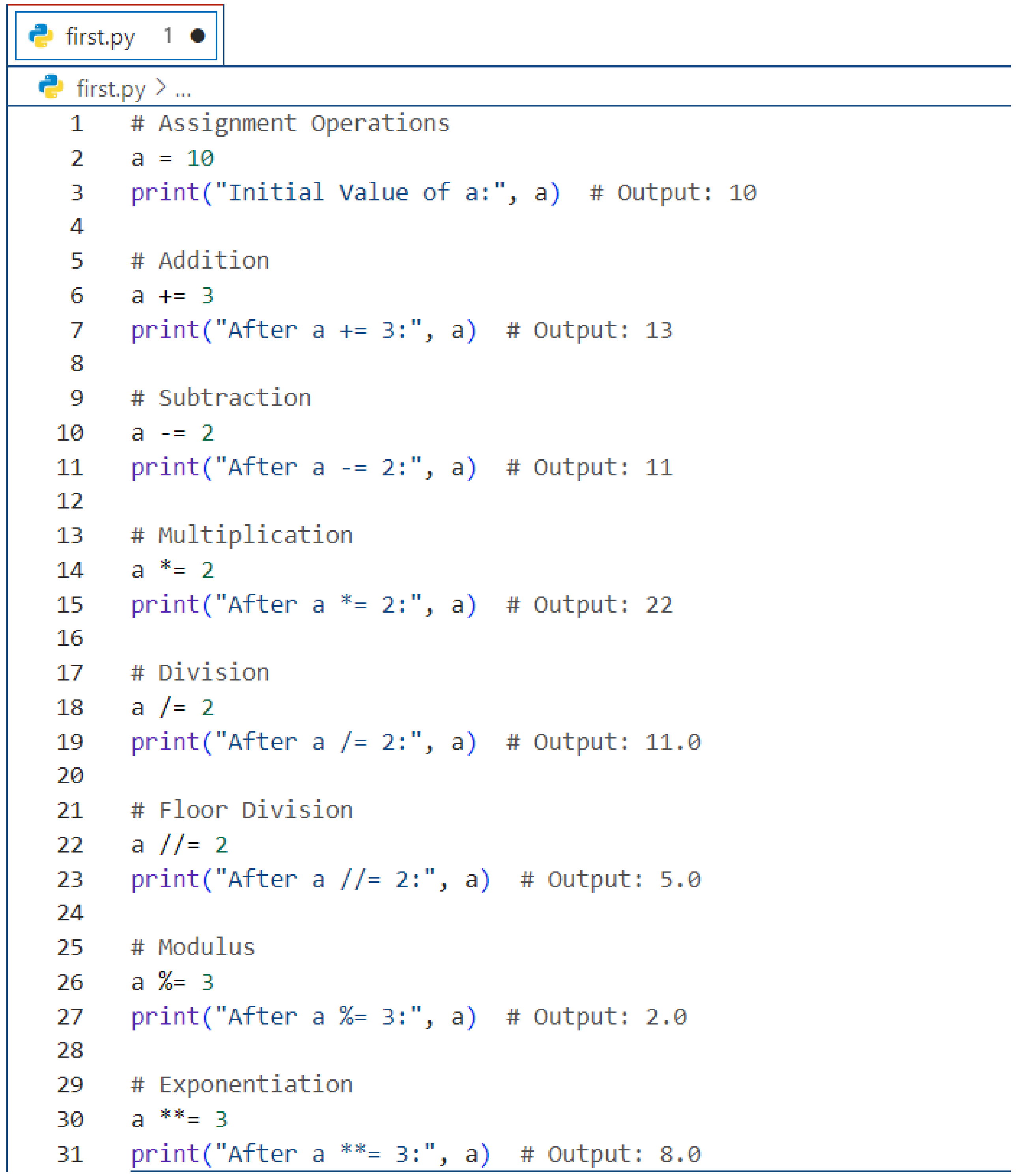
```python
number = 5


print(int(number < 10 and number > 2))  # result: 1
print(number < 10 or number < 2)         # result: True
print(int(not (number < 10 and number > 2)))  # result: 0
```



```python

# Logical AND
result_and = a and b
print(f'a and b: {result_and}')  # Output: False, because b is False

# Logical OR
result_or = a or b
print(f'a or b: {result_or}')  # Output: True, because a is True

# Logical NOT
result_not = not a
print(f'not a: {result_not}')  # Output: False, because a is True

# Combining logical operators
combined_result = (a or b) and not c
print(f'(a or b) and not c: {combined_result}')  # Output: False

```

EXAMPLE 01

```python
a = 10
b = 3

addition = a + b
subtraction = a - b
multiplication = a * b
division = a / b
floor_division = a // b
modulus = a % b
exponentiation = a ** b

print("Addition:", addition)        # Output: 13
print("Subtraction:", subtraction)  # Output: 7
print("Multiplication:", multiplication)    # Output: 30
print("Division:", division)        # Output: 3.3333333333333335
print("Floor Division:", floor_division)    # Output: 3
print("Modulus:", modulus)          # Output: 1
print("Exponentiation:", exponentiation)    # Output: 1000
```



first.py 1 ●

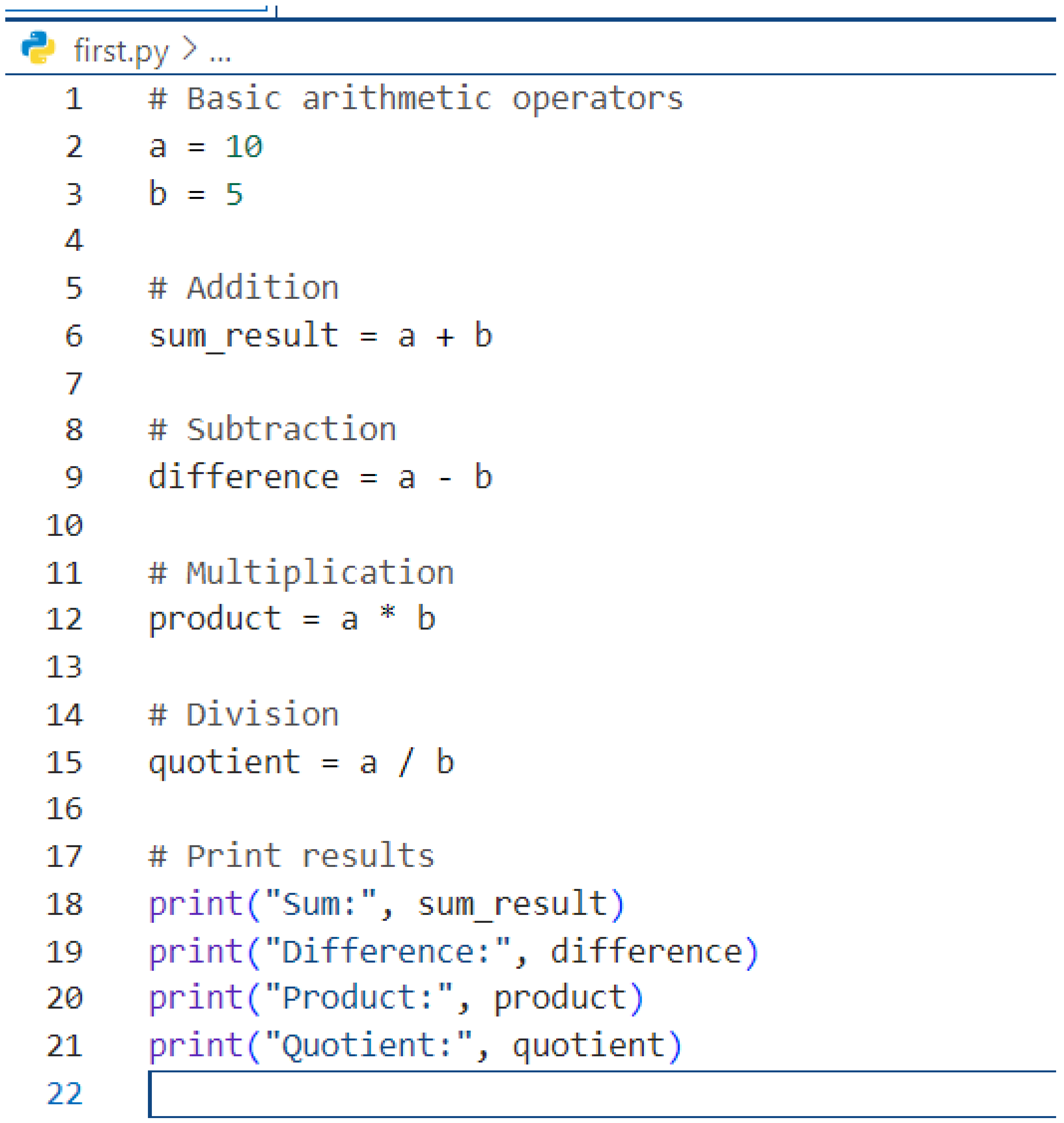
first.py > ...

```python
# Assignment Operations
a = 10
print("Initial Value of a:", a)  # Output: 10

# Addition
a += 3
print("After a += 3:", a)  # Output: 13

# Subtraction
a -= 2
print("After a -= 2:", a)  # Output: 11

# Multiplication
a *= 2
print("After a *= 2:", a)  # Output: 22

# Division
a /= 2
print("After a /= 2:", a)  # Output: 11.0

# Floor Division
a //= 2
print("After a //= 2:", a)  # Output: 5.0

# Modulus
a %= 3
print("After a %= 3:", a)  # Output: 2.0

# Exponentiation
a **= 3
print("After a **= 3:", a)  # Output: 8.0
```

EXAMPLE 02

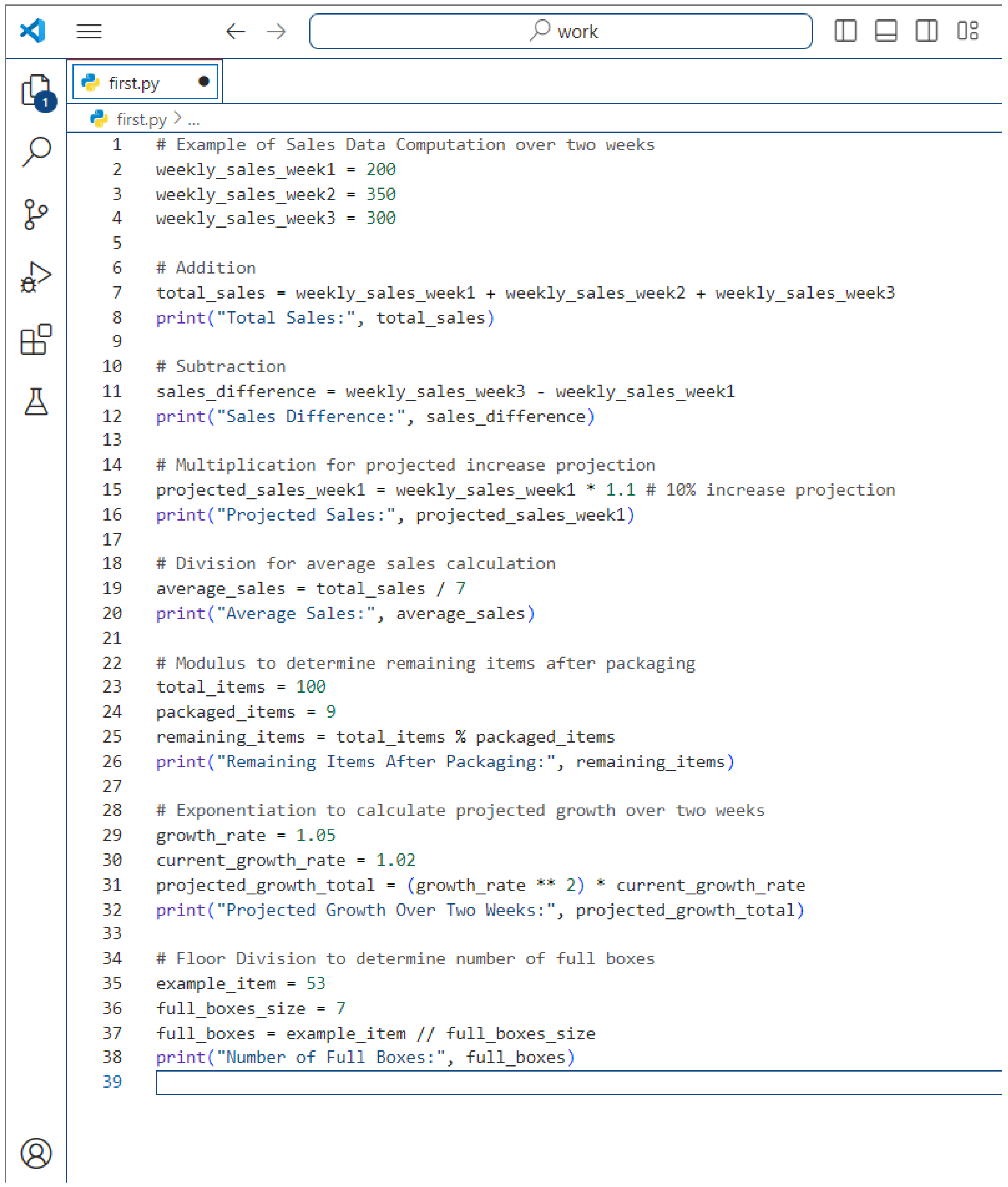
```python
# Logical Operations
a = True
b = False


print("a and b:", a and b)  # Output: False
print("a or b:", a or b)    # Output: True
```

```python
# Comparison Operations
a = 10
b = 5


print("a == b:", a == b)  # Output: False
print("a != b:", a != b)  # Output: True
print("a > b:", a > b)    # Output: True
print("a < b:", a < b)    # Output: False
print("a >= b:", a >= b)  # Output: True
print("a <= b:", a <= b)  # Output: False
```

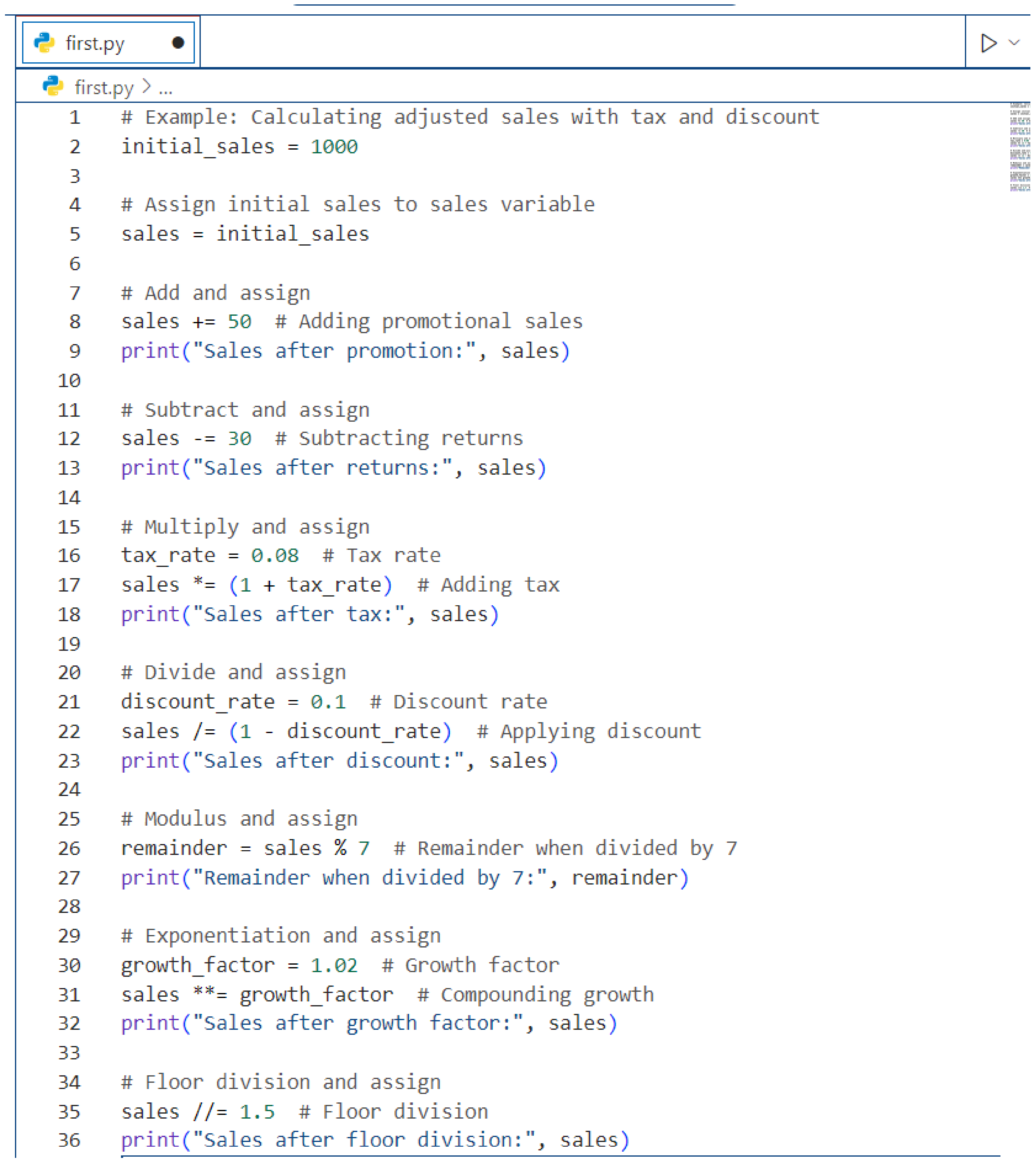

first.py > ...

```python
# Basic arithmetic operators
a = 10
b = 5

# Addition
sum_result = a + b

# Subtraction
difference = a - b

# Multiplication
product = a * b

# Division
quotient = a / b

# Print results
print("Sum:", sum_result)
print("Difference:", difference)
print("Product:", product)
print("Quotient:", quotient)

```

EXAMPLE 03

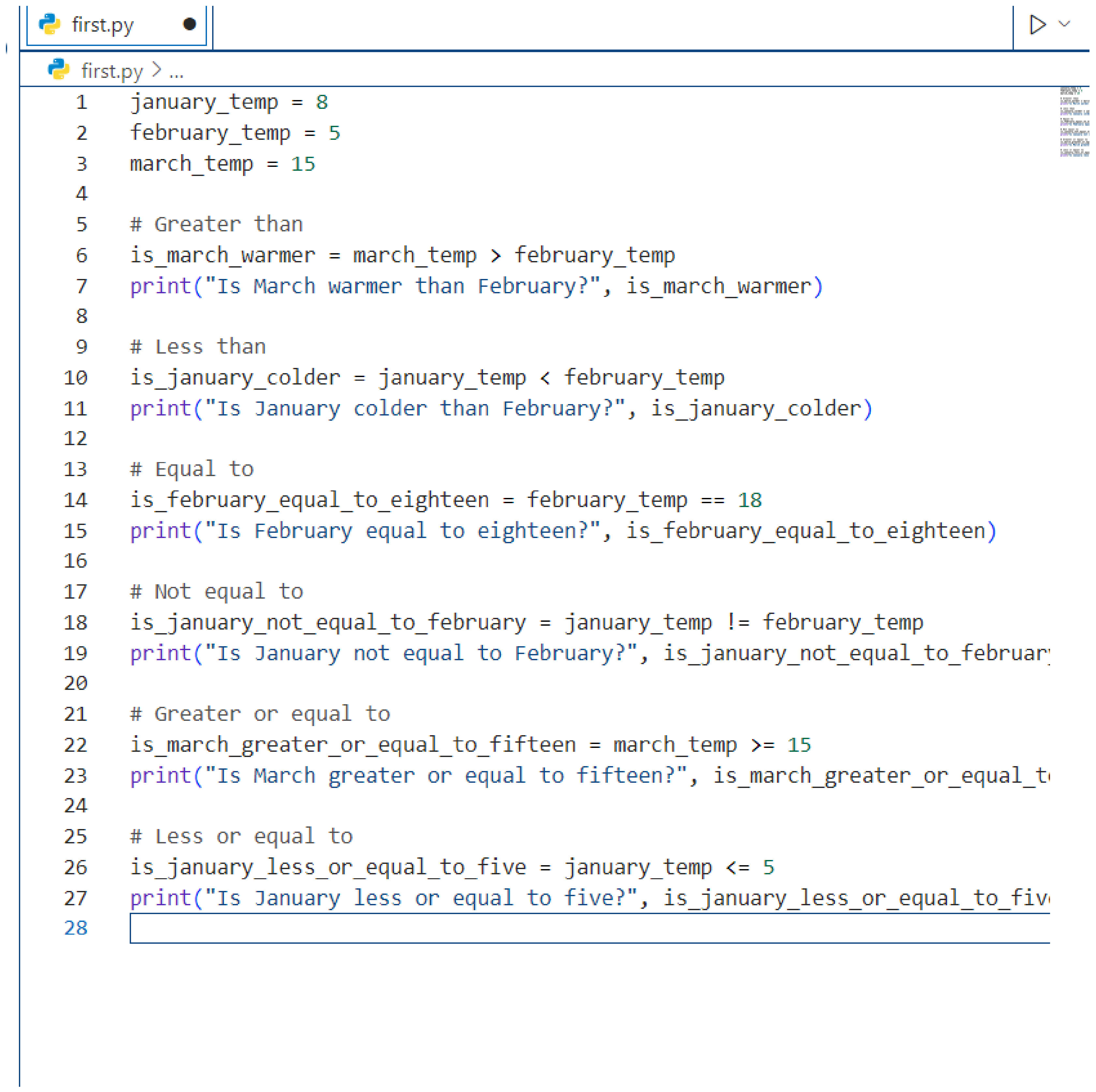


```python
# Example of Sales Data Computation over two weeks
weekly_sales_week1 = 200
weekly_sales_week2 = 350
weekly_sales_week3 = 300

# Addition
total_sales = weekly_sales_week1 + weekly_sales_week2 + weekly_sales_week3
print("Total Sales:", total_sales)

# Subtraction
sales_difference = weekly_sales_week3 - weekly_sales_week1
print("Sales Difference:", sales_difference)

# Multiplication for projected increase projection
projected_sales_week1 = weekly_sales_week1 * 1.1 # 10% increase projection
print("Projected Sales:", projected_sales_week1)

# Division for average sales calculation
average_sales = total_sales / 7
print("Average Sales:", average_sales)

# Modulus to determine remaining items after packaging
total_items = 100
packaged_items = 9
remaining_items = total_items % packaged_items
print("Remaining Items After Packaging:", remaining_items)

# Exponentiation to calculate projected growth over two weeks
growth_rate = 1.05
current_growth_rate = 1.02
projected_growth_total = (growth_rate ** 2) * current_growth_rate
print("Projected Growth Over Two Weeks:", projected_growth_total)

# Floor Division to determine number of full boxes
example_item = 53
full_boxes_size = 7
full_boxes = example_item // full_boxes_size
print("Number of Full Boxes:", full_boxes)
```



```python
# Example: Calculating adjusted sales with tax and discount
initial_sales = 1000

# Assign initial sales to sales variable
sales = initial_sales

# Add and assign
sales += 50  # Adding promotional sales
print("Sales after promotion:", sales)

# Subtract and assign
sales -= 30  # Subtracting returns
print("Sales after returns:", sales)

# Multiply and assign
tax_rate = 0.08  # Tax rate
sales *= (1 + tax_rate)  # Adding tax
print("Sales after tax:", sales)

# Divide and assign
discount_rate = 0.1  # Discount rate
sales /= (1 - discount_rate)  # Applying discount
print("Sales after discount:", sales)

# Modulus and assign
remainder = sales % 7  # Remainder when divided by 7
print("Remainder when divided by 7:", remainder)

# Exponentiation and assign
growth_factor = 1.02  # Growth factor
sales **= growth_factor  # Compounding growth
print("Sales after growth factor:", sales)

# Floor division and assign
sales //= 1.5  # Floor division
print("Sales after floor division:", sales)
```

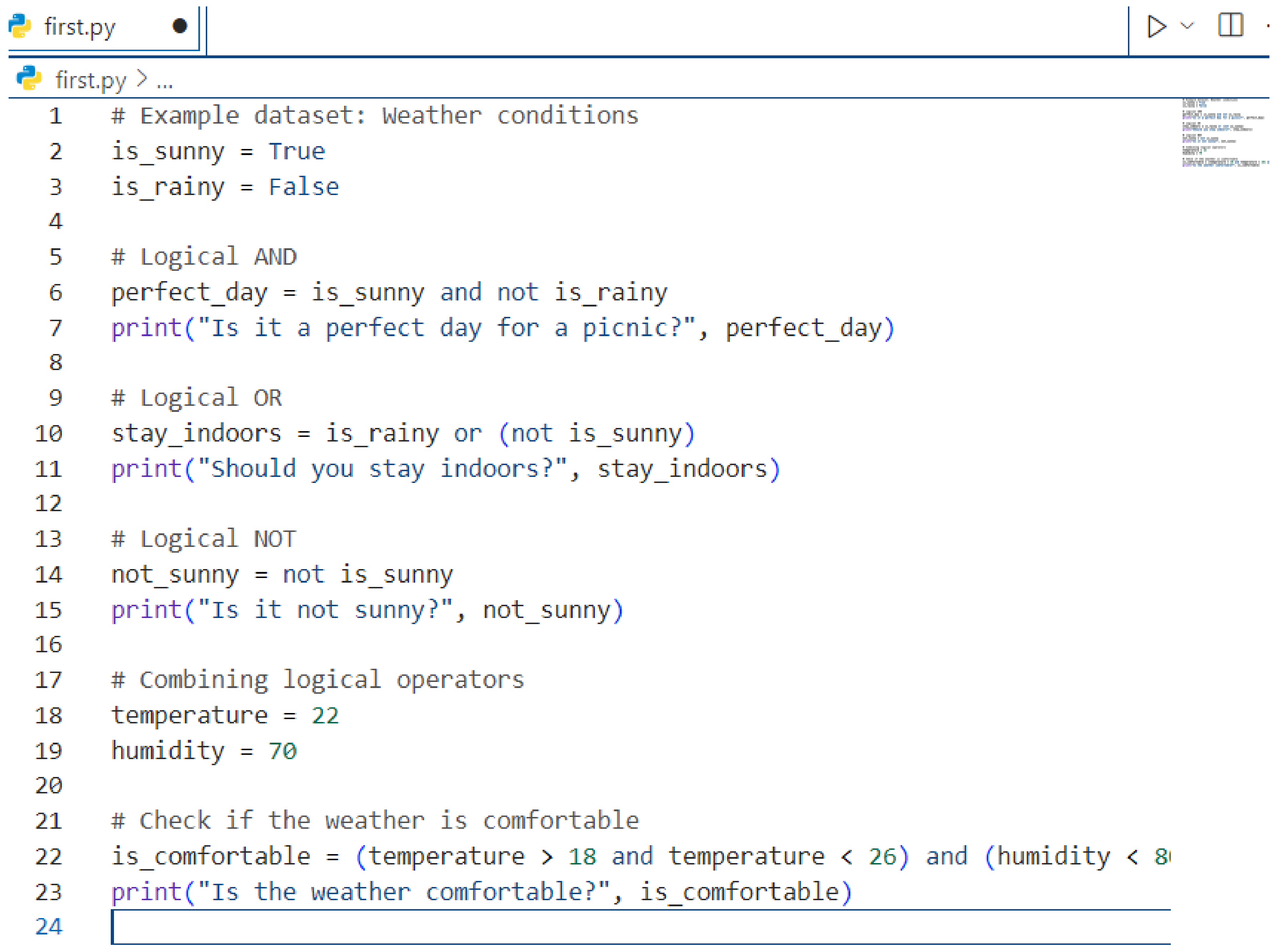
EXAMPLE 04



first.py

first.py > ...

```python
january_temp = 8
february_temp = 5
march_temp = 15

# Greater than
is_march_warmer = march_temp > february_temp
print("Is March warmer than February?", is_march_warmer)

# Less than
is_january_colder = january_temp < february_temp
print("Is January colder than February?", is_january_colder)

# Equal to
is_february_equal_to_eighteen = february_temp == 18
print("Is February equal to eighteen?", is_february_equal_to_eighteen)

# Not equal to
is_january_not_equal_to_february = january_temp != february_temp
print("Is January not equal to February?", is_january_not_equal_to_februar

# Greater or equal to
is_march_greater_or_equal_to_fifteen = march_temp >= 15
print("Is March greater or equal to fifteen?", is_march_greater_or_equal_t

# Less or equal to
is_january_less_or_equal_to_five = january_temp <= 5
print("Is January less or equal to five?", is_january_less_or_equal_to_fiv

```



first.py

first.py > ...

```python
# Example dataset: Weather conditions
is_sunny = True
is_rainy = False

# Logical AND
perfect_day = is_sunny and not is_rainy
print("Is it a perfect day for a picnic?", perfect_day)

# Logical OR
stay_indoors = is_rainy or (not is_sunny)
print("Should you stay indoors?", stay_indoors)

# Logical NOT
not_sunny = not is_sunny
print("Is it not sunny?", not_sunny)

# Combining logical operators
temperature = 22
humidity = 70

# Check if the weather is comfortable
is_comfortable = (temperature > 18 and temperature < 26) and (humidity < 8
print("Is the weather comfortable?", is_comfortable)

```

PRACTICE EXERCISE

1. ចូរសរសេ កូដមួយដែលអនុញ្ញាតអោយគេ បញ្ចូល width និង height របស់ចតុកោណកែងមួយ បន្ទាប់គណនា បរិមាត្រ និង ផ្ទៃក្រឡានៃ ចតុកោណនោះ ហើយបង្ហាញមកលើ console នូវតម្លៃ width, height, បរិមាត្រ និង ផ្ទៃក្រឡា នោះ
2. ចូរសរសេរកូដអោយគេបញ្ចូលពិន្ទុ ៥មុខដូចជា score1(float), score2(float), score3(float), score4(float) និង score5(float) បន្ទាប់មកបង្ហាញទិន្នន័យនោះ ចេញ មកក្រៅវិញរួមមាន score ទាំង ៥មុខ និងពិន្ទុសរុប(total) និង មធ្យមភាគ(average)?
3. ចូរសរសេរកូដអោយគេបញ្ចូលផលិតផល ដូចជា code(int), name(string), qty(int), price(double) និង discount(int) បន្ទាប់មកបង្ហាញពត៏មាន ទាំងនោះ មកក្រៅវិញ រួមទាំង total(double) និង payment(double)?
4. ចូរសរសេ កូដមួយដែលអនុញ្ញាតអោយគេអាចបញ្ចូល តម្លៃចំនួន ៣ បន្ទាប់មកយើងធ្វើការ គណនា និង បង្ហាញចេញមកវិញនូវ តម្លៃមធ្យម តម្លៃធំបំផុត និង តម្លៃតូចបំផុតក្នុងចំណោមនោះ។
5. ចូរសរសេ កូដដែលអនុញ្ញាតអោយគេបញ្ចូលនូវតម្លៃ នៃ ប្រាក់កម្ចីសរុប(principle) អត្រាការប្រាក់(rate) និង រយៈពេលត្រូវបង់សរុប(time) បន្ទាប់មកធ្វើការគណនា ចំនួនការប្រាក់សរុបត្រូវបង់ និង បង្ហាញទិន្នន័យទាំងនោះចេញមកវិញ ។

CONTROL STATEMENTS

Control Statements គឺជាការគ្រប់គ្រងដំណើរការរបស់ code ក្នុងលក្ខណ្ឌណាមួយ ដើម្បីអោយវាអនុវត្តបាន។ Control Statements សំដៅលើការ គ្រប់គ្រងលើ លំហូនៃ ដំណើរការ របស់កូដ។

នៅក្នុង Control Statements មាន Type ដូចជា៖

- Condition Statement
- Loop Statement

CONDITION

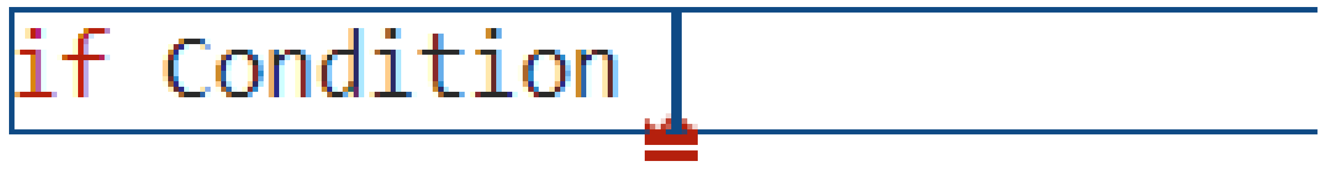
Condition Statement គឺជាការកំណត់នូវលក្ខខណ្ឌដើម្បីអោយកូដនោះអាចអនុវត្តបានក្រោមលក្ខណ្ឌពីរគឺលក្ខណ្ឌ ពិត (true) ឬ មិនពិត (false) ហើយកូដ នឹង អនុវត្តនៅពេលដែលលក្ខណ្ឌនោះពិត។

នៅក្នុង Condition Statement ត្រូវបានបែងចែកជា 3 ដូចជា៖

1. if statement
2. if else statement
3. if elif else statement

if statement: គឺជា Condition ដែលអាចអោយកូដអនុវត្តបានក្រោមលក្ខណ្ឌពិតបើសិនជាមិនពិតវាមិនកូដ មិនអាចអនុវត្តបាននោះទេ

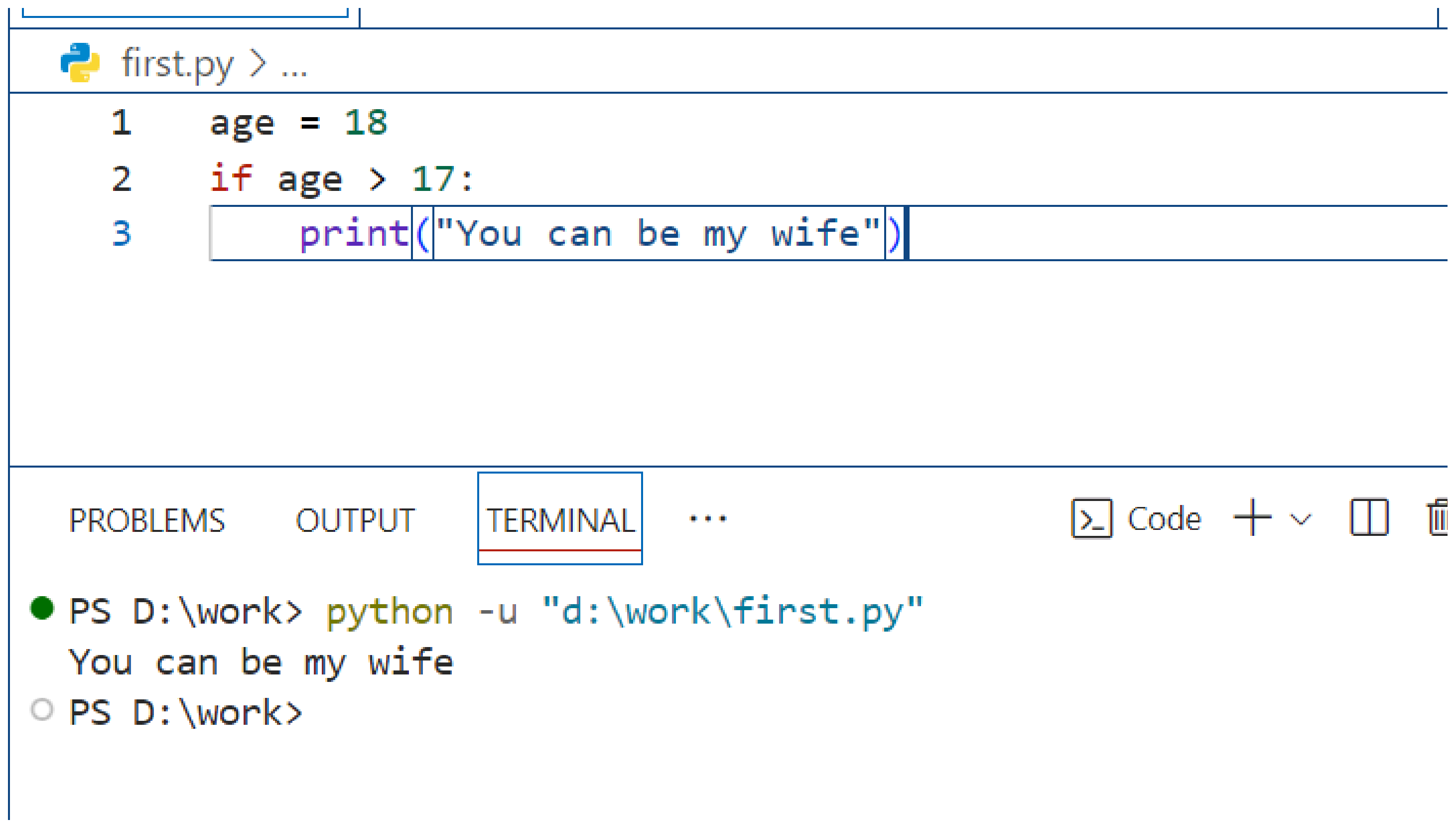
Syntax:



```
if Condition
```

note: ត្រូវចាំថាការសរសេរកូដនៅក្រោម control statement ត្រូវចុះបន្ទាត់ជានិច

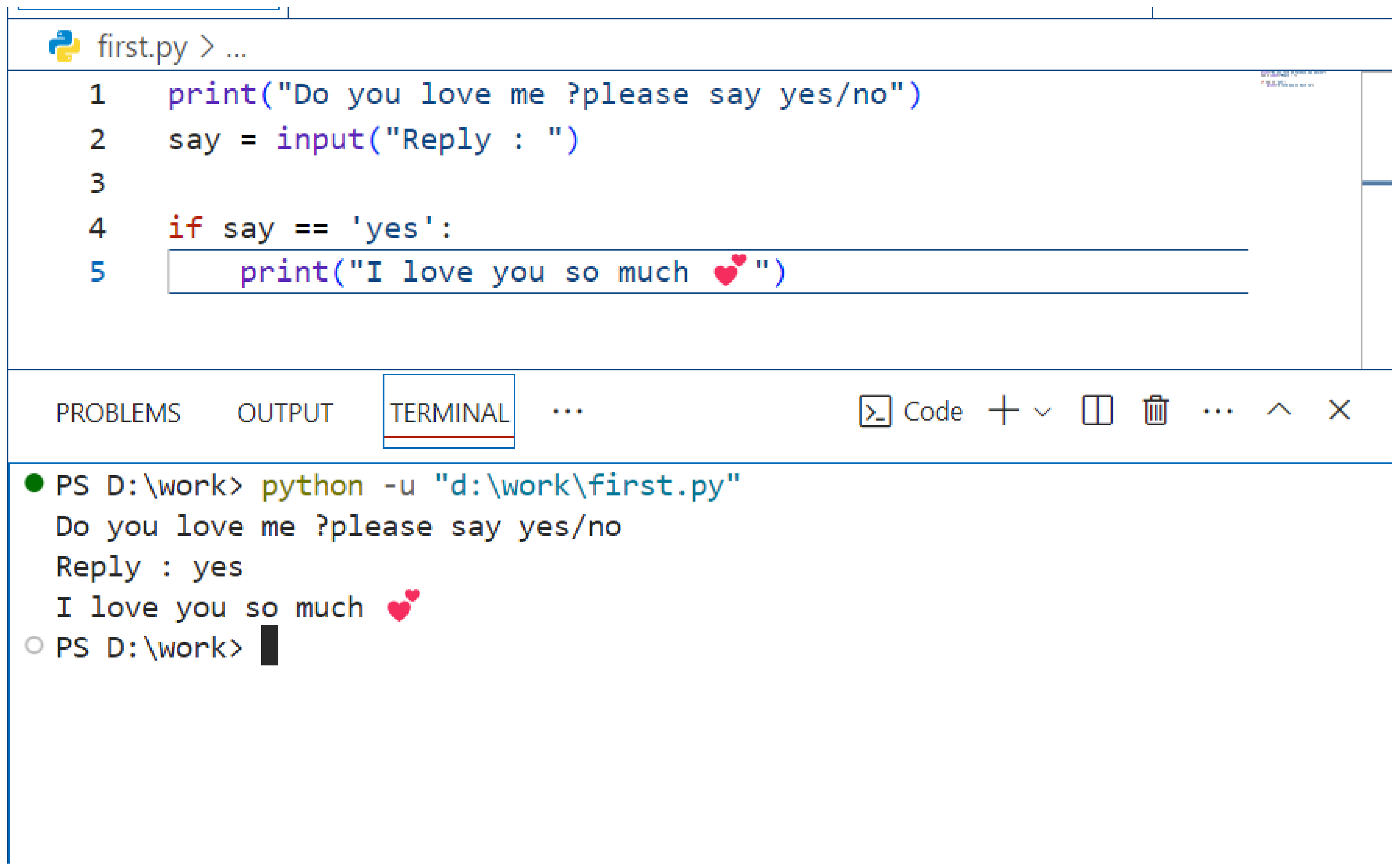
Example:



first.py > ...

```python
age = 18
if age > 17:
    print("You can be my wife")
```

```
PS D:\work> python -u "d:\work\first.py"
You can be my wife
PS D:\work>
```



first.py > ...

```python
print("Do you love me ?please say yes/no")
say = input("Reply : ")

if say == 'yes':
    print("I love you so much 💕")
```

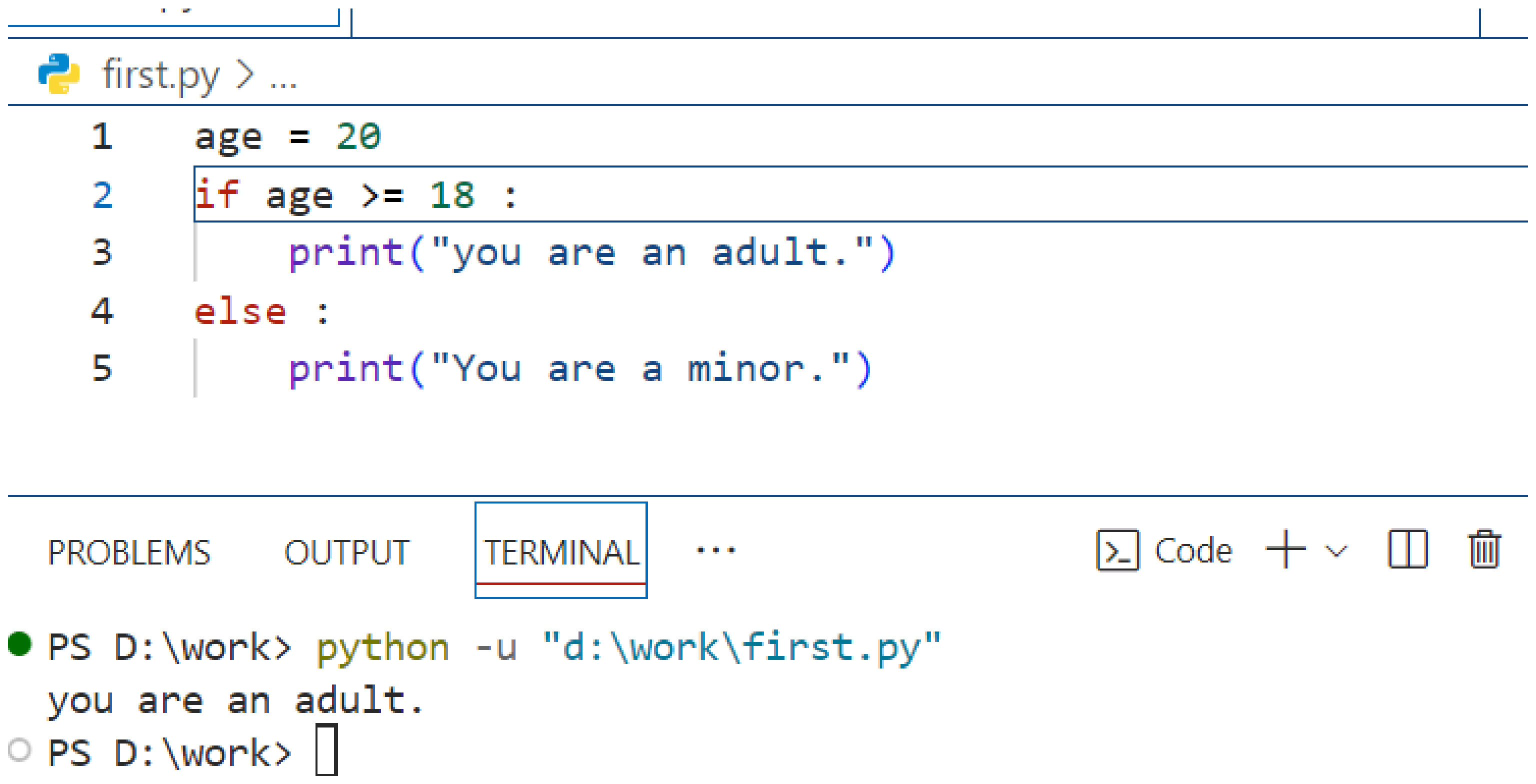
```
PS D:\work> python -u "d:\work\first.py"
Do you love me ?please say yes/no
Reply : yes
I love you so much 💕
PS D:\work>
```

if else statement: គឺជា Condition ដែលអាចអោយកូដអនុវត្តបានក្រោមលក្ខណ្ឌពីរ បើសិនជាលក្ខណ្ឌពិតវានិងអនុវត្តកូដក្នុងលក្ខណ្ឌ if បើសិនជាមិនពិតវានឹងចូលធ្វើការជាមួយលក្ខណ្ឌ else

Syntax:

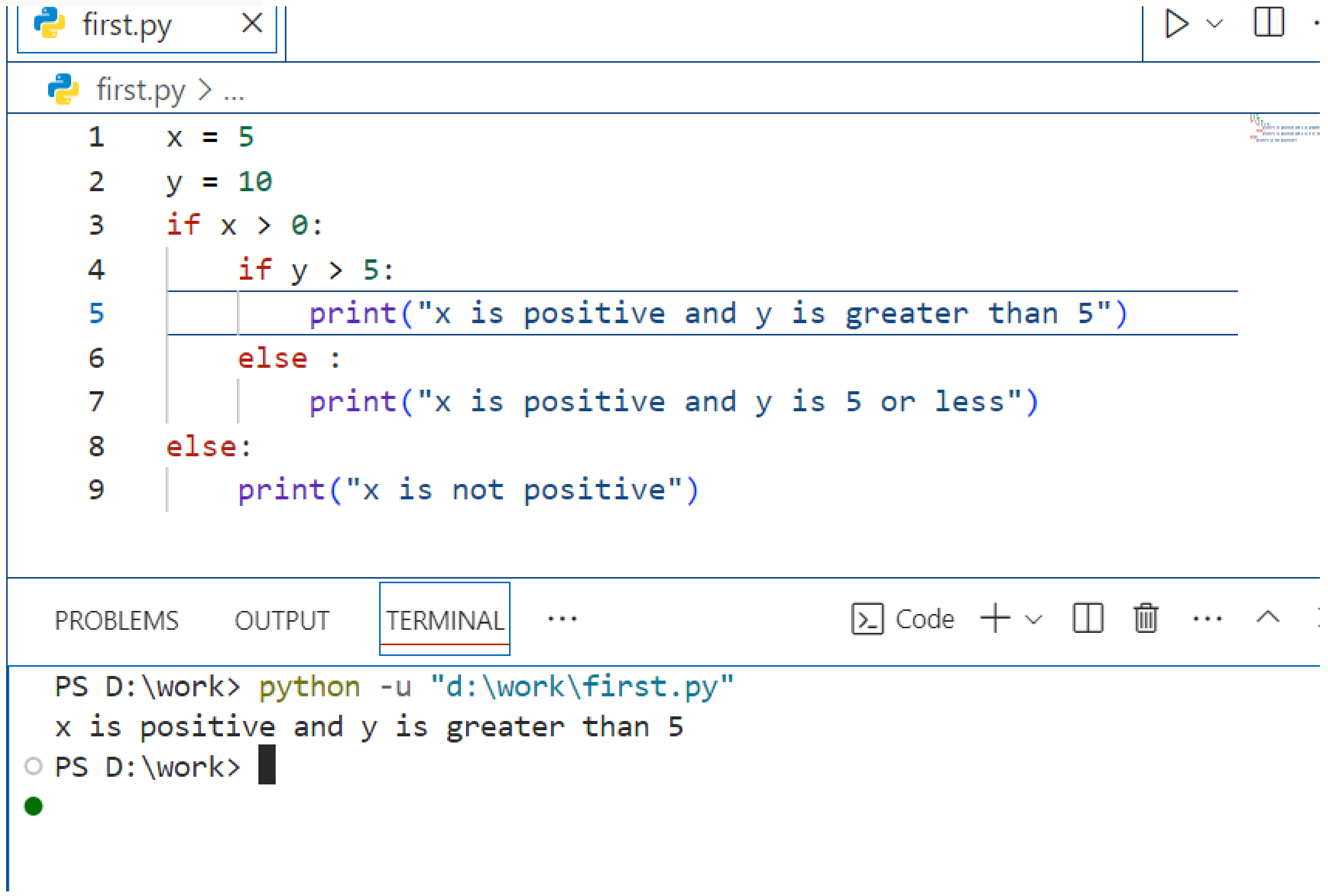
```
if condition :
    # code...
else :
    # code...
```

Example:



```python
age = 20
if age >= 18 :
    print("you are an adult.")
else :
    print("You are a minor.")
```

```
PS D:\work> python -u "d:\work\first.py"
you are an adult.
PS D:\work>
```



```python
x = 5
y = 10
if x > 0:
    if y > 5:
        print("x is positive and y is greater than 5")
    else :
        print("x is positive and y is 5 or less")
else:
    print("x is not positive")
```

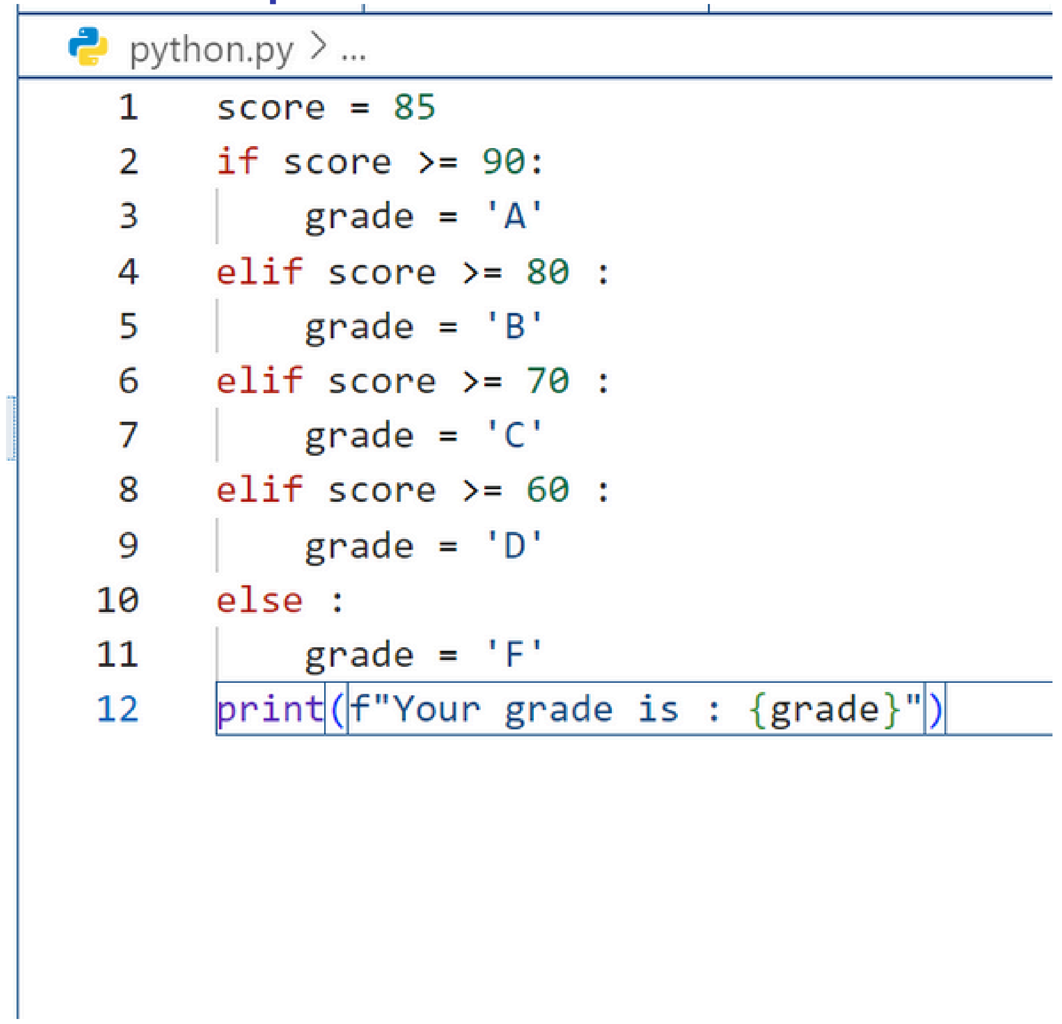
```
PS D:\work> python -u "d:\work\first.py"
x is positive and y is greater than 5
PS D:\work>
```

if elif else statement: គឺជា Condition ដែលយើងអាចកំណត់នៅបានច្រើនចាប់ពីពីរឡើងទៅ

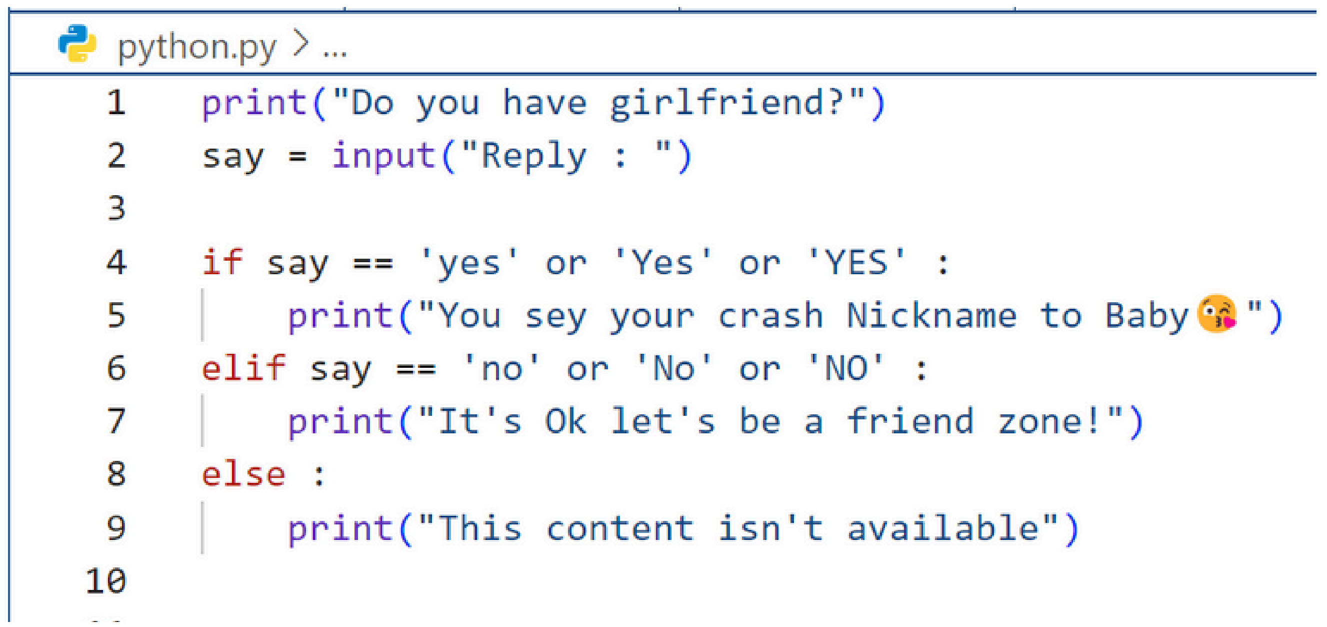
Syntax:

```
if condition1 :
    # code to excite if condition1 is True
elseif condition2 :
    # code to excite if condition2 True
else :
    # code to excite if both condition1 and condition2 are false
```

Example:

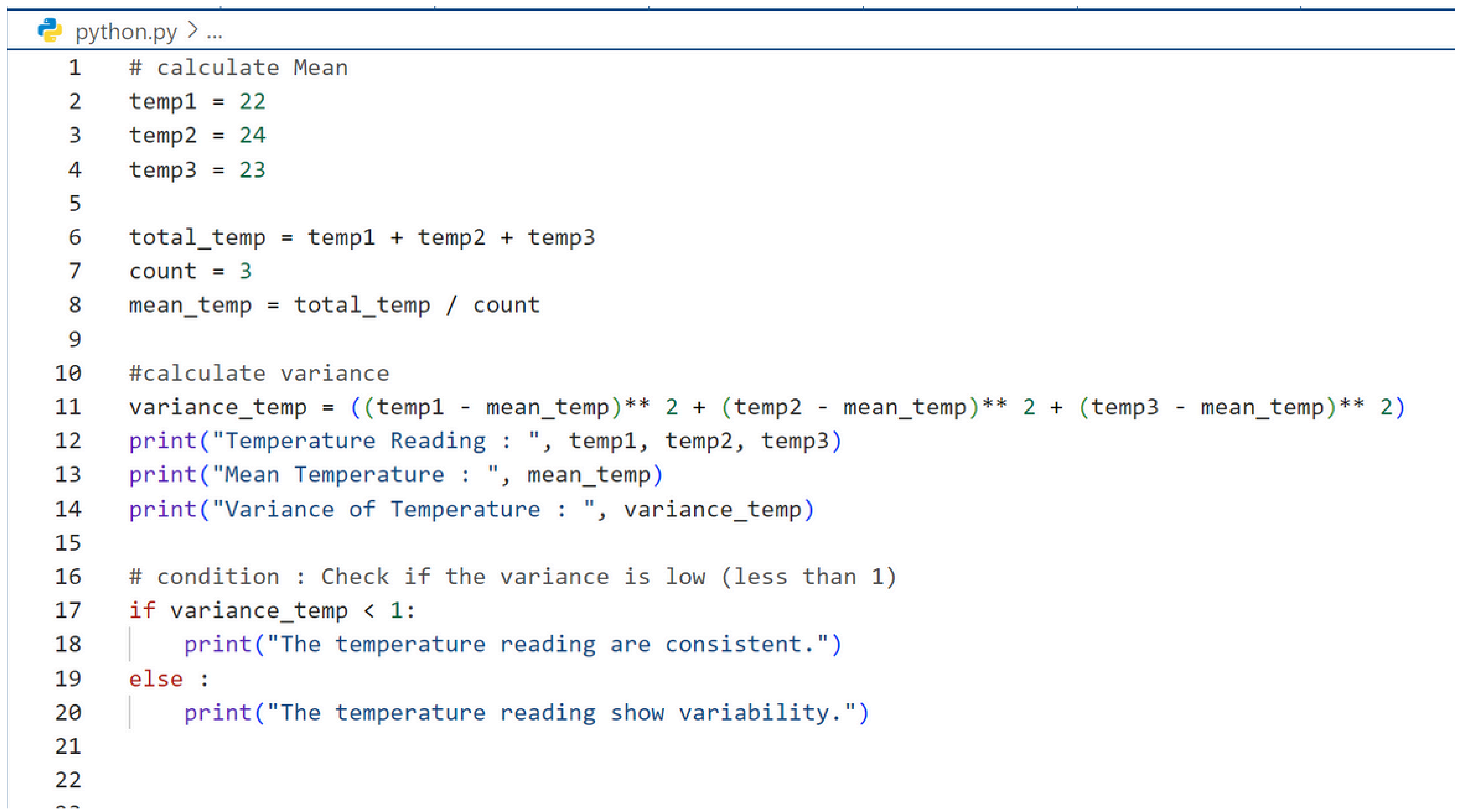

python.py > ...

```python
score = 85
if score >= 90:
    grade = 'A'
elif score >= 80 :
    grade = 'B'
elif score >= 70 :
    grade = 'C'
elif score >= 60 :
    grade = 'D'
else :
    grade = 'F'
print(f"Your grade is : {grade}")
```


python.py > ...

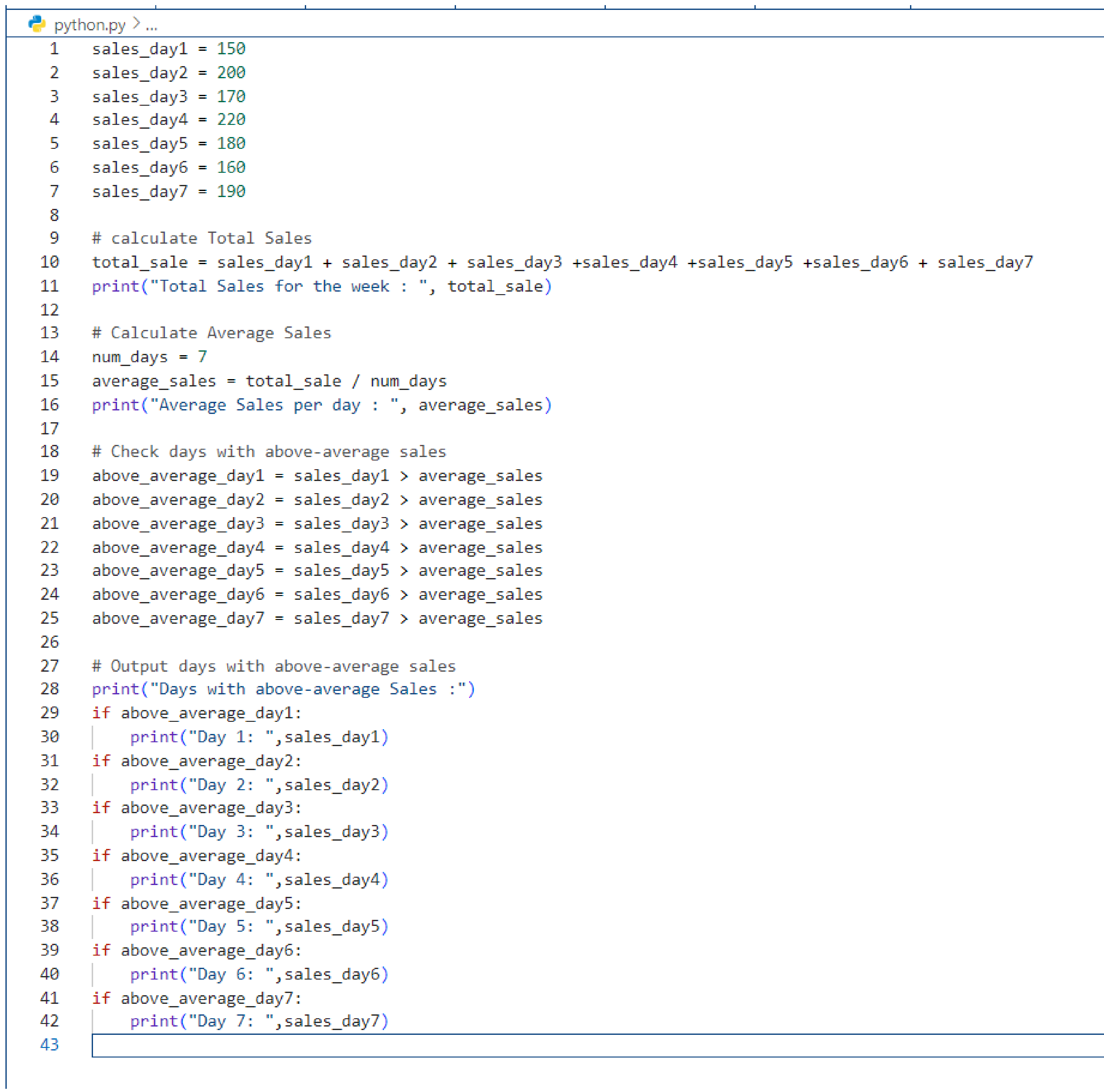
```python
print("Do you have girlfriend?")
say = input("Reply : ")

if say == 'yes' or 'Yes' or 'YES' :
    print("You sey your crash Nickname to Baby😘")
elif say == 'no' or 'No' or 'NO' :
    print("It's Ok let's be a friend zone!")
else :
    print("This content isn't available")
```

EXAMPLE 01



python.py > ...

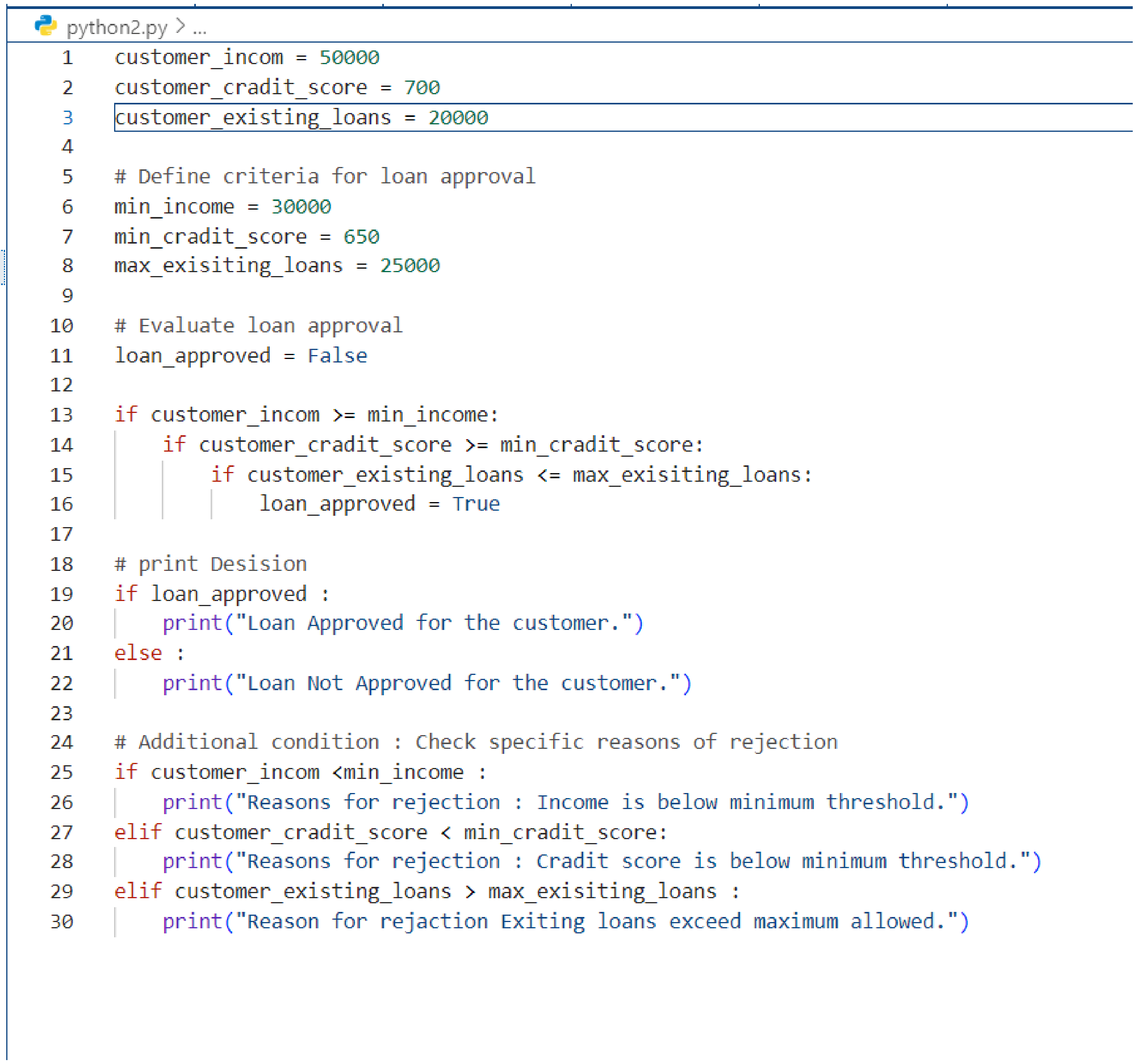
```python
# calculate Mean
temp1 = 22
temp2 = 24
temp3 = 23

total_temp = temp1 + temp2 + temp3
count = 3
mean_temp = total_temp / count

#calculate variance
variance_temp = ((temp1 - mean_temp)** 2 + (temp2 - mean_temp)** 2 + (temp3 - mean_temp)** 2)
print("Temperature Reading : ", temp1, temp2, temp3)
print("Mean Temperature : ", mean_temp)
print("Variance of Temperature : ", variance_temp)

# condition : Check if the variance is low (less than 1)
if variance_temp < 1:
    print("The temperature reading are consistent.")
else :
    print("The temperature reading show variability.")


```



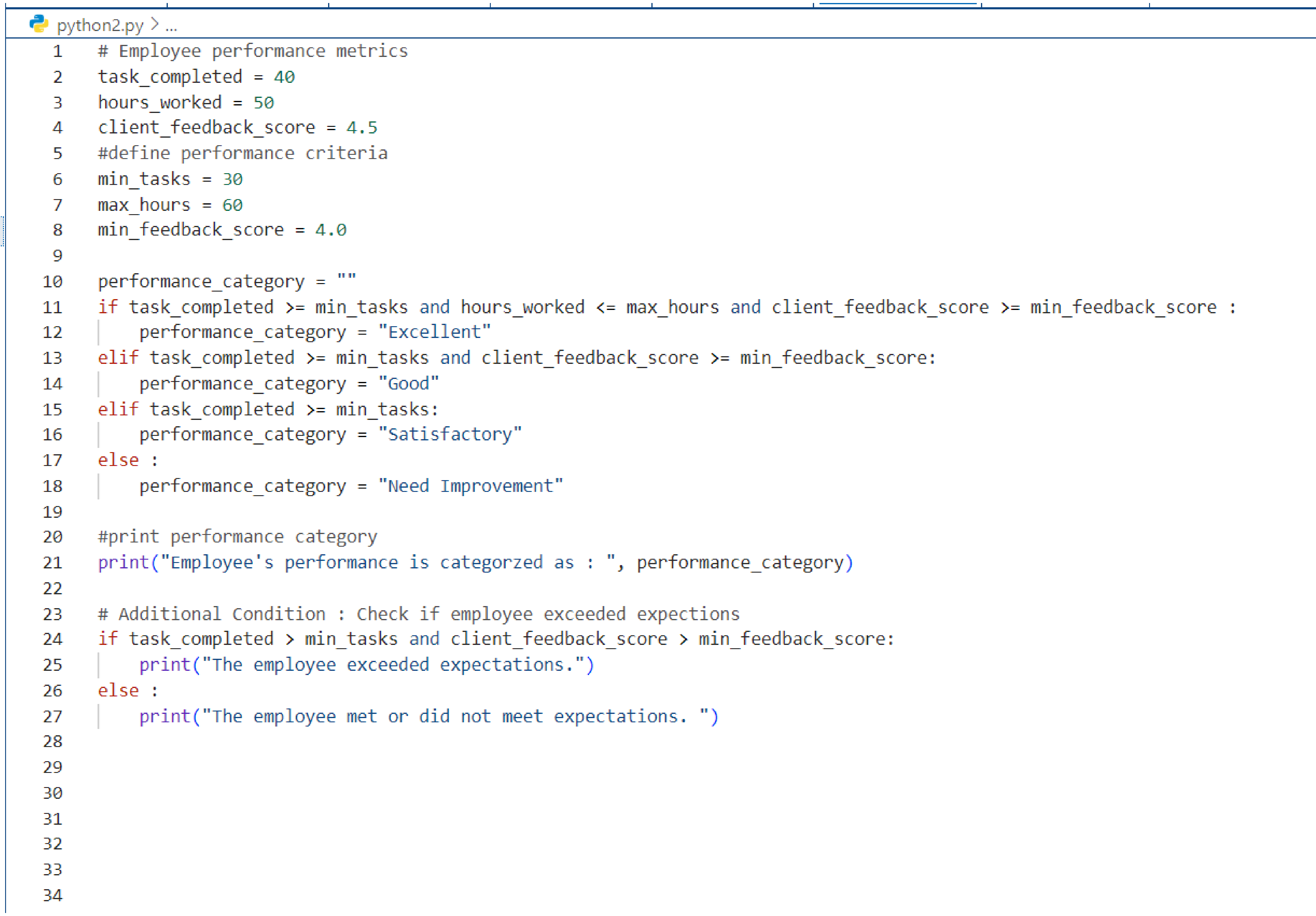
python.py > ...

```python
sales_day1 = 150
sales_day2 = 200
sales_day3 = 170
sales_day4 = 220
sales_day5 = 180
sales_day6 = 160
sales_day7 = 190

# calculate Total Sales
total_sale = sales_day1 + sales_day2 + sales_day3 +sales_day4 +sales_day5 +sales_day6 + sales_day7
print("Total Sales for the week : ", total_sale)

# Calculate Average Sales
num_days = 7
average_sales = total_sale / num_days
print("Average Sales per day : ", average_sales)

# Check days with above-average sales
above_average_day1 = sales_day1 > average_sales
above_average_day2 = sales_day2 > average_sales
above_average_day3 = sales_day3 > average_sales
above_average_day4 = sales_day4 > average_sales
above_average_day5 = sales_day5 > average_sales
above_average_day6 = sales_day6 > average_sales
above_average_day7 = sales_day7 > average_sales

# Output days with above-average sales
print("Days with above-average Sales :")
if above_average_day1:
    print("Day 1: ",sales_day1)
if above_average_day2:
    print("Day 2: ",sales_day2)
if above_average_day3:
    print("Day 3: ",sales_day3)
if above_average_day4:
    print("Day 4: ",sales_day4)
if above_average_day5:
    print("Day 5: ",sales_day5)
if above_average_day6:
    print("Day 6: ",sales_day6)
if above_average_day7:
    print("Day 7: ",sales_day7)

```

EXAMPLE 02


python2.py > ...

```python
customer_incom = 50000
customer_cradit_score = 700
customer_existing_loans = 20000

# Define criteria for loan approval
min_income = 30000
min_cradit_score = 650
max_exisiting_loans = 25000

# Evaluate loan approval
loan_approved = False

if customer_incom >= min_income:
    if customer_cradit_score >= min_cradit_score:
        if customer_existing_loans <= max_exisiting_loans:
            loan_approved = True

# print Desision
if loan_approved :
    print("Loan Approved for the customer.")
else :
    print("Loan Not Approved for the customer.")

# Additional condition : Check specific reasons of rejection
if customer_incom <min_income :
    print("Reasons for rejection : Income is below minimum threshold.")
elif customer_cradit_score < min_cradit_score:
    print("Reasons for rejection : Cradit score is below minimum threshold.")
elif customer_existing_loans > max_exisiting_loans :
    print("Reason for rejection Exiting loans exceed maximum allowed.")
```


python2.py > ...

```python
# Employee performance metrics
task_completed = 40
hours_worked = 50
client_feedback_score = 4.5
#define performance criteria
min_tasks = 30
max_hours = 60
min_feedback_score = 4.0

performance_category = ""
if task_completed >= min_tasks and hours_worked <= max_hours and client_feedback_score >= min_feedback_score :
    performance_category = "Excellent"
elif task_completed >= min_tasks and client_feedback_score >= min_feedback_score:
    performance_category = "Good"
elif task_completed >= min_tasks:
    performance_category = "Satisfactory"
else :
    performance_category = "Need Improvement"

#print performance category
print("Employee's performance is categorzed as : ", performance_category)

# Additional Condition : Check if employee exceeded expections
if task_completed > min_tasks and client_feedback_score > min_feedback_score:
    print("The employee exceeded expectations.")
else :
    print("The employee met or did not meet expectations. ")
```

LOOP

Loop គឺសំដៅលើការ វង្វិលជុំដែលធ្វើការម្តងហើយម្តងទៀតរហូតដល់លក្ខខណ្ឌចុងក្រោយឬក៏លក្ខខណ្ឌបញ្ឈប់ណាមួយ ។

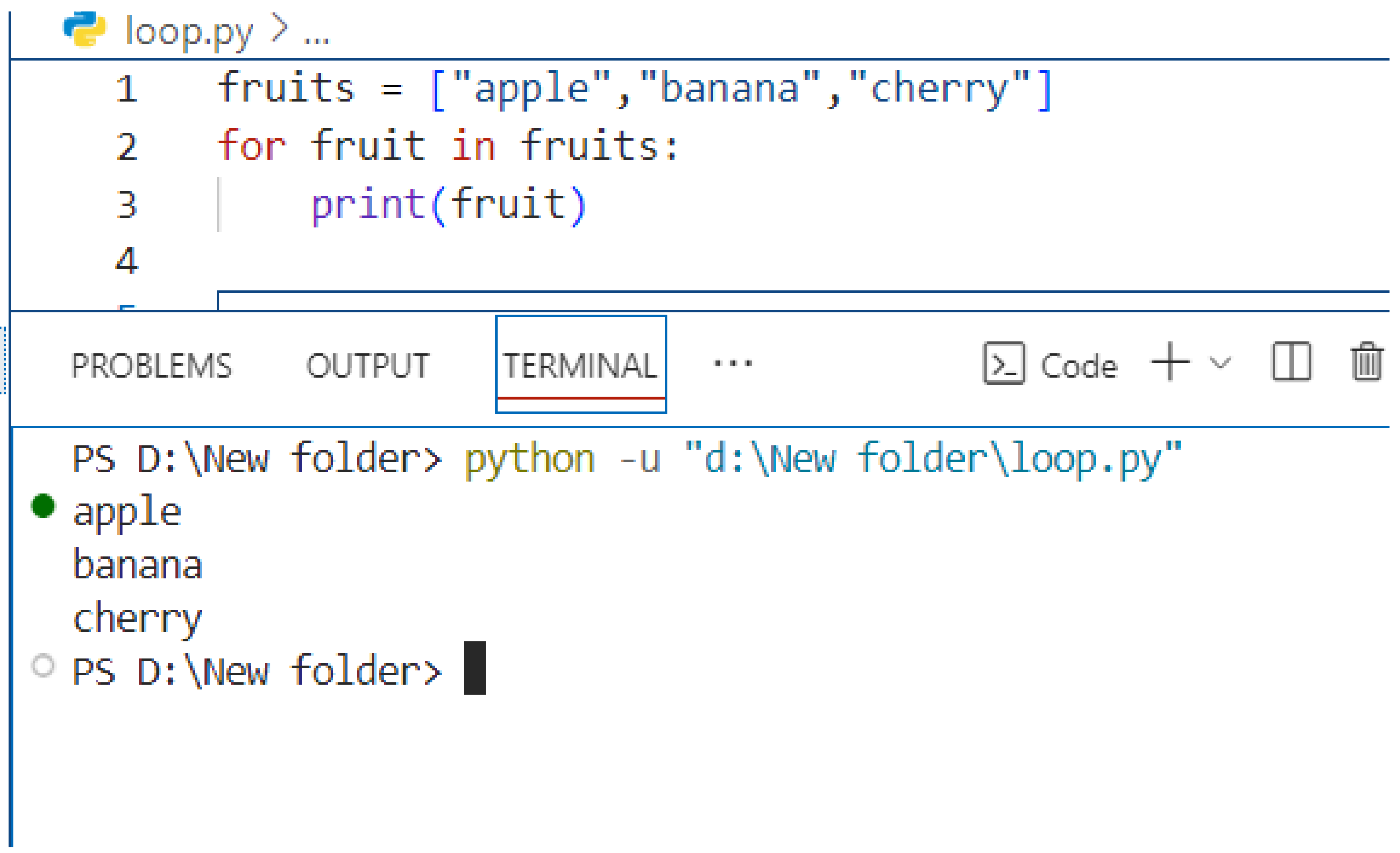
នៅក្នុង Python Loop ត្រូវបានបែងចែកជា ពីរ គឺ៖ for និង while

for : គឺជា loop ដែលធ្វើការ តាមលំដាប់លំដោយរហូតដល់លក្ខណ្ឌចុងក្រោយ

Syntax:

```
for item in sequence :
    # code to execute for each item
```

Example

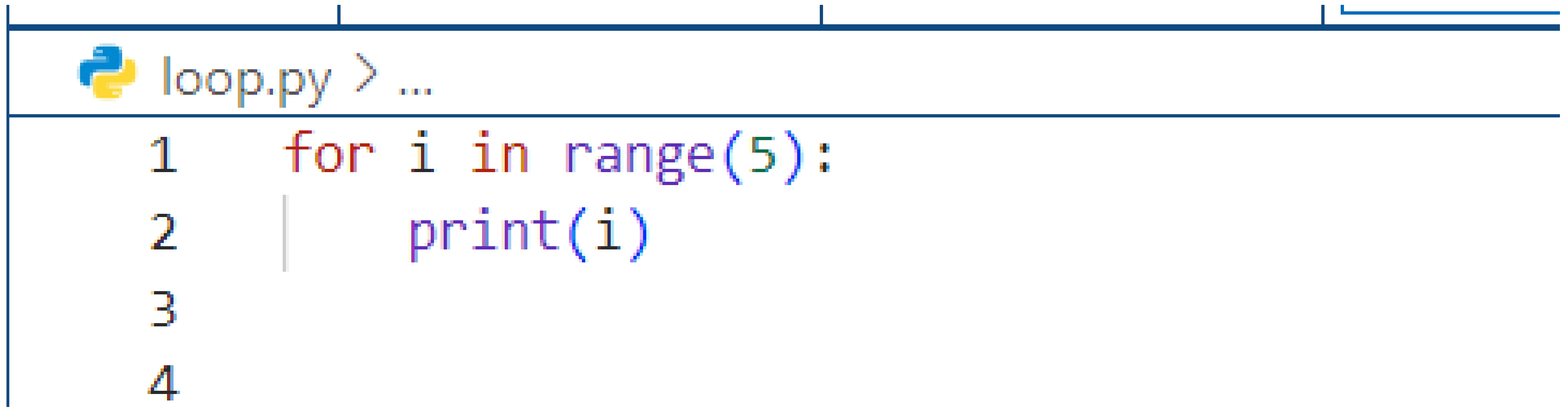


loop.py > ...

```python
fruits = ["apple","banana","cherry"]
for fruit in fruits:
    print(fruit)

```

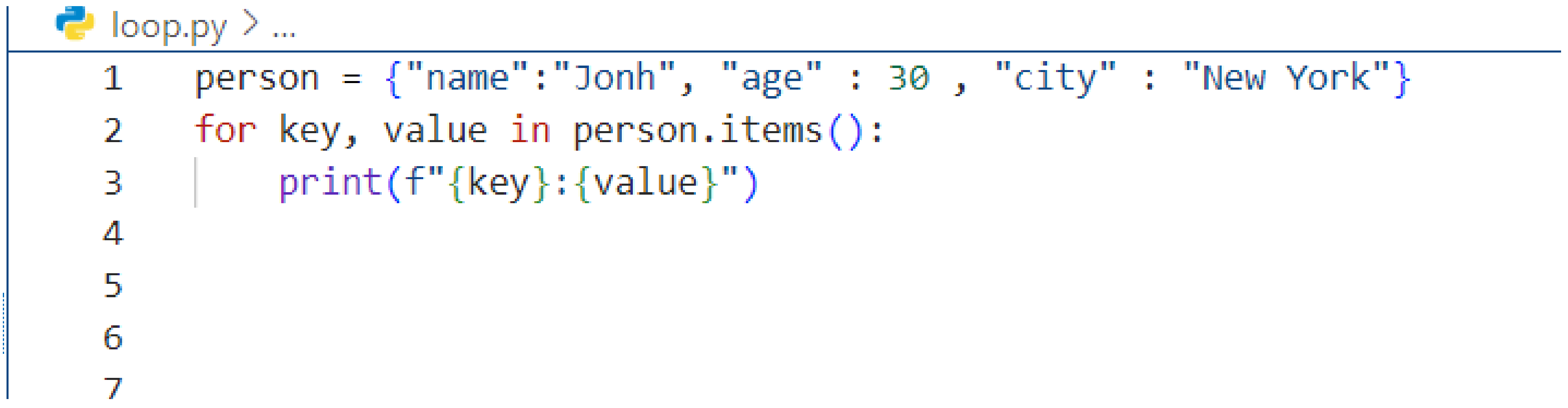
PROBLEMS OUTPUT TERMINAL ... Code

```
PS D:\New folder> python -u "d:\New folder\loop.py"
apple
banana
cherry
PS D:\New folder>
```



loop.py > ...

```python
for i in range(5):
    print(i)


```



loop.py > ...

```python
person = {"name":"Jonh", "age" : 30 , "city" : "New York"}
for key, value in person.items():
    print(f"{key}:{value}")




```

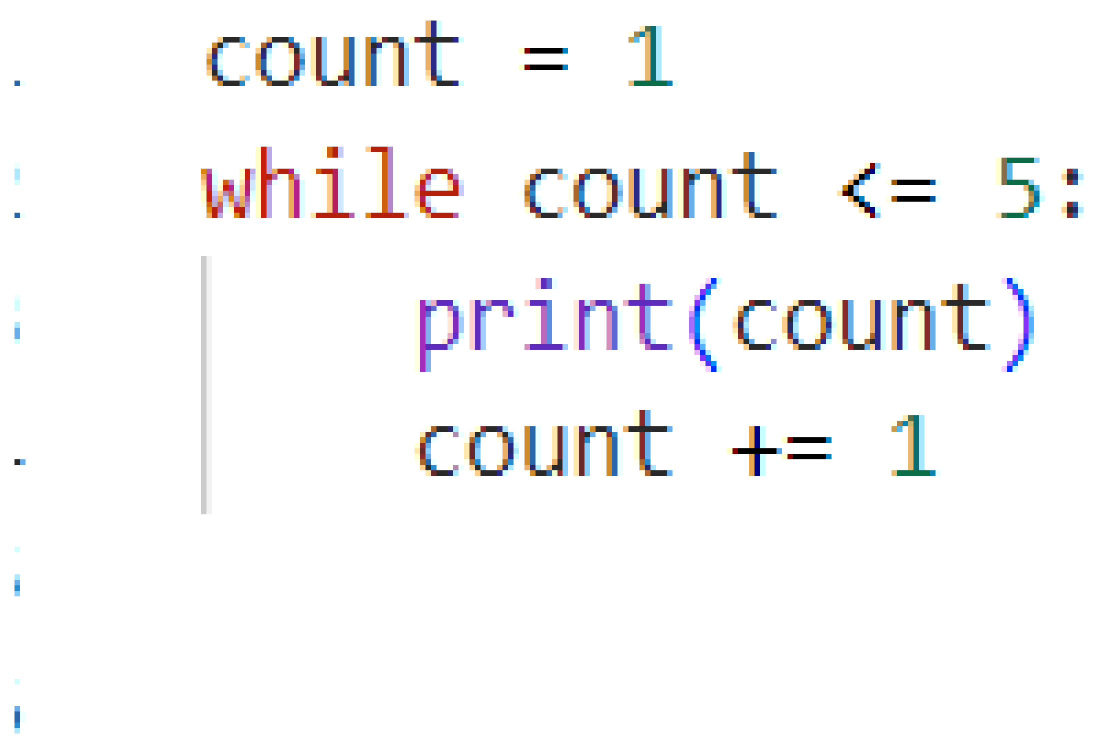
```python
for char in "Hello":
    print(char)
```

while : គឺជា loop ដែលអនុវត្តនូវ Statement Code នៅពេលដែលលក្ខណ្ឌពិត ។ ពោលគឺវាធ្វើការប្រៀបធៀបលក្ខណ្ឌជាមុនសិន។

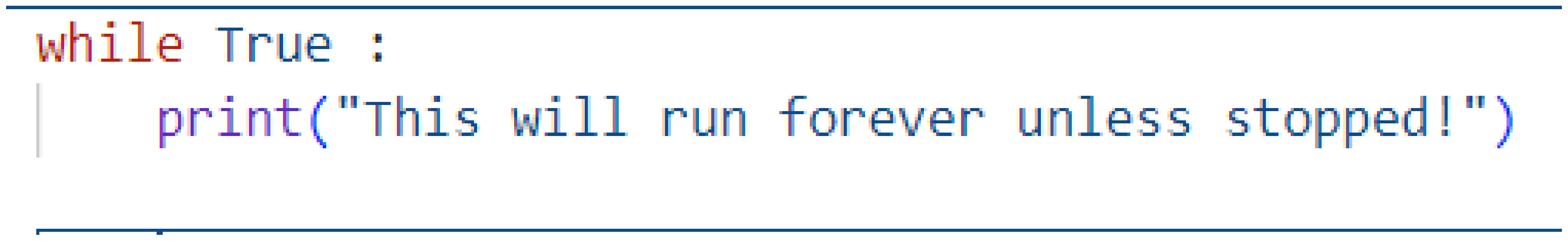
Syntax:

```
while condition :
      # code to execute while the condition is True
```

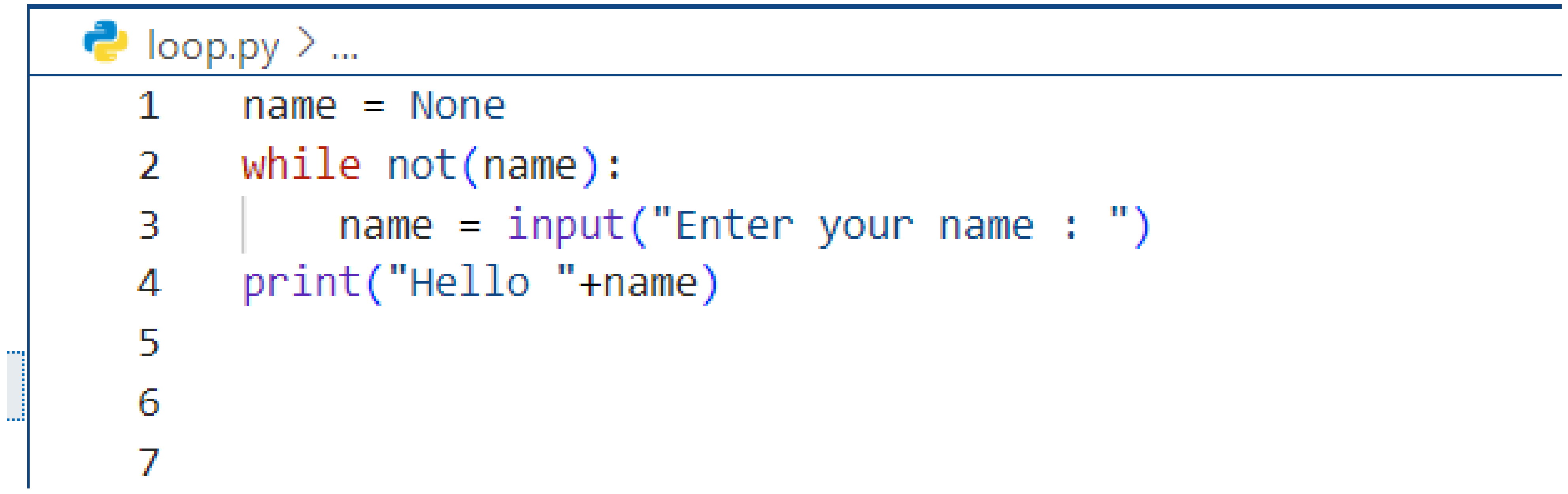
Example



```
count = 1
while count <= 5:
    print(count)
    count += 1
```



```
while True :
    print("This will run forever unless stopped!")
```



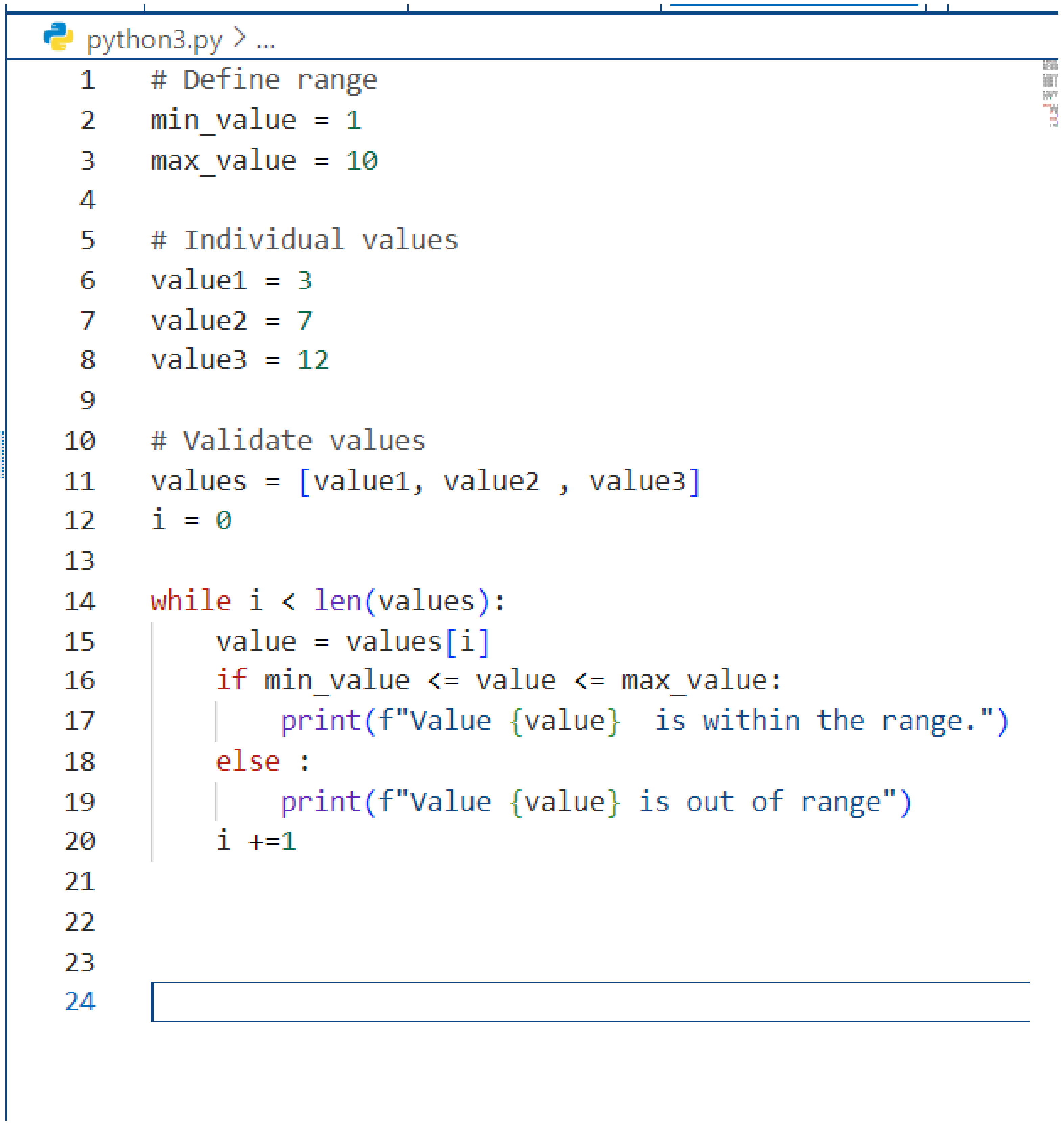
loop.py > ...

```
name = None
while not(name):
    name = input("Enter your name : ")
print("Hello "+name)
```

EXAMPLE 01

```python
number = 5
factorial = 1

for i in range(1, number + 1):
    factorial *= i

print(f"The factorial for {number} is {factorial}")
```

```python
data1  = 10
data2  = 20
data3  = 30
data4  = 40
data5  = 50

# Calculate mean
total = data1 + data2 + data3 + data4 + data5
count = 5
mean = total / count

print(f"The mean of the data poins is {mean}")
```

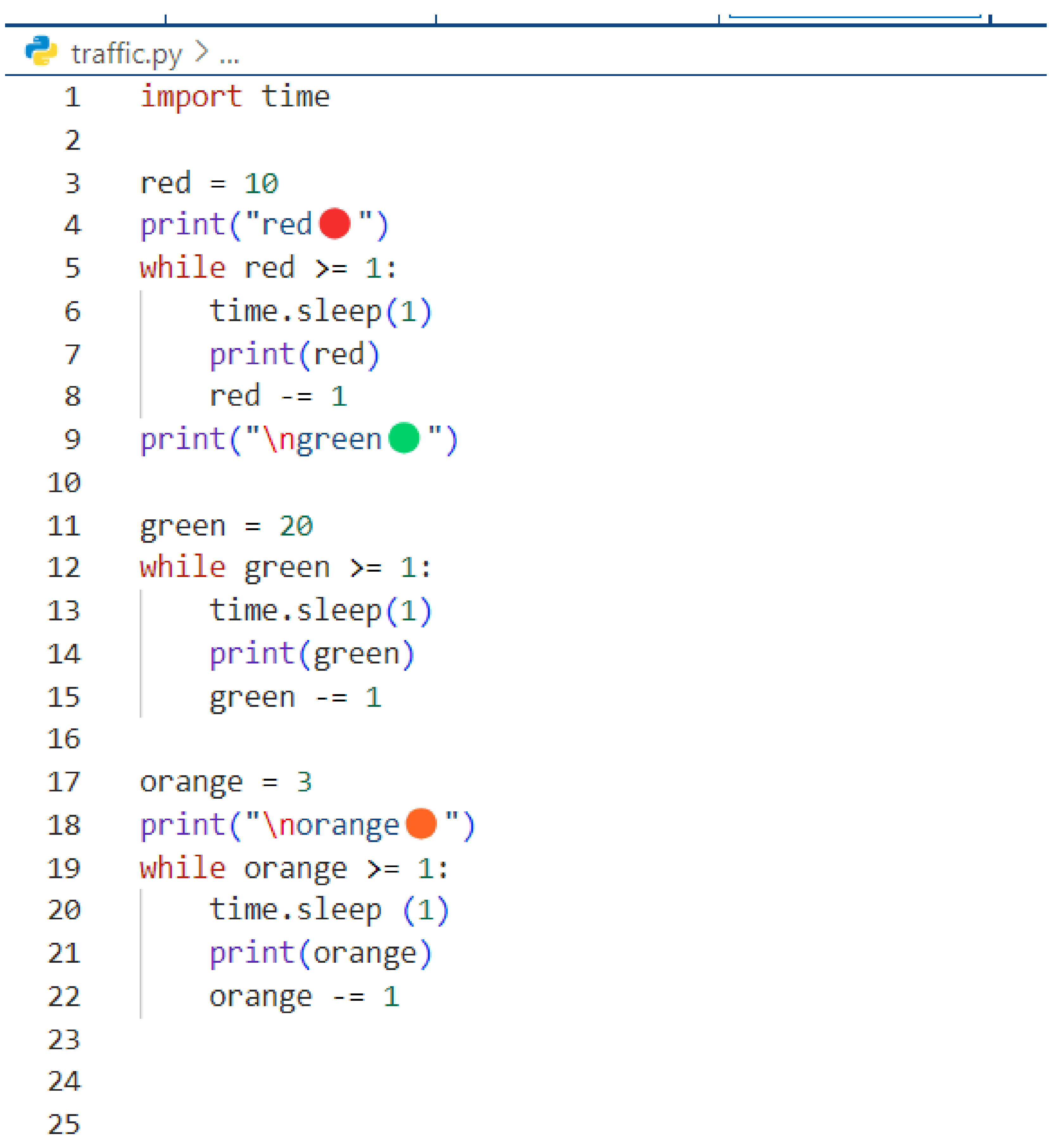


python3.py > ...

```python
# Define range
min_value = 1
max_value = 10

# Individual values
value1 = 3
value2 = 7
value3 = 12

# Validate values
values = [value1, value2 , value3]
i = 0

while i < len(values):
    value = values[i]
    if min_value <= value <= max_value:
        print(f"Value {value}  is within the range.")
    else :
        print(f"Value {value} is out of range")
    i +=1
```

EXAMPLE 02

Simple Traffic light exercise



traffic.py > ...

```python
import time

red = 10
print("red🔴")
while red >= 1:
    time.sleep(1)
    print(red)
    red -= 1
print("\ngreen🟢")

green = 20
while green >= 1:
    time.sleep(1)
    print(green)
    green -= 1

orange = 3
print("\norange🟠")
while orange >= 1:
    time.sleep (1)
    print(orange)
    orange -= 1
```

PRACTICE EXERCISE

1. ចូរសរសេរកូដអោយគេបញ្ចូលផលិតផល ដូចជា code(int), name(string), qty(int), price(double) បន្ទាប់មកបង្ហាញពត៏មាន ទាំងនោះមកក្រៅវិញ រួមទាំង total(float) និង payment(float)?
2. ចូរសរសេរកូដអោយគេបញ្ចូលពិន្ទុ ៥មុខដូចជា score1(float), score2(float), score3(float), score4(float) និង score5(float) បន្ទាប់មកបង្ហាញទិន្នន័យនោះ ចេញ មកក្រៅវិញរួមមាន score ទាំង ៥មុខ និងពិន្ទុសរុប(total),មធ្យមភាគ(average) និង Grade?

តំលៃសរុប(Total)	បញ្ចុះ តំលៃ %
1 ដល់ 10$	10%
10 ដល់ 20$	20%
20 ដល់ 30$	30%
30 ដល់40$	40%
40 ដល់ 50$	50%
50 ដល់ 60$	60%
60$-	70%

មធ្យមភាព/Average	និទេស/Grade
90-100	A
80-90	B
70-80	C
60-70	D
50-60	E
0-50	F

PRACTICE EXERCISE

3.សរសេកូដ បង្កើត Guessing number game ដោយប្រើប្រាស់នូវ concept ចេញពី loop និង condition។

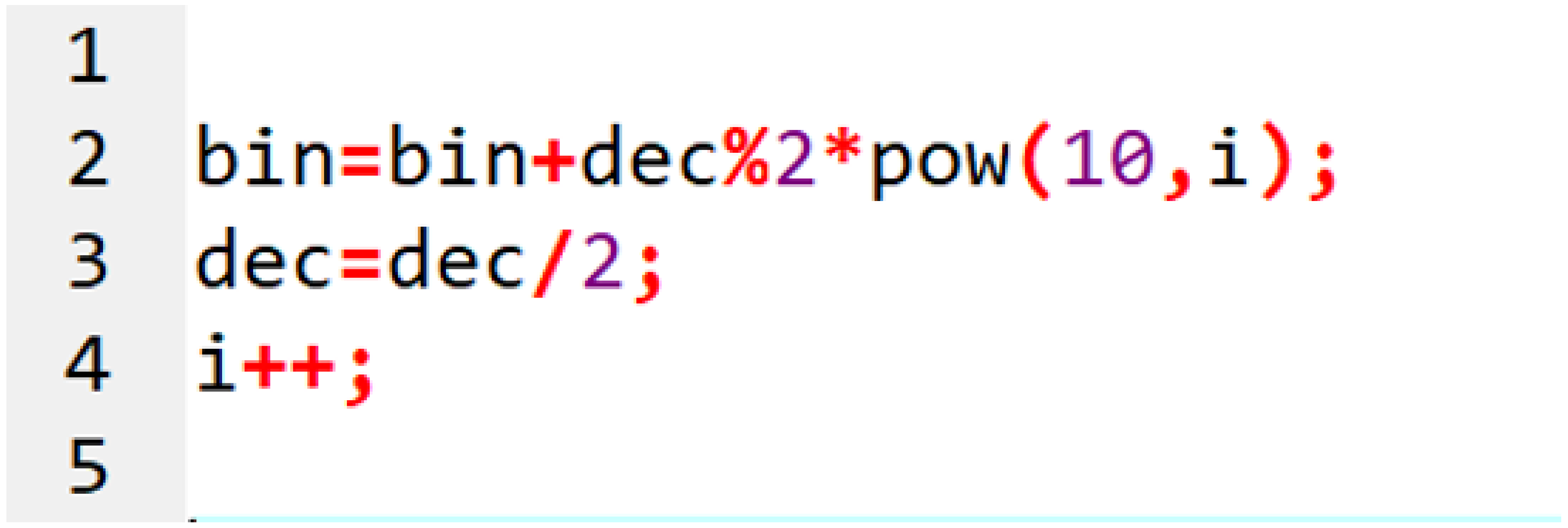
4.ចូរគណនាផលបូក និងគូរ Algorithm flowchart ដូចខាងក្រោម៖

A) 2+4+6+.....N

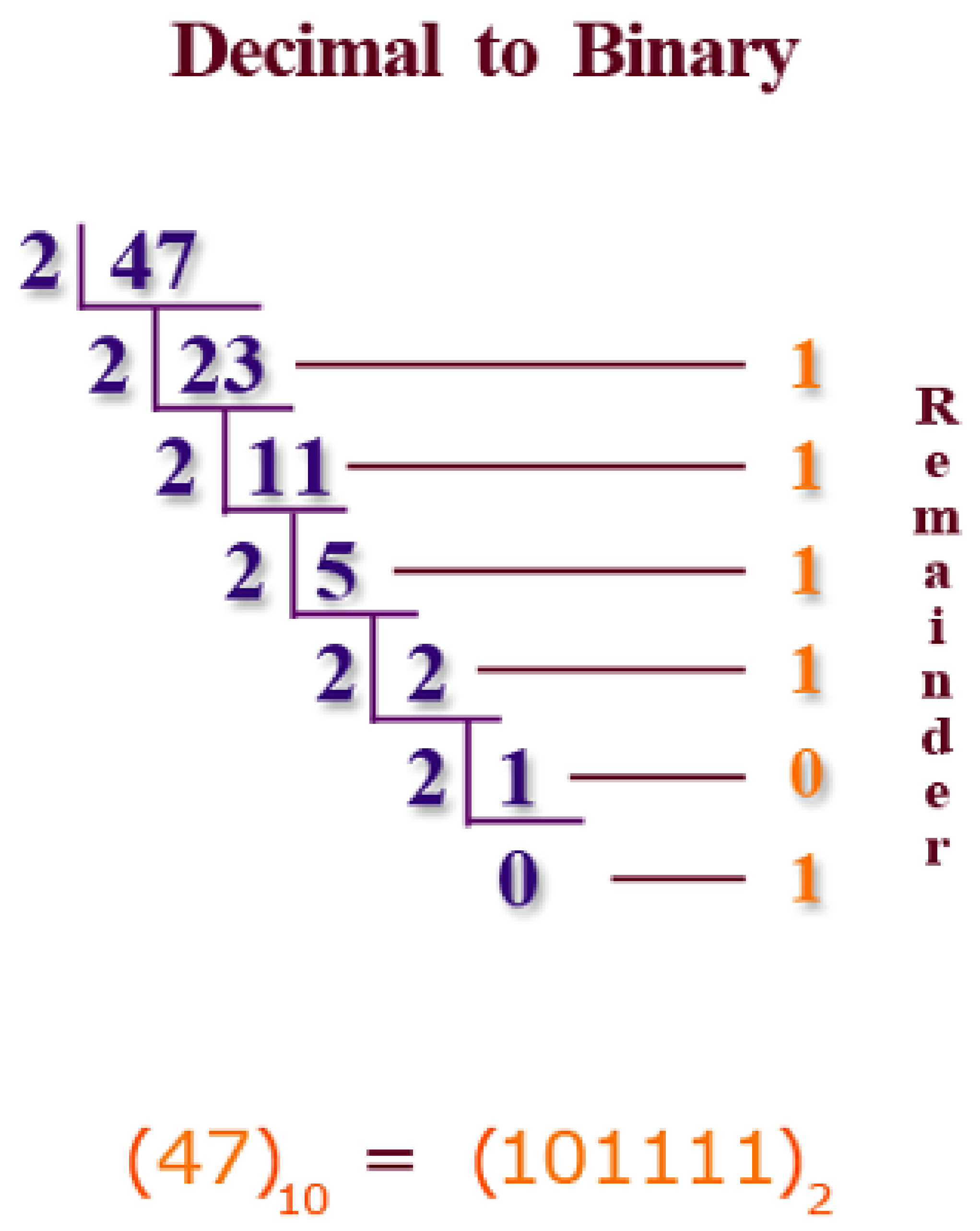
B). 3+5+7+......N

C). Cos(1)+Cos(2)+Cos(3)+......Con(N)

5. ចូរសរសេរកម្មវិធីអោយគេបំលែងពី Decimal ទៅ Binary Number?



```
bin=bin+dec%2*pow(10,i);
dec=dec/2;
i++;
```



FUNCTION

Function : is បណ្ដុំនៃ code។ គេប្រើ **Function** ដើម្បីបម្លែង Code ទៅជា Block-Block ហើយវាមានផលប្រយោជន៍ដូចខាងក្រោម:

- ងាយស្រួលប្រើប្រាស់ និង ហៅកូដដែលយើងបានបង្កើតយកមកប្រើបានច្រើនដង
- ងាយស្រួលស្វែងរក Error Code ព្រោះវា មានលក្ខណៈ Block
- បើមាន បញ្ហា Error code យើងគ្រាន់តែកែ Function មេមួយប៉ុណ្ណោះ

នៅក្នុង Python គេបែងចែក Function ជា 2 ធំៗគឺ

-> <u>None Return Function</u>

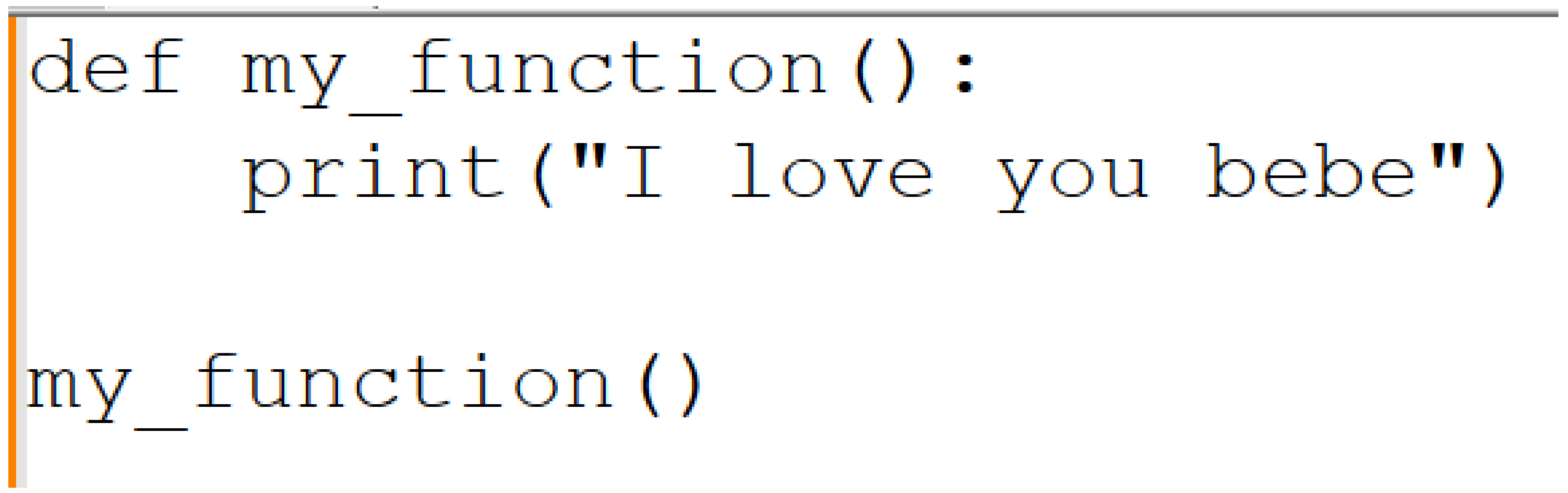
- None Return Function No Parameter
- None Return Function with Parameter

-> <u>Return Function</u>

- Return Function No Parameter
- Return Function with Parameter

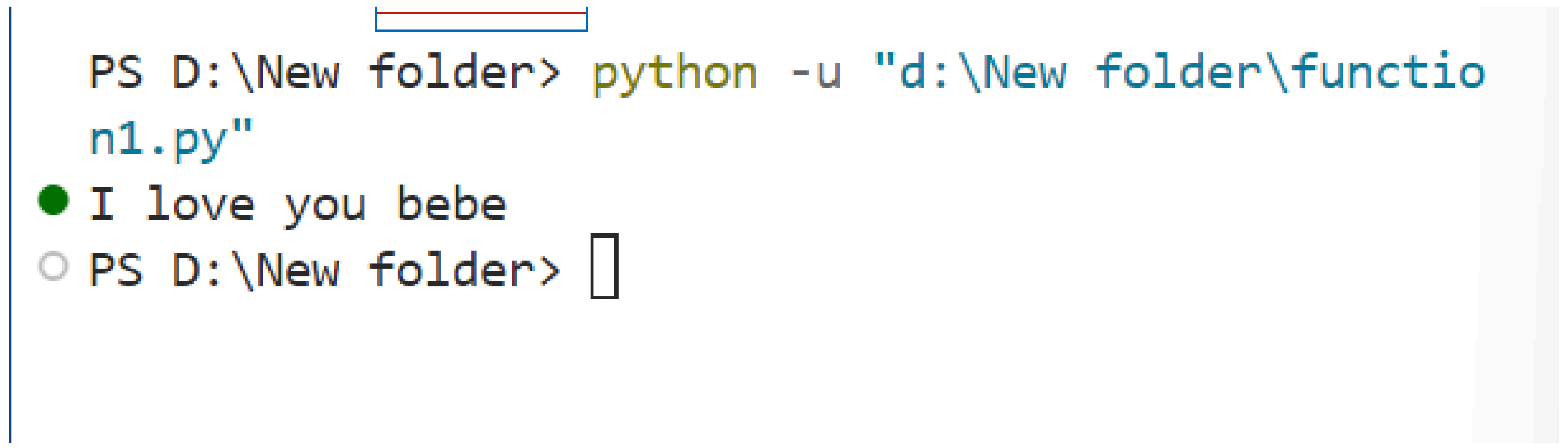
None Return Function No Parameter

code:



```
def my_function():
    print("I love you bebe")

my_function()
```

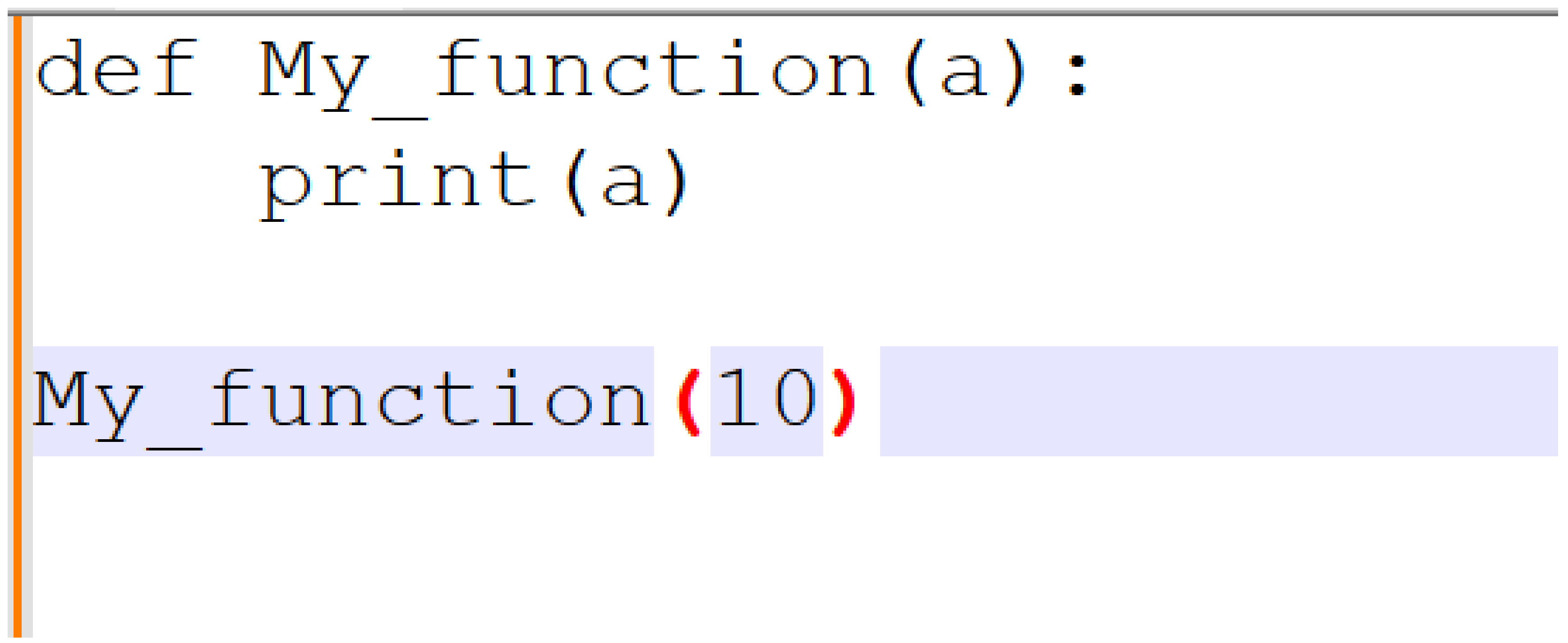
Result:



```
  PS D:\New folder> python -u "d:\New folder\functio
  n1.py"
● I love you bebe
○ PS D:\New folder> 
```

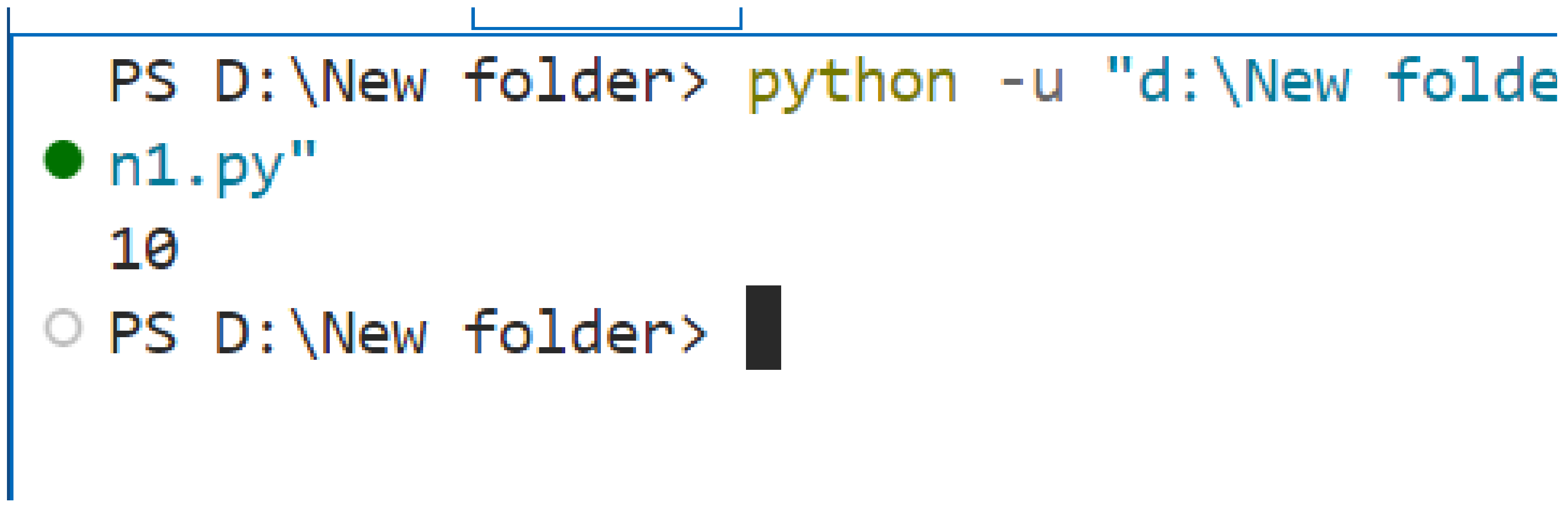
None Return Function With Parameter

code:



```
def My_function(a):
    print(a)

My_function(10)
```

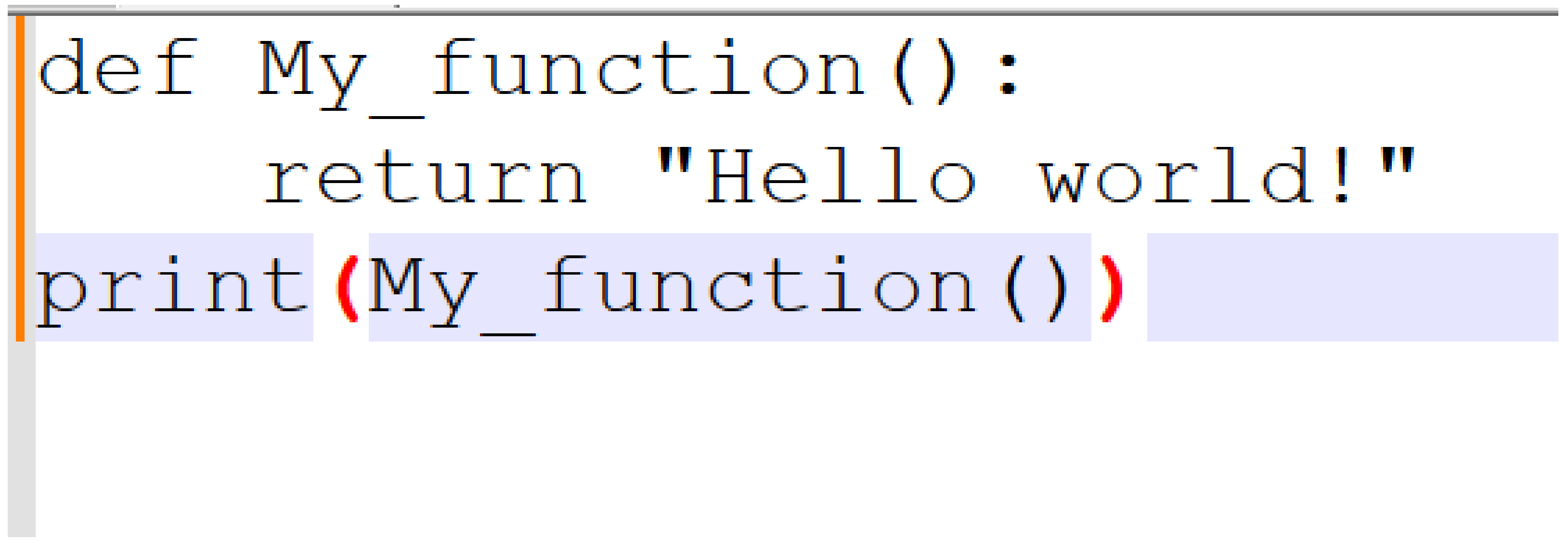
Result:



```
  PS D:\New folder> python -u "d:\New folde
● n1.py"
  10
○ PS D:\New folder> 
```

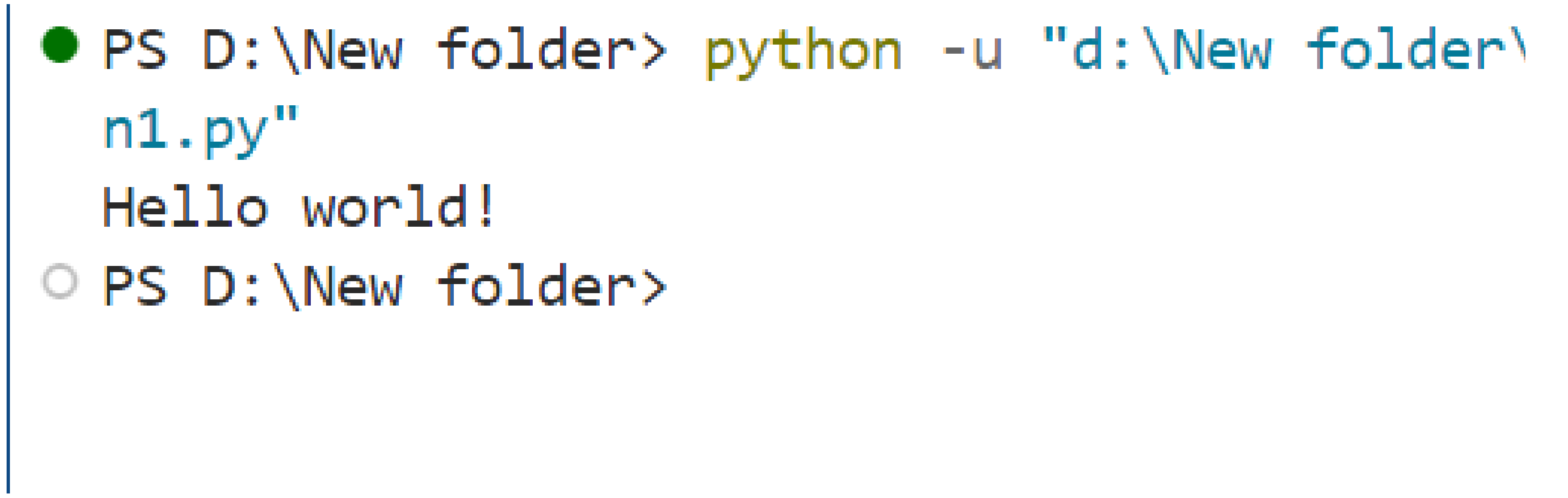
Return Function No Parameter

code:



```
def My_function():
    return "Hello world!"
print(My_function())
```

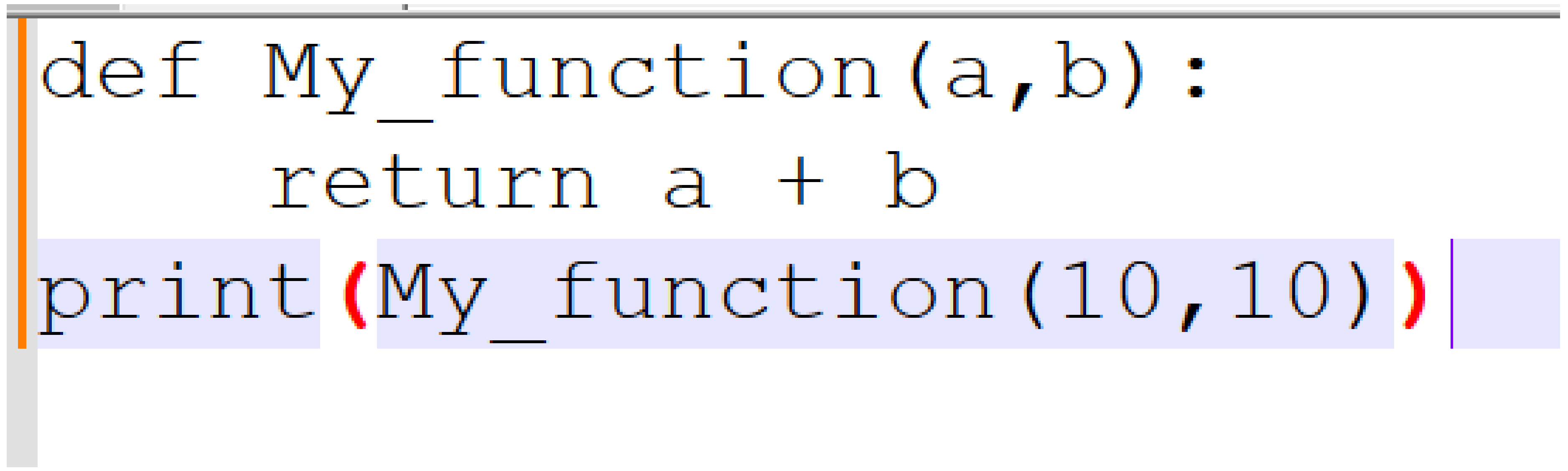
Result:



```
● PS D:\New folder> python -u "d:\New folder\
  n1.py"
  Hello world!
○ PS D:\New folder>
```

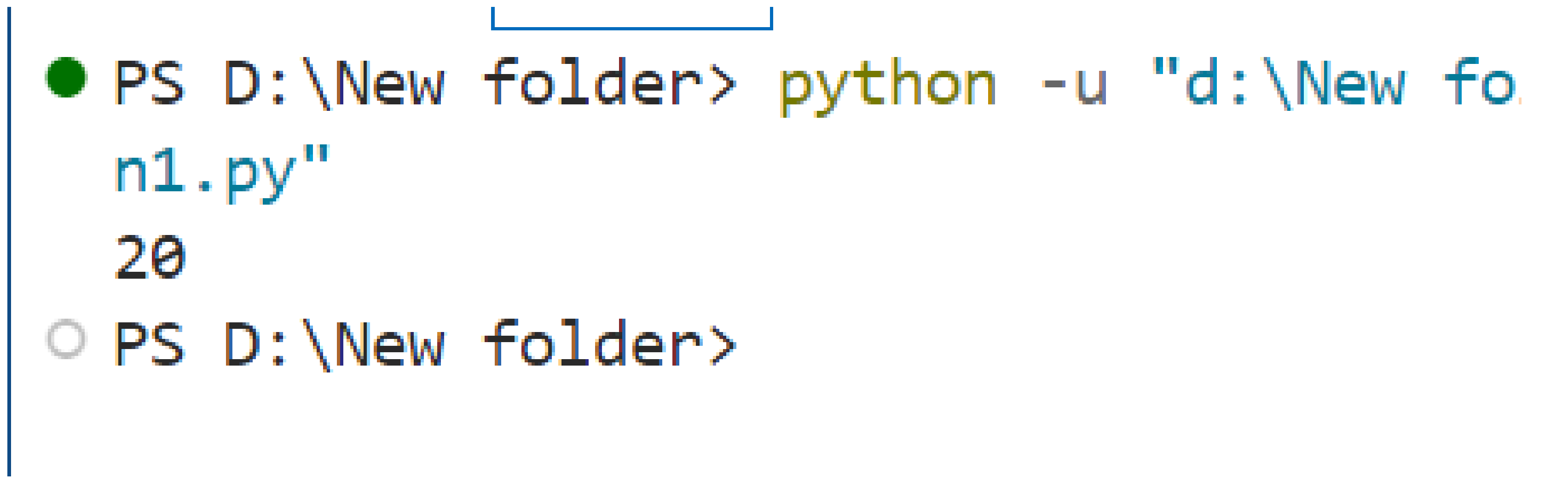
Return Function With Parameter

code:



```
def My_function(a,b):
    return a + b
print(My_function(10,10))
```

Result:

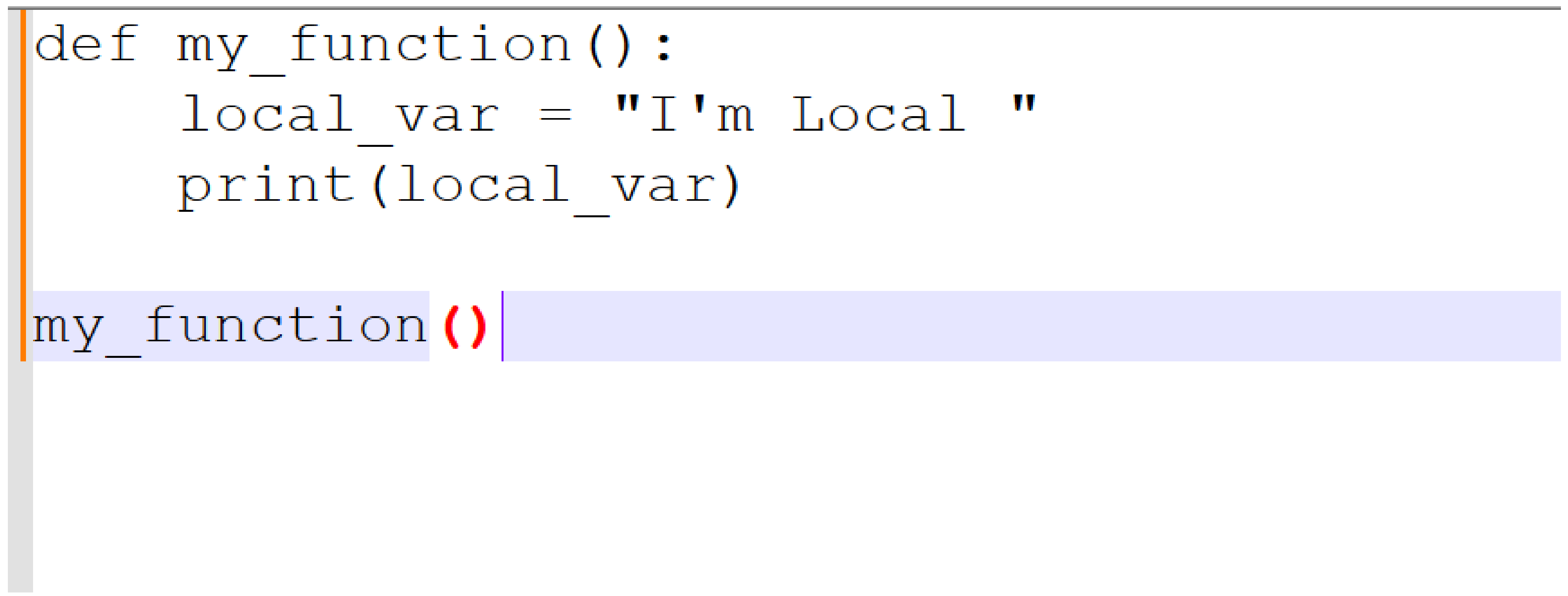


```
● PS D:\New folder> python -u "d:\New fo
  n1.py"
  20
○ PS D:\New folder>
```

ក្នុង Function មាន Variable មួយឈ្មោះថា Variable Scope ។ Variable Scope បែងចែកជា3ប្រភេទ គឺ

Local scope : វាអាចធ្វើការបានតែនៅក្នុង Function តែប៉ុណ្ណោះ

Example:

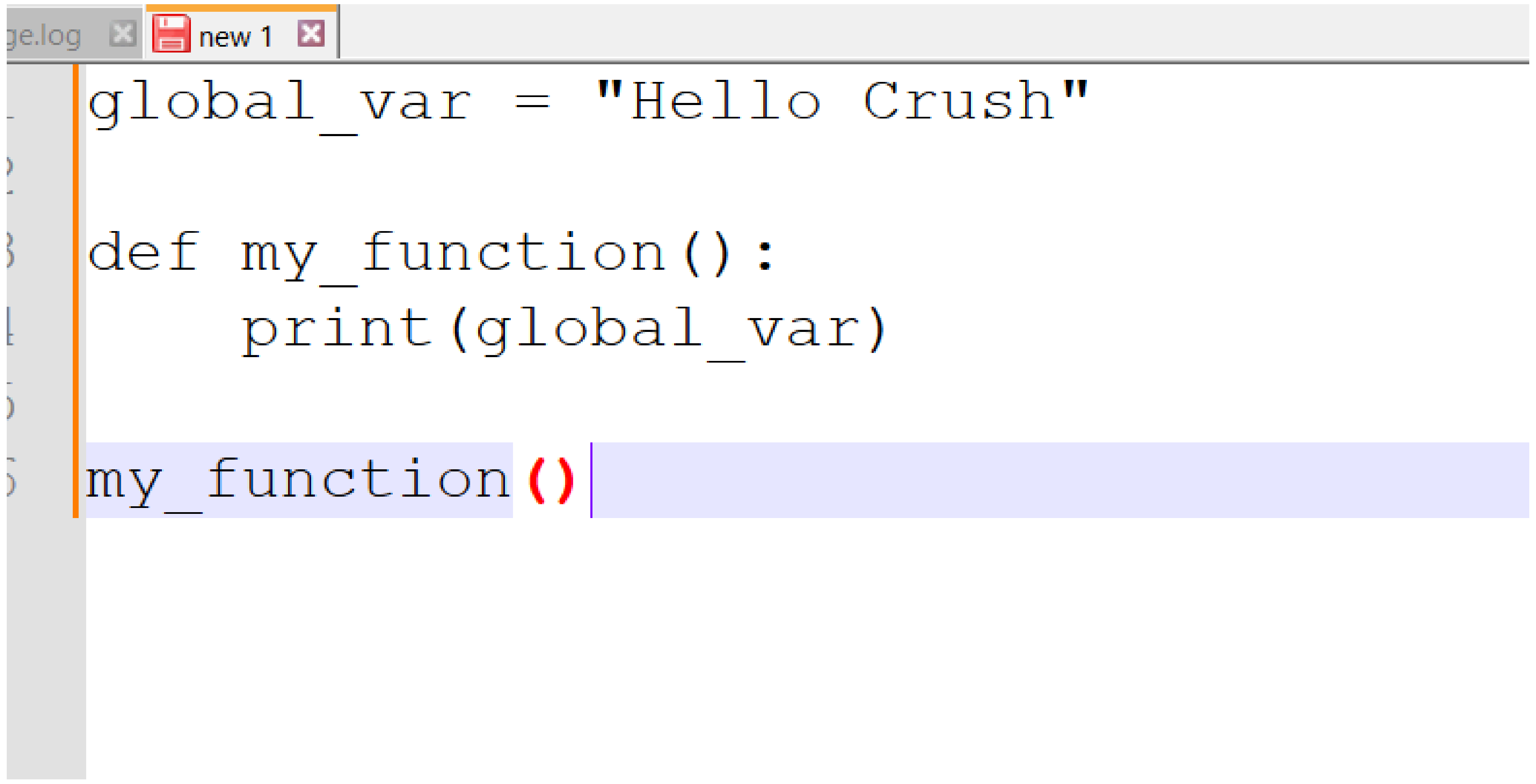


```
def my_function():
    local_var = "I'm Local "
    print(local_var)

my_function()
```

Global Scope

បើយើងមានទិន្នន័យ នៅលើ Function យើងអាចទាញយកវាមកប្រើតាមរយៈ variable Global Scope។

Example



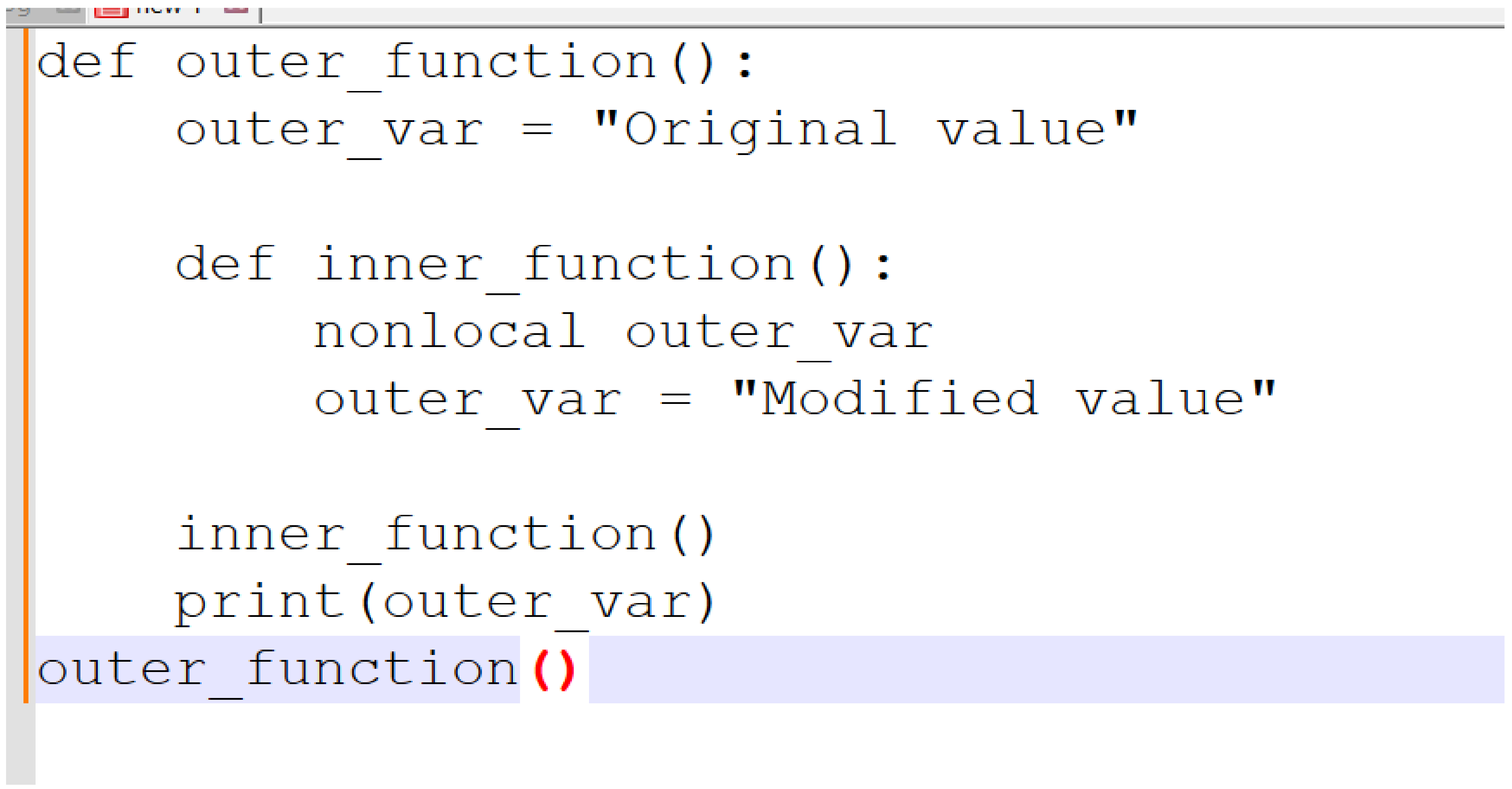
ge.log new 1

```
global_var = "Hello Crush"

def my_function():
    print(global_var)

my_function()
```

Enclosing Scope (Nonlocal)

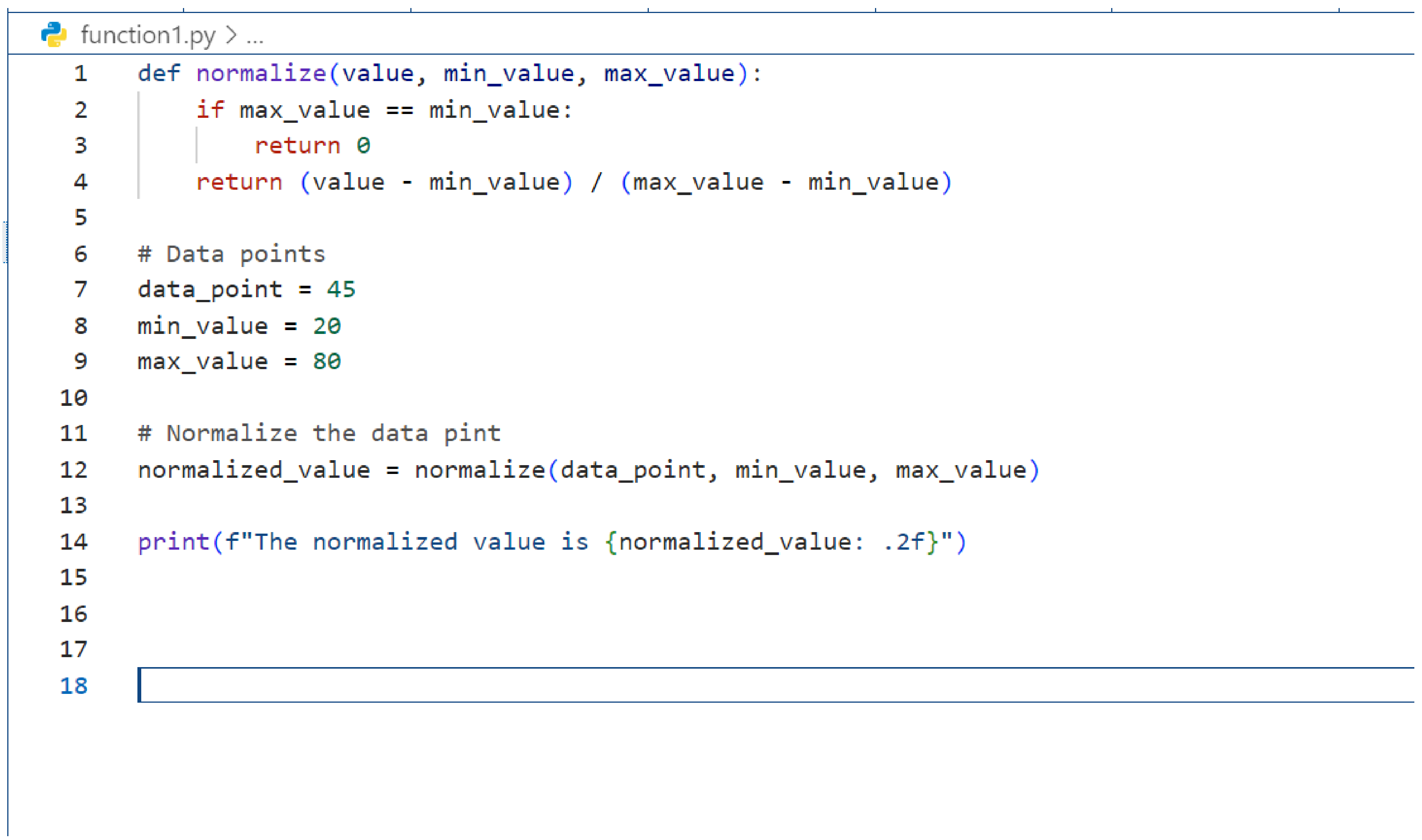
ជាមួយ Nonlocal យើងអាចប្រើដើម្បីកែតម្លៃ local variable បានដោយមិនបាច់ដូរឈ្មោះ

Example



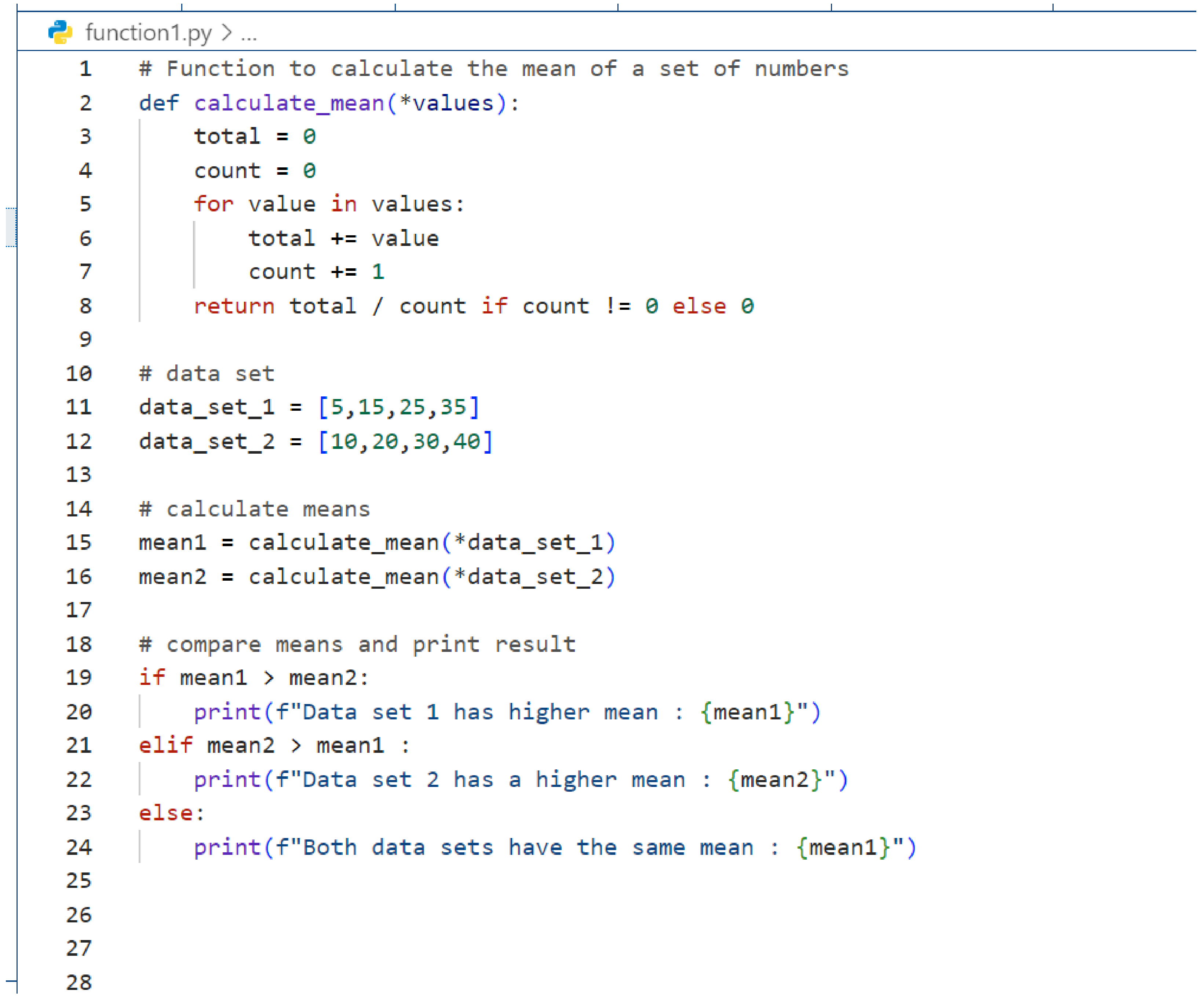
```
def outer_function():
    outer_var = "Original value"

    def inner_function():
        nonlocal outer_var
        outer_var = "Modified value"

    inner_function()
    print(outer_var)
outer_function()
```

EXAMPLE 01



function1.py > ...

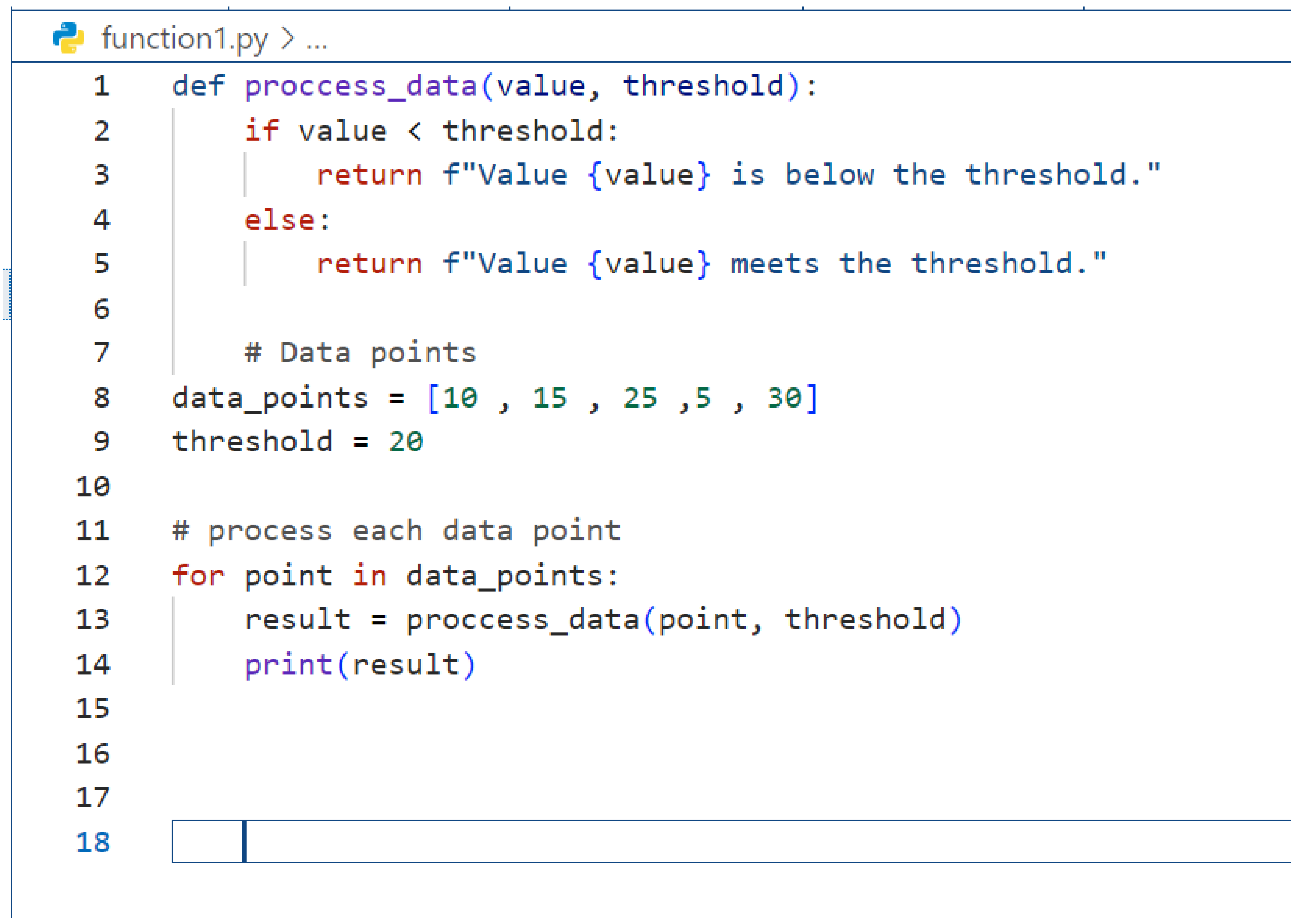
```python
def normalize(value, min_value, max_value):
    if max_value == min_value:
        return 0
    return (value - min_value) / (max_value - min_value)

# Data points
data_point = 45
min_value = 20
max_value = 80

# Normalize the data pint
normalized_value = normalize(data_point, min_value, max_value)

print(f"The normalized value is {normalized_value: .2f}")



```



function1.py > ...

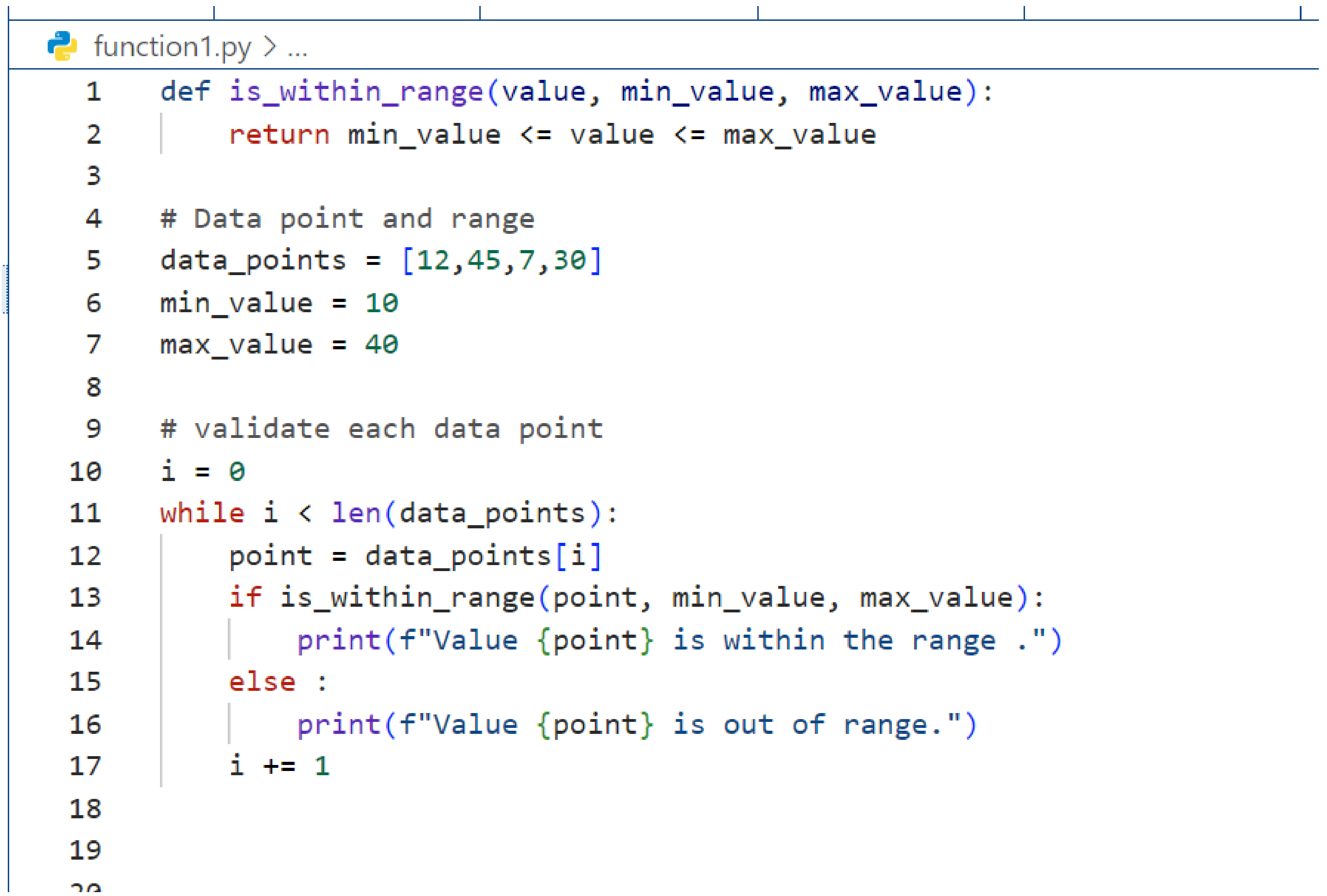
```python
# Function to calculate the mean of a set of numbers
def calculate_mean(*values):
    total = 0
    count = 0
    for value in values:
        total += value
        count += 1
    return total / count if count != 0 else 0

# data set
data_set_1 = [5,15,25,35]
data_set_2 = [10,20,30,40]

# calculate means
mean1 = calculate_mean(*data_set_1)
mean2 = calculate_mean(*data_set_2)

# compare means and print result
if mean1 > mean2:
    print(f"Data set 1 has higher mean : {mean1}")
elif mean2 > mean1 :
    print(f"Data set 2 has a higher mean : {mean2}")
else:
    print(f"Both data sets have the same mean : {mean1}")



```

EXAMPLE 02



function1.py > ...

```python
def proccess_data(value, threshold):
    if value < threshold:
        return f"Value {value} is below the threshold."
    else:
        return f"Value {value} meets the threshold."

    # Data points
data_points = [10 , 15 , 25 ,5 , 30]
threshold = 20

# process each data point
for point in data_points:
    result = proccess_data(point, threshold)
    print(result)



```



function1.py > ...

```python
def is_within_range(value, min_value, max_value):
    return min_value <= value <= max_value

# Data point and range
data_points = [12,45,7,30]
min_value = 10
max_value = 40

# validate each data point
i = 0
while i < len(data_points):
    point = data_points[i]
    if is_within_range(point, min_value, max_value):
        print(f"Value {point} is within the range .")
    else :
        print(f"Value {point} is out of range.")
    i += 1


```

PRACTICE EXERCISE

1. ចូរសចូរបង្កើត function សំរាប់ការប្រើប្រាស់នូវ Loop ដូចខាងក្រោម៖
   a. 1+2+3+....+N
   b. 2+4+6+....+N
   c. 3+5+7+.....+N

***អ្នកអាចជ្រើសប្រភេទ Loop ទំាង៣ ដូចជា for loop, while loop & do while

2. ចូរបង្កើត Function ដែលអាច អោយគេបញ្ចូល សីតុណ្ហភាពគិតជា °C រួចធ្វើការដូរ ខ្នាតនៃ សីតុណ្ហភាព ពី °C ទៅជា °F ។ { 1 °F= (°C * 9/5) + 32 }

3. ចូរបង្កើត Function ដែលអាច អោយគេបញ្ចូល សីតុណ្ហភាពគិតជា °F រួចធ្វើការដូរ ខ្នាតនៃ សីតុណ្ហភាព ពី °F ទៅជា °C ។ { 1 °C= (°F - 32) * 5/9 }

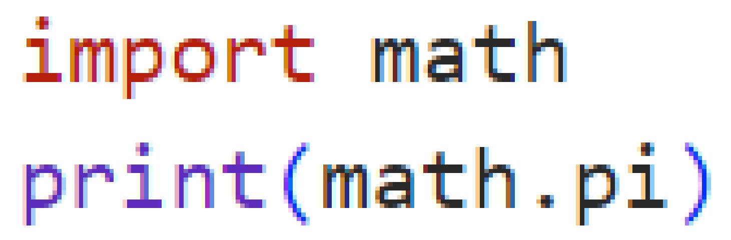
4. ចូរសរសេ Function ដែលអាចអោយគេ ធ្វើការ Vote បាន រវាង A និង B ហើយ កូដនេះដំណើរការជាមួយ loop រហូតទាល់តែគេចុច Q ដើម្បីបញ្ចប់ និង បូកសរុប លទ្ធផល ។

MODULE

Module គឺជា File ដែលត្រូវបានគេសសេរកូត python សម្រាប់ការងារណាមួយជាក់លាក់រួចរាល់ ហើយ យើងគ្រាន់តែហៅ File នោះមកប្រើគឺ អាចប្រើប្រាស់ Function ក្នុងនោះបានហើយ។

Module មាន ២ប្រភេទ៖
1.Built-in Modules
2.User-define Modules

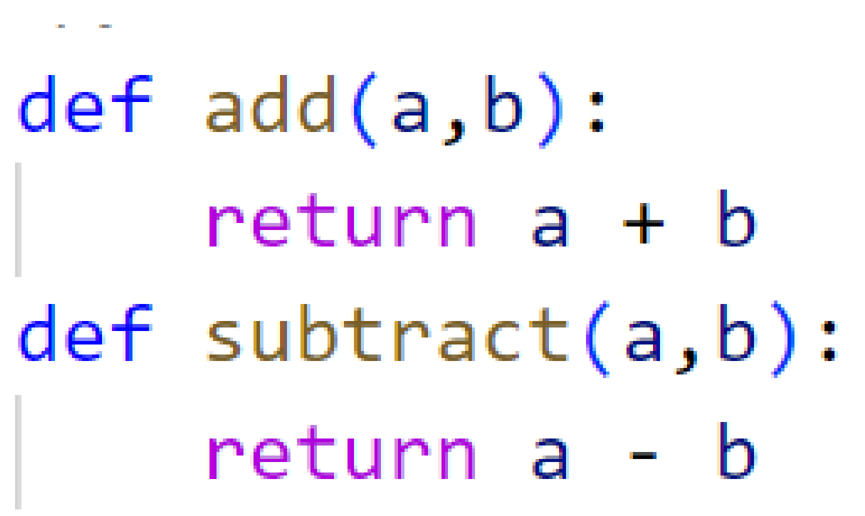
របៀបនៃការប្រើប្រាស់គឺយើងអាច import ឈ្មោះ File យកមកប្រើតែម្តងសម្រាប់ពពួក Build-in សម្រាប់ User-define ដំបូងត្រូវ Create file បន្ទាប់មក import File ដើម្បីប្រើ



```
import math
print(math.pi)
```

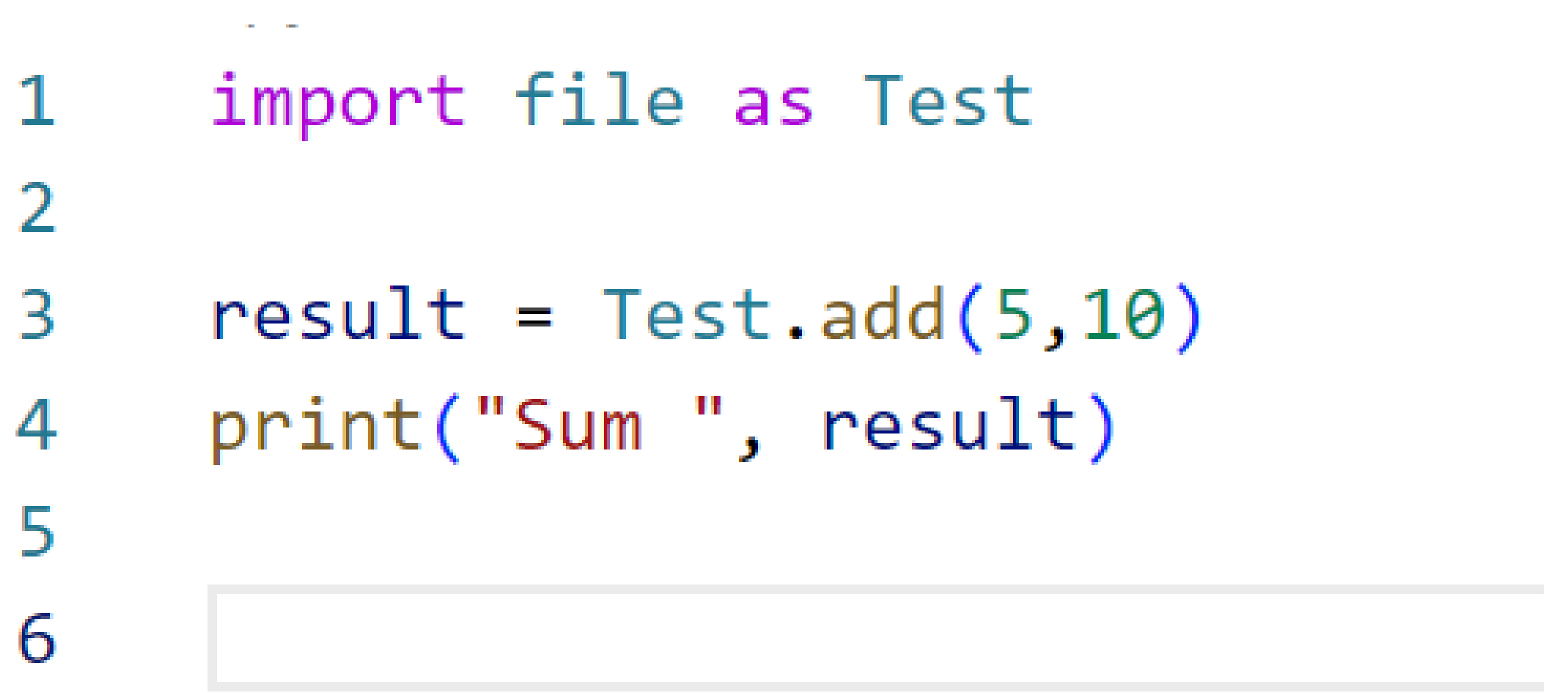
EXAMPLE 01

create file name “file”

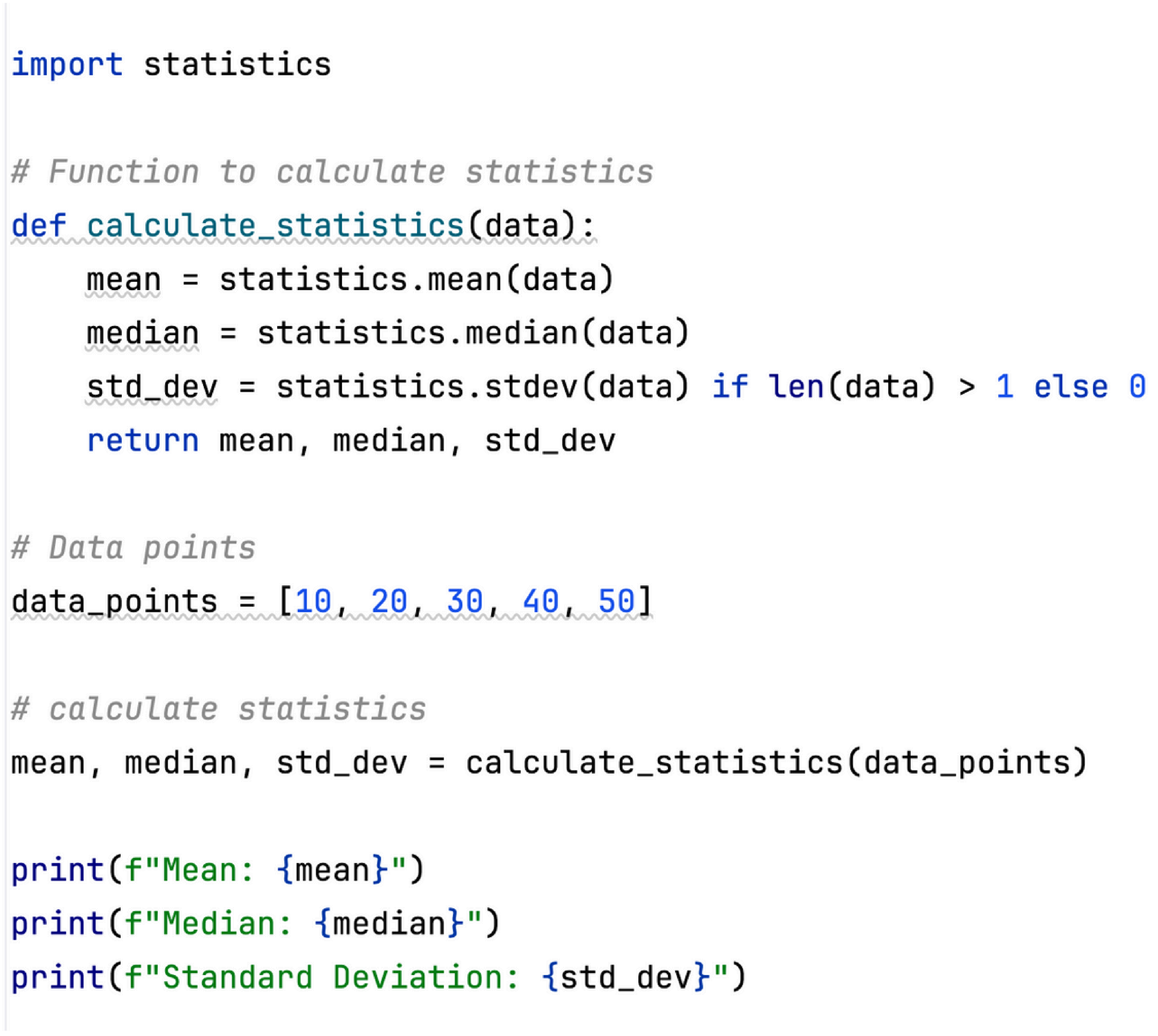


```python
def add(a,b):
    return a + b
def subtract(a,b):
    return a - b
```

create new file .import “file”

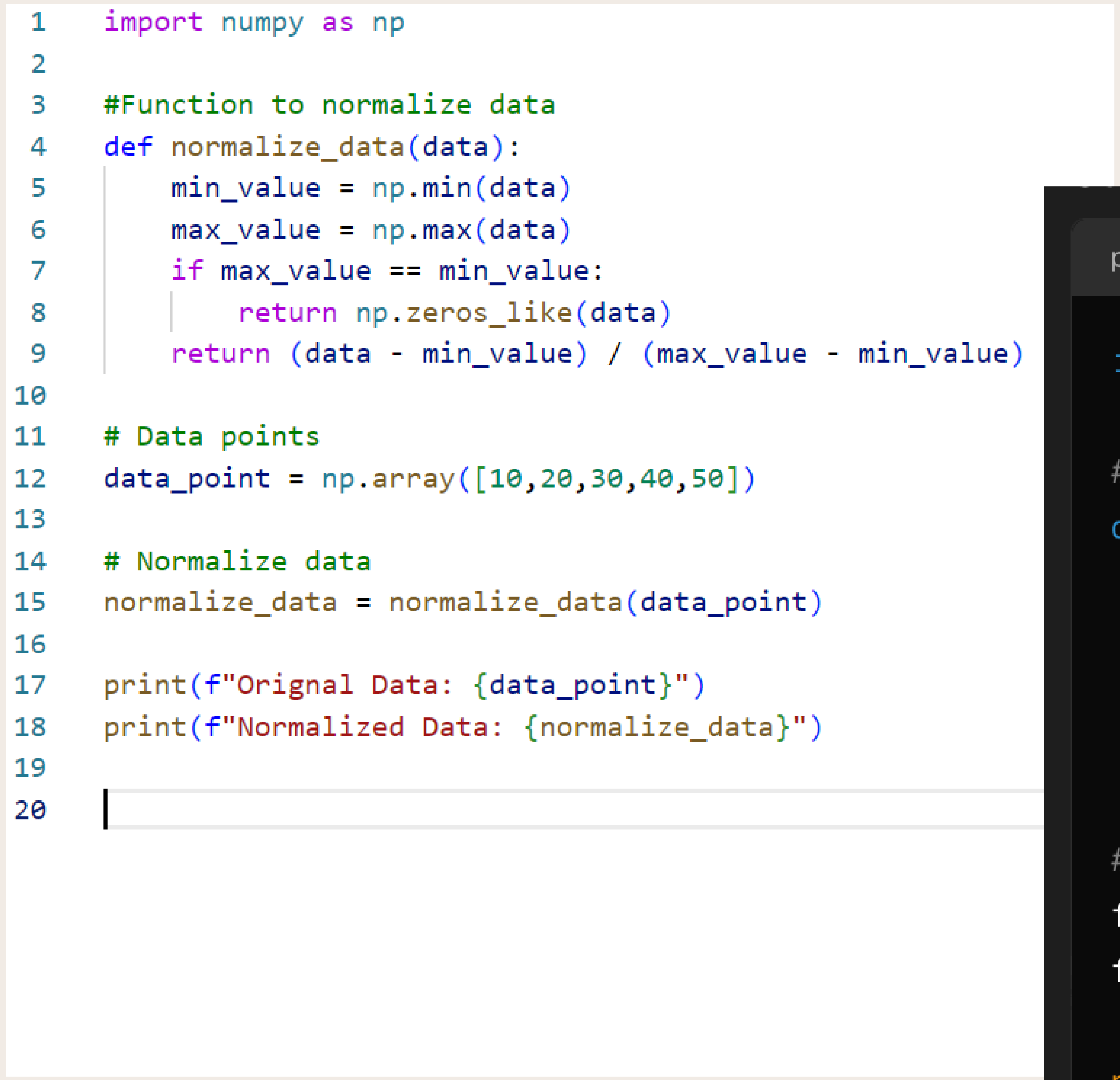


```python
import file as Test

result = Test.add(5,10)
print("Sum ", result)

```

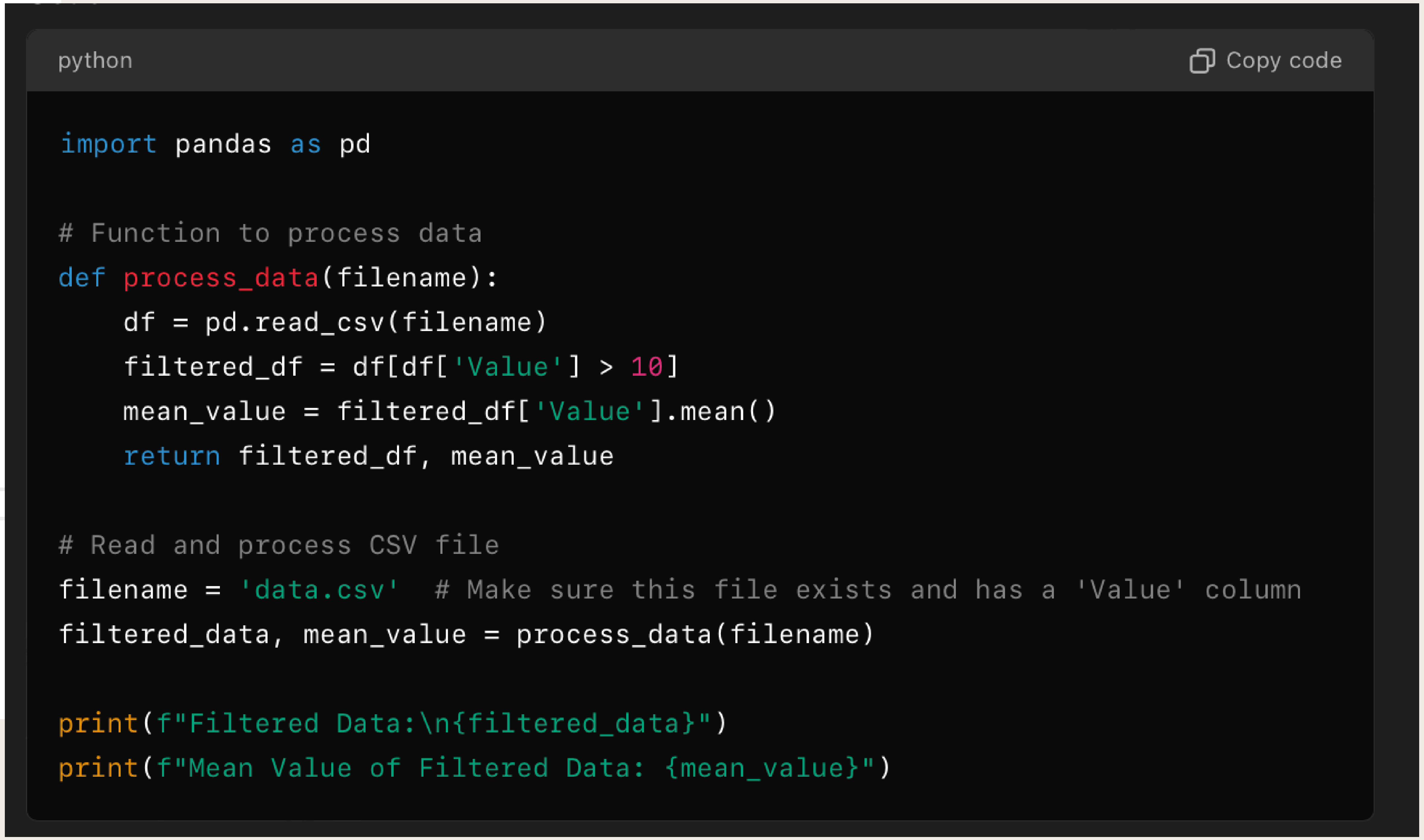


```python
import statistics

# Function to calculate statistics
def calculate_statistics(data):
    mean = statistics.mean(data)
    median = statistics.median(data)
    std_dev = statistics.stdev(data) if len(data) > 1 else 0
    return mean, median, std_dev

# Data points
data_points = [10, 20, 30, 40, 50]

# calculate statistics
mean, median, std_dev = calculate_statistics(data_points)

print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Standard Deviation: {std_dev}")
```

EXAMPLE 02



```python
import numpy as np

#Function to normalize data
def normalize_data(data):
    min_value = np.min(data)
    max_value = np.max(data)
    if max_value == min_value:
        return np.zeros_like(data)
    return (data - min_value) / (max_value - min_value)

# Data points
data_point = np.array([10,20,30,40,50])

# Normalize data
normalize_data = normalize_data(data_point)

print(f"Orignal Data: {data_point}")
print(f"Normalized Data: {normalize_data}")

```



```python
import pandas as pd

# Function to process data
def process_data(filename):
    df = pd.read_csv(filename)
    filtered_df = df[df['Value'] > 10]
    mean_value = filtered_df['Value'].mean()
    return filtered_df, mean_value

# Read and process CSV file
filename = 'data.csv'  # Make sure this file exists and has a 'Value' column
filtered_data, mean_value = process_data(filename)

print(f"Filtered Data:\n{filtered_data}")
print(f"Mean Value of Filtered Data: {mean_value}")
```

EXCEPTION HANDLING

Syntax Errors vs. Exceptions:

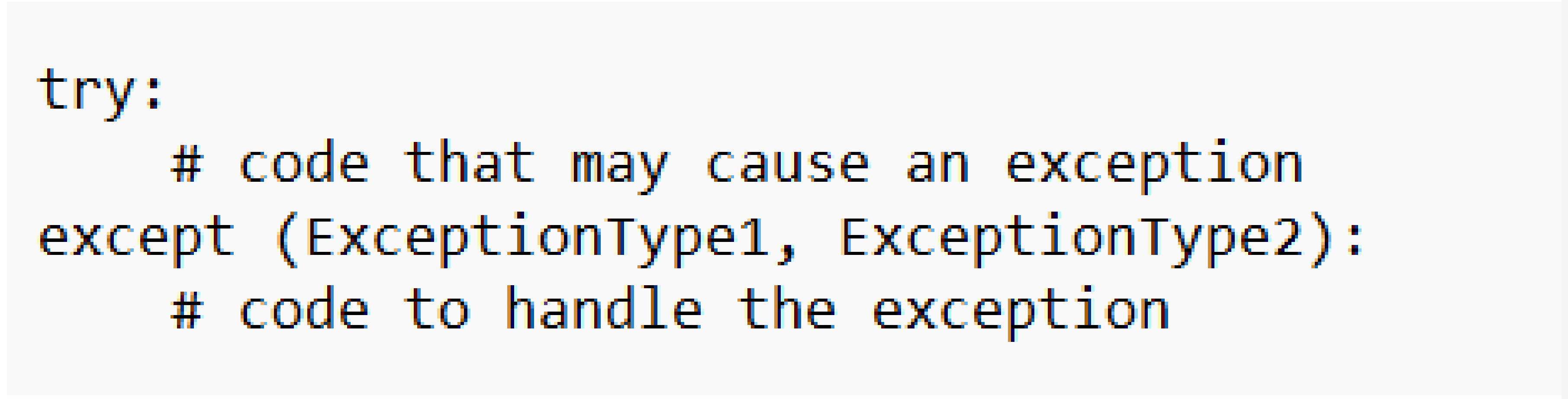
- Syntax Errors: គឺជាការ Error នៃការសសេរកូត

(e.g., missing colon or parentheses).

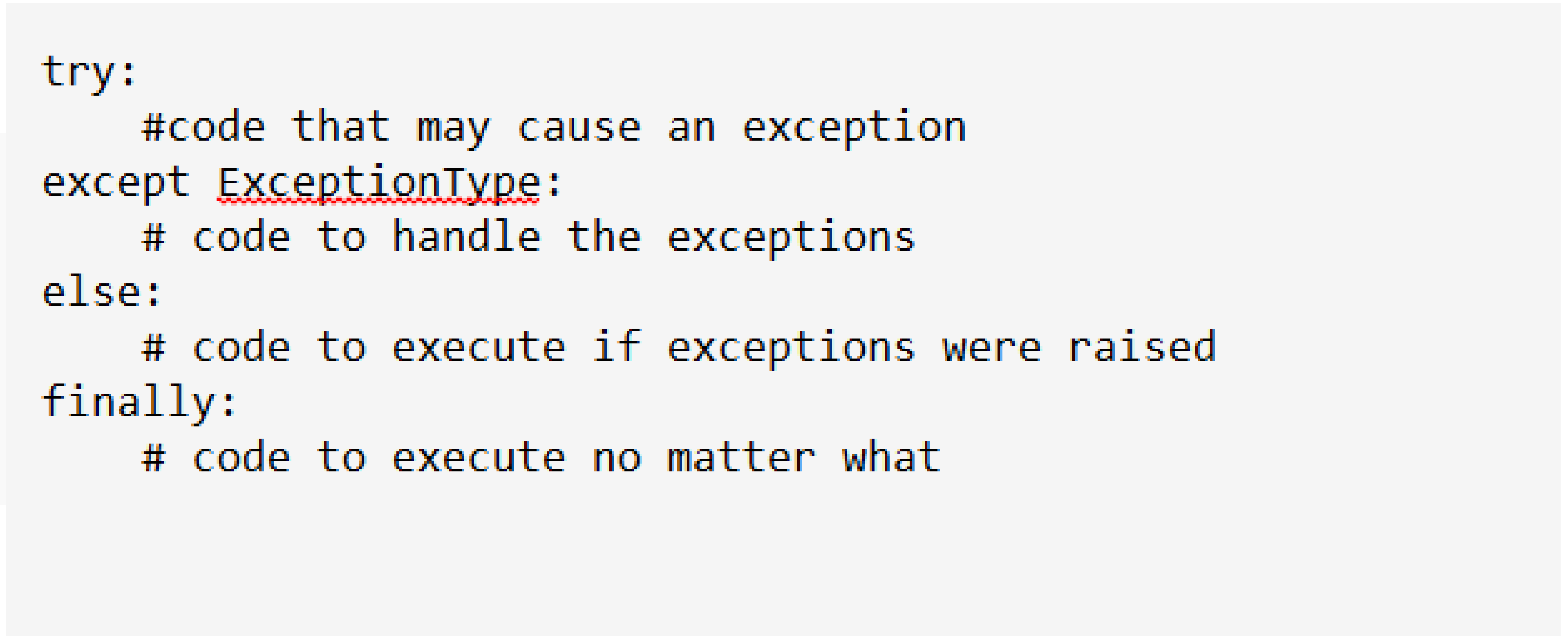
- Exceptions: គឺជាការ Error នៃការគិត concept កូត

(e.g., division by zero, accessing a file that doesn't exist).

Exception Handling: គឺជាការ ចាប់យក Exception ដើម្បីកុំអោយ program របស់យើង crash

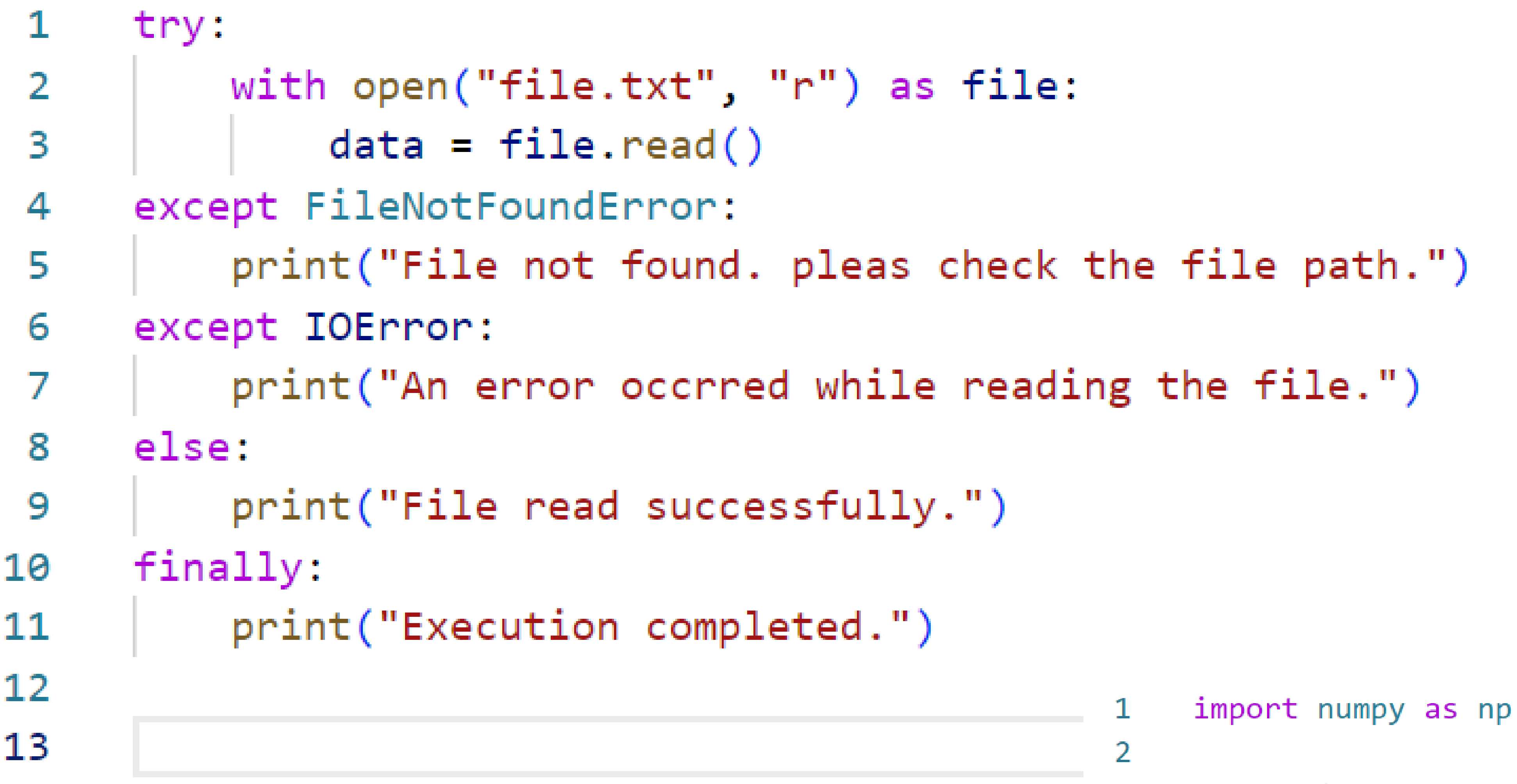


```
try:
    # code that may cause an exception
except (ExceptionType1, ExceptionType2):
    # code to handle the exception
```

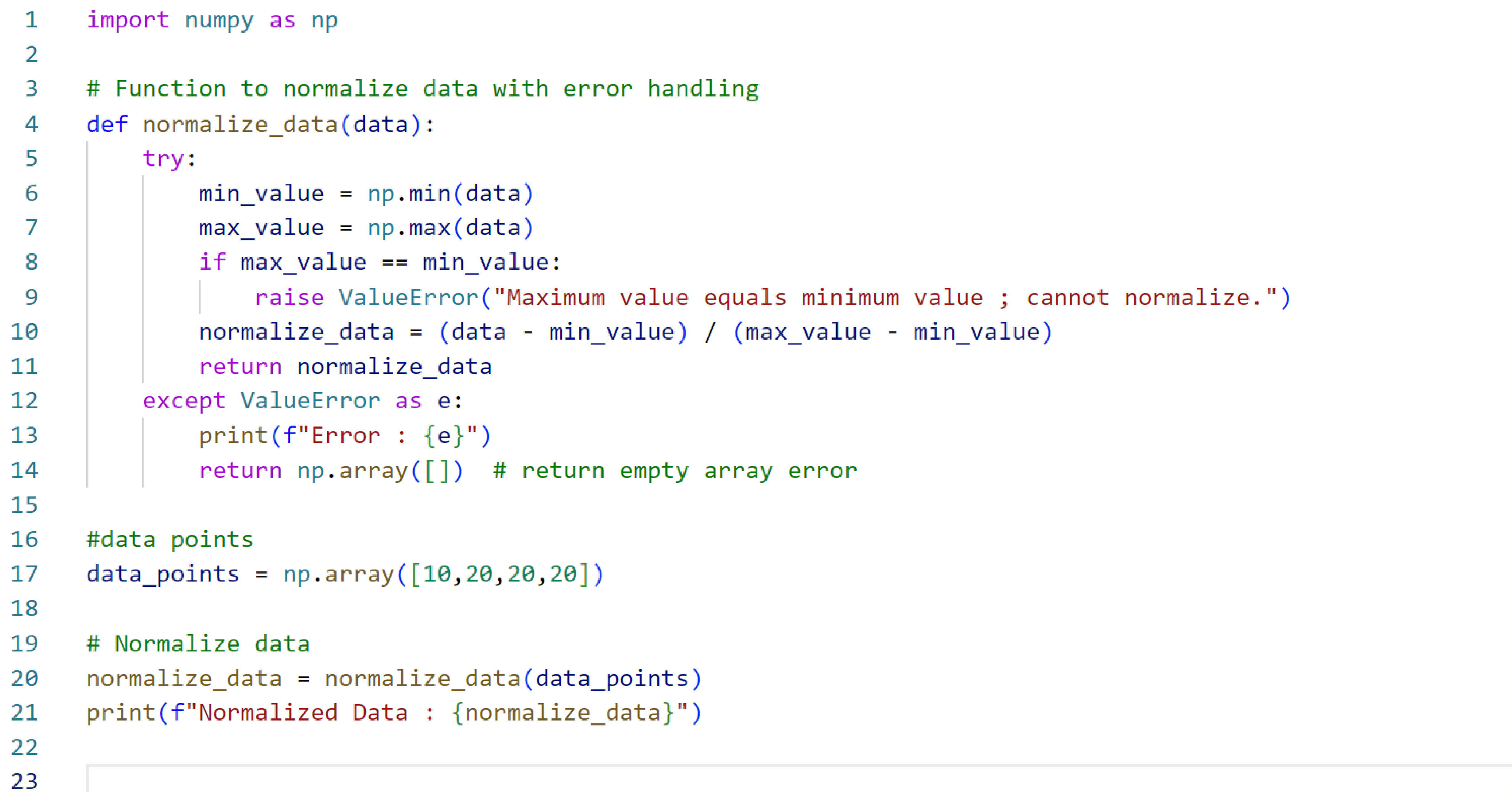


```
try:
    #code that may cause an exception
except ExceptionType:
    # code to handle the exceptions
else:
    # code to execute if exceptions were raised
finally:
    # code to execute no matter what
```

EXAMPLE 01

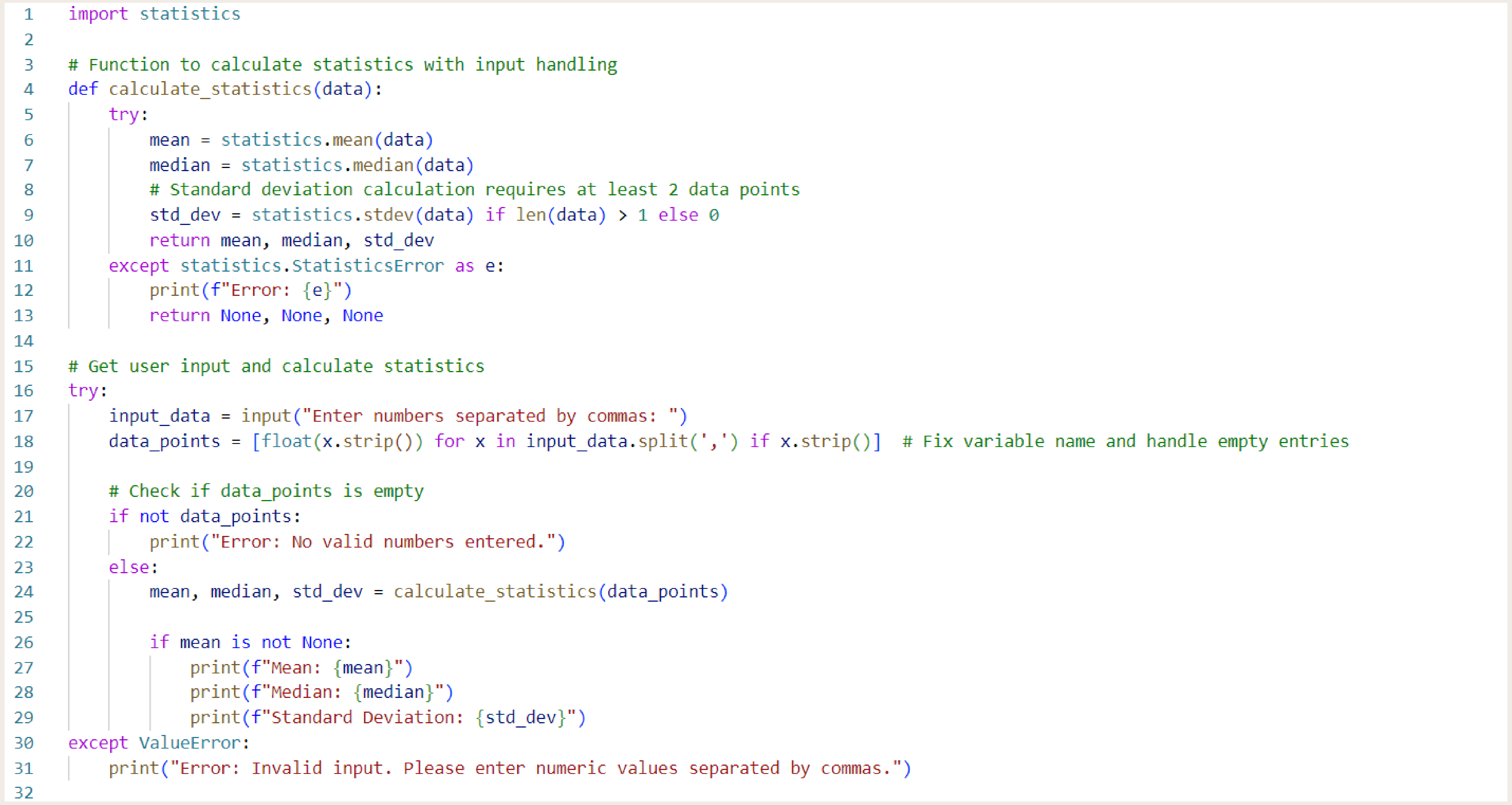


```python
try:
    with open("file.txt", "r") as file:
        data = file.read()
except FileNotFoundError:
    print("File not found. pleas check the file path.")
except IOError:
    print("An error occrred while reading the file.")
else:
    print("File read successfully.")
finally:
    print("Execution completed.")
```



```python
import numpy as np

# Function to normalize data with error handling
def normalize_data(data):
    try:
        min_value = np.min(data)
        max_value = np.max(data)
        if max_value == min_value:
            raise ValueError("Maximum value equals minimum value ; cannot normalize.")
        normalize_data = (data - min_value) / (max_value - min_value)
        return normalize_data
    except ValueError as e:
        print(f"Error : {e}")
        return np.array([])  # return empty array error

#data points
data_points = np.array([10,20,20,20])

# Normalize data
normalize_data = normalize_data(data_points)
print(f"Normalized Data : {normalize_data}")
```

EXAMPLE 02



```python
import statistics

# Function to calculate statistics with input handling
def calculate_statistics(data):
    try:
        mean = statistics.mean(data)
        median = statistics.median(data)
        # Standard deviation calculation requires at least 2 data points
        std_dev = statistics.stdev(data) if len(data) > 1 else 0
        return mean, median, std_dev
    except statistics.StatisticsError as e:
        print(f"Error: {e}")
        return None, None, None

# Get user input and calculate statistics
try:
    input_data = input("Enter numbers separated by commas: ")
    data_points = [float(x.strip()) for x in input_data.split(',') if x.strip()]  # Fix variable name and handle empty entries

    # Check if data_points is empty
    if not data_points:
        print("Error: No valid numbers entered.")
    else:
        mean, median, std_dev = calculate_statistics(data_points)

        if mean is not None:
            print(f"Mean: {mean}")
            print(f"Median: {median}")
            print(f"Standard Deviation: {std_dev}")
except ValueError:
    print("Error: Invalid input. Please enter numeric values separated by commas.")

```

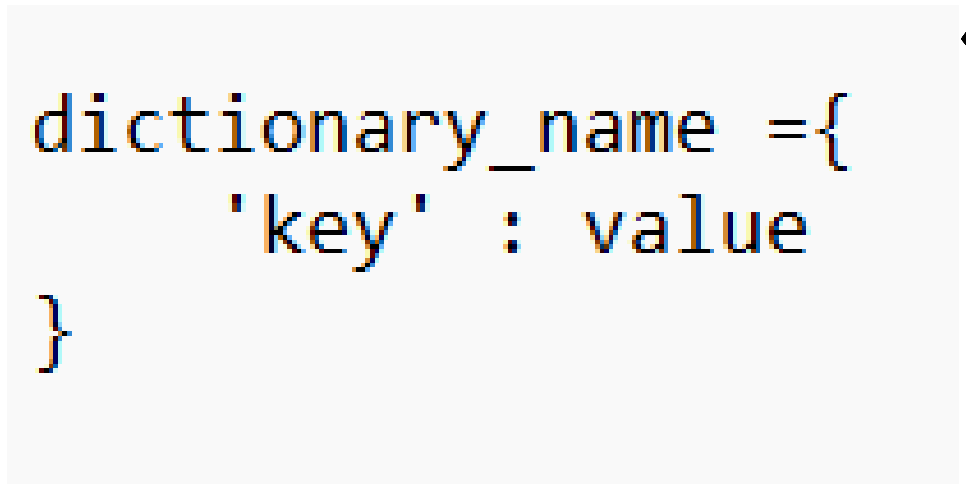
END OF BASIC

Dictionary in python

នៅក្នុង Python Dictionary គឺជាបណ្តុំនៃគូ key-value ដែល key នីមួយៗ គឺជាកូនសោដើម្បីចូលទៅប្រើប្រាស់តម្លៃរបស់វា។

របៀបបង្កើត Dictionry in python

Syntax1:

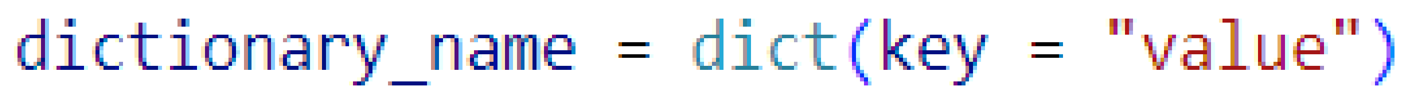


```
dictionary_name ={
    'key' : value
}
```

យើងអាចបង្កើត dictionary ដោយប្រើដង្កៀបអង្កាញ់(curly braces) {} ជាមួយ key និង value

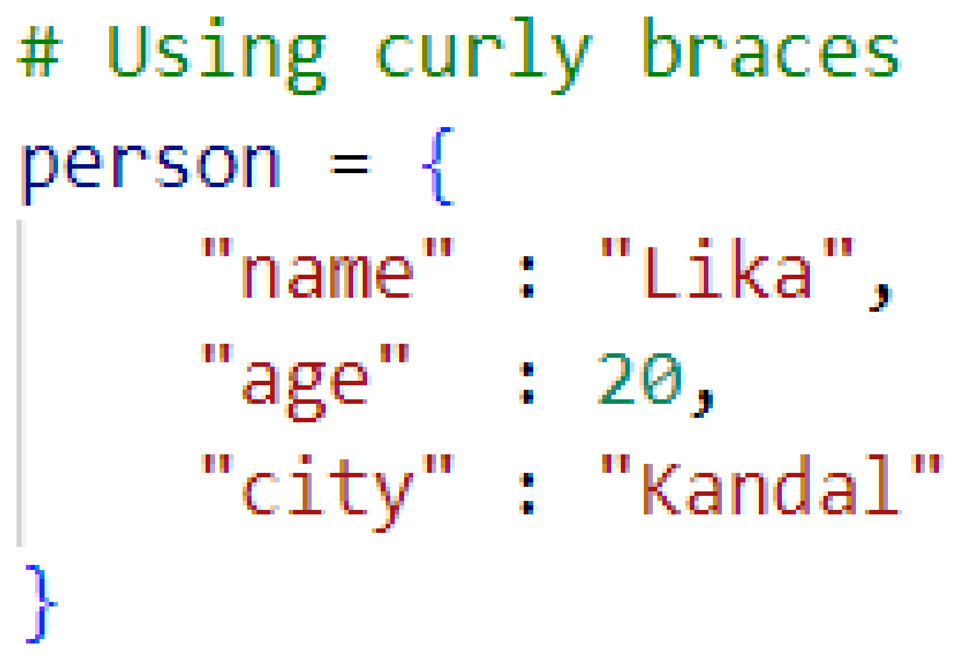
Syntax2:

បង្កើត dictionary ដោយប្រើ function dict()

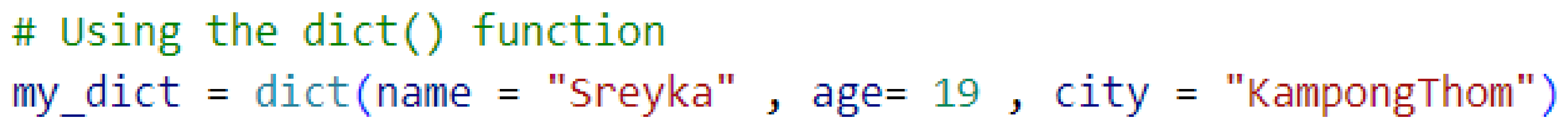


```
dictionary_name = dict(key = "value")
```

Example create dictionary

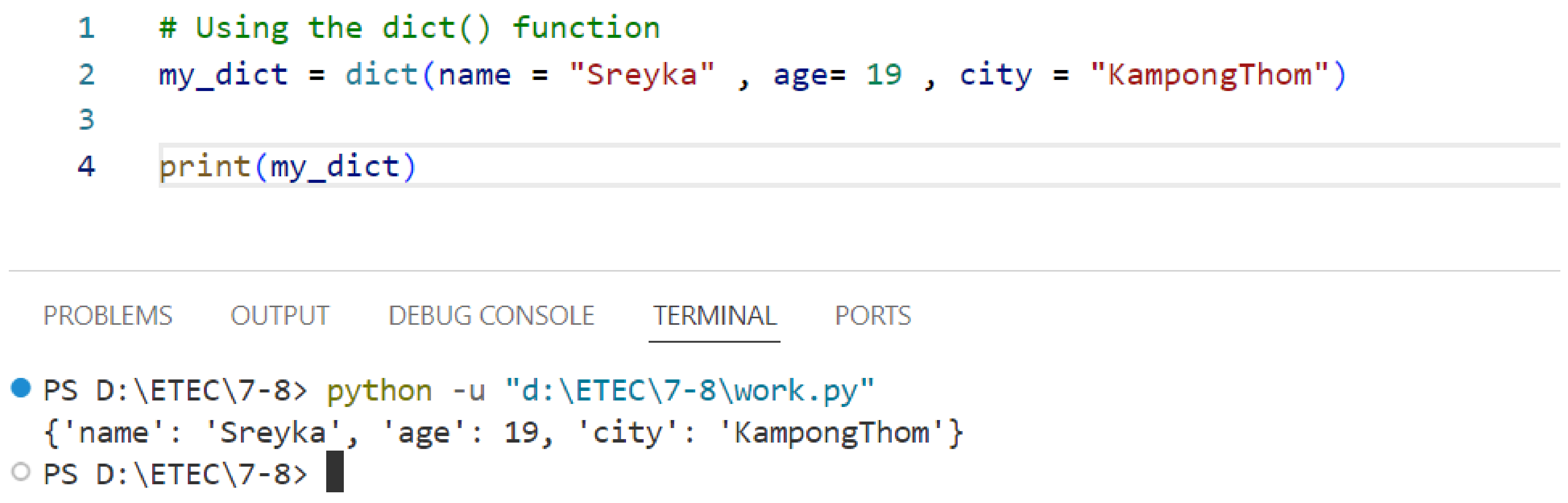


```python
# Using curly braces
person = {
     "name" : "Lika",
     "age"  : 20,
     "city" : "Kandal"
}
```



```python
# Using the dict() function
my_dict = dict(name = "Sreyka" , age= 19 , city = "KampongThom")
```

បង្ហាញទិន្ន័យទាំងអស់ដែលមានក្នុង Dictionary

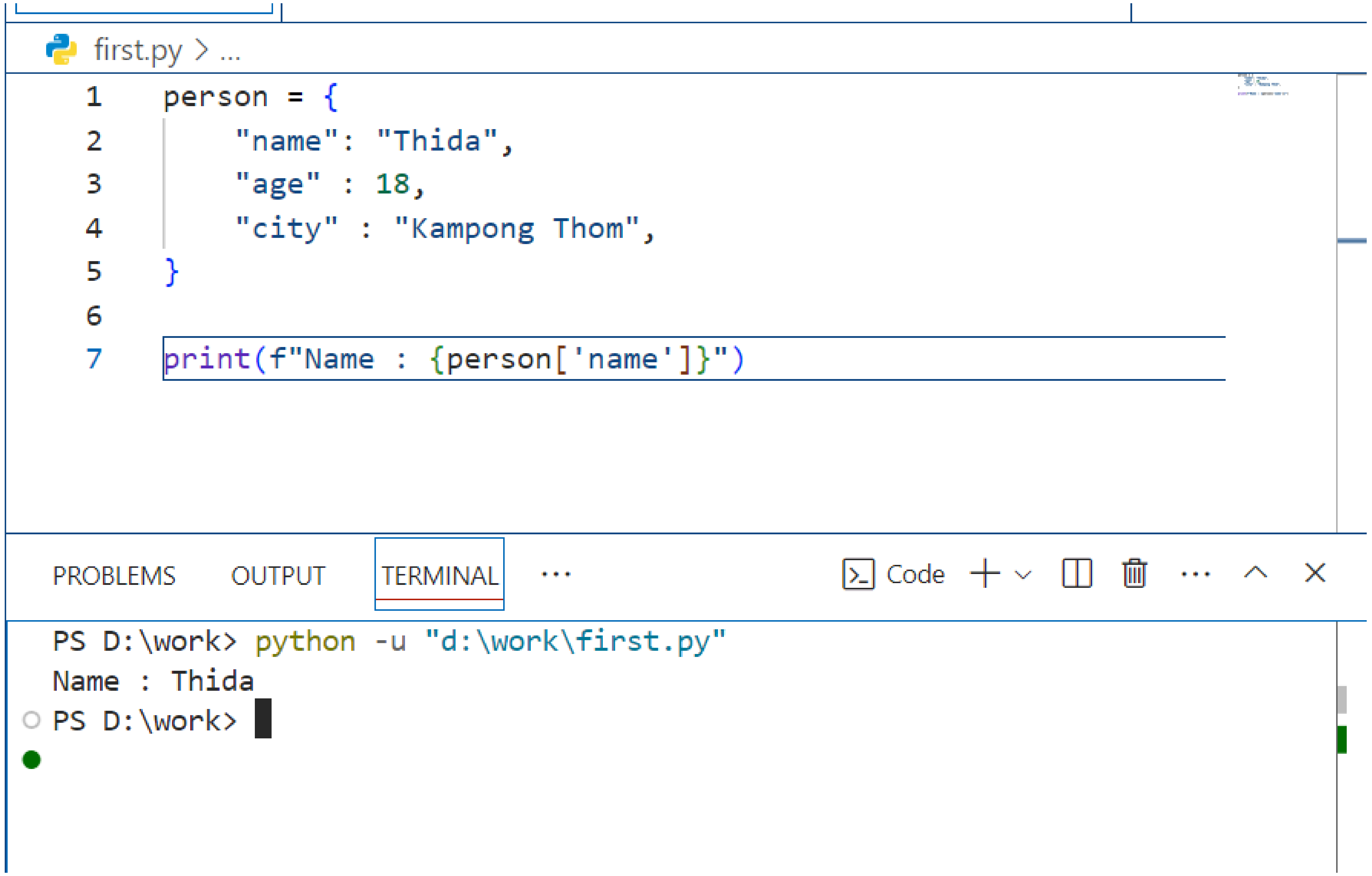


```python
# Using the dict() function
my_dict = dict(name = "Sreyka" , age= 19 , city = "KampongThom")

print(my_dict)
```

```
PS D:\ETEC\7-8> python -u "d:\ETEC\7-8\work.py"
{'name': 'Sreyka', 'age': 19, 'city': 'KampongThom'}
PS D:\ETEC\7-8>
```

Accessing and Modify

ការចូលប្រើតម្លៃ៖ តម្លៃរបស់ dictionary អាចត្រូវបានចូលប្រើដោយគ្រាន់មានឈ្មោះរបស់ dictionary ផ្ទាប់ជាមួយ សញ្ញា [] និង Key របស់វានៅខាងក្នុងសញ្ញា [’key’]


first.py > ...

```python
person = {
    "name": "Thida",
    "age" : 18,
    "city" : "Kampong Thom",
}

print(f"Name : {person['name']}")
```

PROBLEMS   OUTPUT   TERMINAL

```
PS D:\work> python -u "d:\work\first.py"
Name : Thida
PS D:\work>
```