Python Variables

Variables

Assign value to variable:

variable_name = expression

- Example:

```
[21]: x=5                  #integer number
      x="sally"            # string
      y='hello world'      #string

[22]: print(x)

      sally

[23]: print(y)

      hello world
```

- Variables are containers for storing data values.

```
[36]: x=5
      y="Python"
      z=26.87
      x=6
      y=202

[39]: print(x)

      6

[40]: x, y, z=5, "Python", 26.87

[41]: print(y)

      Python
```

Variables are case sensitive

```
[42]: print(Z)

      ---------------------------------------------------------------------------
      NameError                                 Traceback (most recent call last)
      <ipython-input-42-fb5bcf912ef3> in <module>
      ----> 1 print(Z)

      NameError: name 'Z' is not defined
```

Multiple assignment

- Python allows you to assign values to multiple variables in one line:

```
[34]: w, x, y, z = "Orange", 1, "apple", 2.25
      print('w=', w)
      print('x=', x)
      print('y=', y)
      print('z=', z)
```

```
w= Orange
x= 1
y= apple
z= 2.25
```

- And you can assign the same value to multiple variables in one line:

```
[37]: x= y = "Orange"
      print('x=', x)
      print('y=', y)
```

```
x= Orange
y= Orange
```

Variables are case sensitive

[56]:
```
age=25
Age=30
AGE=35
print("age", age)
print("Age",Age)
print("AGE", AGE)
```

```
age 25
Age 30
AGE 35
```

[57]:
```
print("AgE", AgE)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-57-b264ba5ca17f> in <module>
----> 1 print("AgE", AgE)

NameError: name 'AgE' is not defined
```

Rules of Naming Variables

- May contain letters (capital, small)

[45]:
```
age=25
Height=150
WEIGTH=75
```

- May contain underscores (_)

[46]:
```
_=150
_age=22
width_=5
```

- May contain digits [0...9]

[51]:
```
age2=30
```

- **Never** start with digit

[52]:
```
2age=25
```

```
  File "<ipython-input-52-106cb3ab9f2f>", line 1
    2age=25
       ^
SyntaxError: invalid syntax
```

[53]:
```
age2=25
```

Naming Conventions

Camel Case: Second and Subsequent words First Character is capitalized
Ex: human**W**alking**D**istance

Pascal Case: Identical to Camel Case, except the first word is also capitalized.
Ex: **H**uman**W**alking**D**istance

Snake Case: Words are separated by underscores.
Ex: human_walking_distance

In [5]:
```
# Variables - Snake Case
age = 10

employee_age = 20
```

In [7]:
```
# Functions - Snake Case
def add():
    pass

def calculate_mean_median():
    pass
```

In [8]:
```
# Class - Pascal Case
class FullTimeEmployee:
    pass
```

In [9]:
```
# Method - Snake Case
class PartTimeEmployee:
    def partial_salary(self): # Snake Case
        pass
```

In [10]:
```
# Constant - ALL UPPERCASE WITH SNAKE CASE FORMAT

PI = 3.14

CONSTANT_PI = 3.14
```

In [11]:
```
# Module - use a short , lowercase words & words separated by underscore
module.py

module_file.py
```

Arithmetic operations on variables

[60]:
```
x=2
y=3
z=x+y**2
print(z)
m=(x+y)**2
print(m)
```

```
11
25
```

Assignment Operators

Operator	Example	Same As
+=	x += 3	x = x + 3
− =	X − =3	x = x − 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

Table 2.2 Short−hand assignment operators

[62]:
```
x=3
x=x+1
print(x)
```
```
4
```

[63]:
```
x=3
x+=1
print(x)
```
```
4
```

Python Data type

Data types in python

Data types specify the type of data like numbers and characters to be stored and manipulated within a program.

- Numbers:
  - int (integer)
    1, -3, 0, 681522,.....
  - float
    0.564, 4.365454
- Boolean
  True, False
- str (string)
  "Hello", 'this is lecture 1'
- Nones
- Collection:
  - List [1, 3, 'hi', [3, 6, 8], 0.6545]
  - Tuple (23, 'hi', 0.3545, [2, 5, 9], (4, 6, 9), 256)
  - Set { 34, 'hi', 0.656, [4, 7], (5, 7), {5, 0}, 23 }
  - Dictionary { 'python': 95, "statistics": 90, "Math": 85, 601: 80 }

Numbers

1- Integer (int)

```
[66]: year=1990
      month=12
      day = 5

[67]: type(day)

[67]: int
```

2- float

```
[69]: width=2.5
      length=5.4
      type(width)

[69]: float
```

String datatype

strings can be declared using "double qoutes"

[76]:
```
firstName="Zeinab"
last_name="Ezz"
course="Python"
type(course)
```

[76]:
```
str
```

Also, strings can be declared using 'single qoute'

[78]:
```
firstName='Zeinab'
last_name='Ezz'
course='Python'
type(course)
```

[78]:
```
str
```

[79]:
```
firstName='Zeinab"
```

```
  File "<ipython-input-79-45299768bdd4>", line 1
    firstName='Zeinab"
                      ^
SyntaxError: EOL while scanning string literal
```

Multi-line String

- Multiline string can be defined using triple single quotes

[125]:
```
message='''This is a multi-line message
using three single qoutes
at the start and at the end of the string'''
print(message)
```

```
This is a multi-line message
using three single qoutes
at the start and at the end of the string
```

- Multiline string can be defined using triple double quotes

[127]:
```
message="""This is a multi-line message
using three double qoutes
at the start and at the end of the string"""
print(message)
```

```
This is a multi-line message
using three double qoutes
at the start and at the end of the string
```

Boolean

Booleans represent one of two values: True or False.

```
[2]: x=3>9

[3]: x

[3]: False

[4]: y=5==5

[5]: y

[5]: True
```

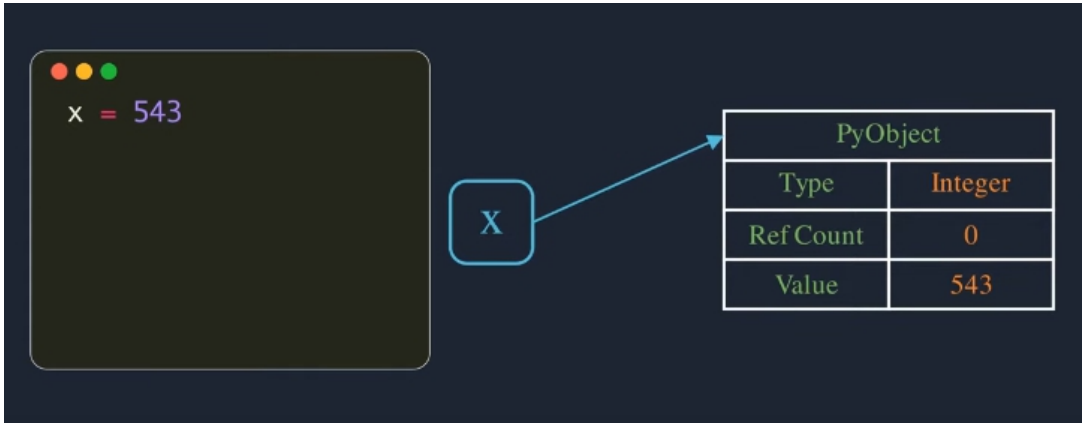
```
[73]: print(10 > 9)
      True

[74]: print(10 == 9)
      False
```

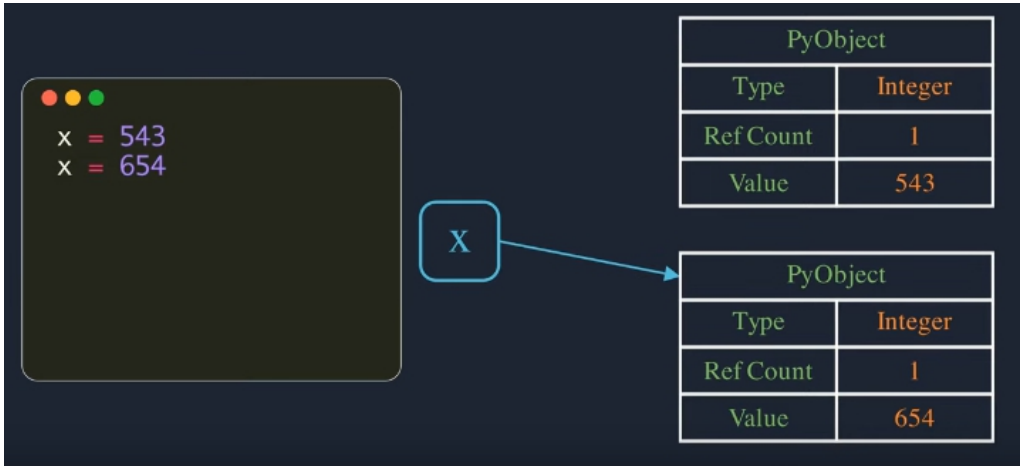
None

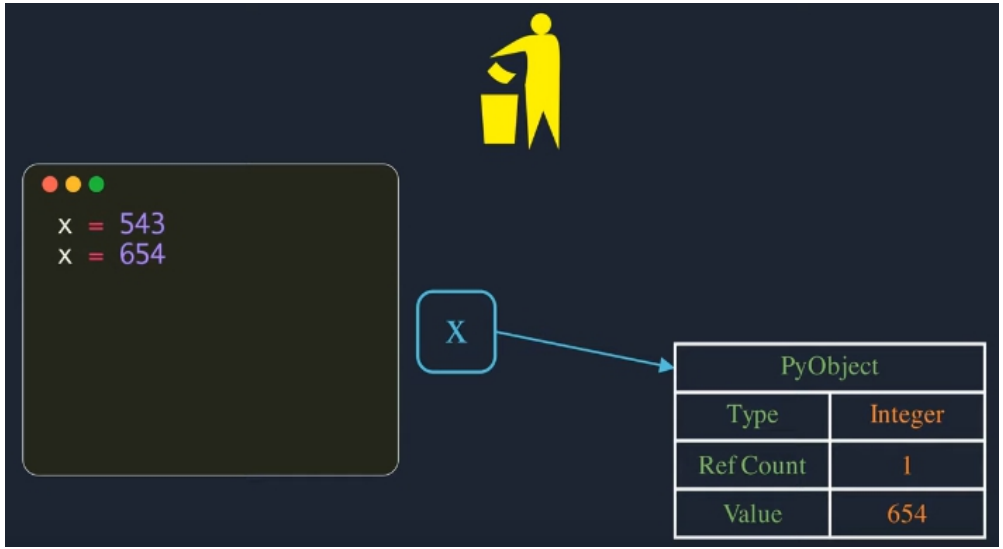
- None is another special data type in Python.
- None is frequently used to represent the absence of a value.
- For example,

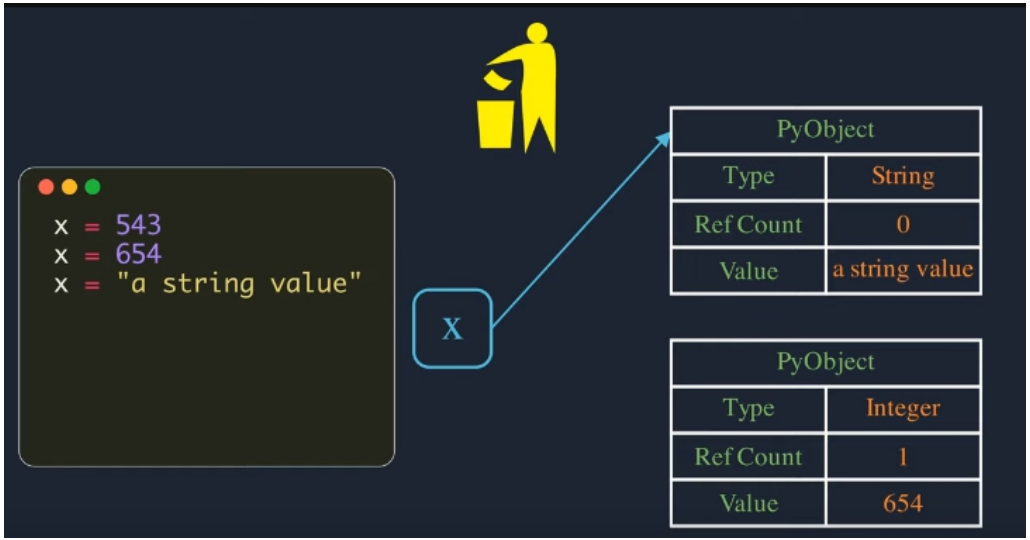
```
>>> money = None
```

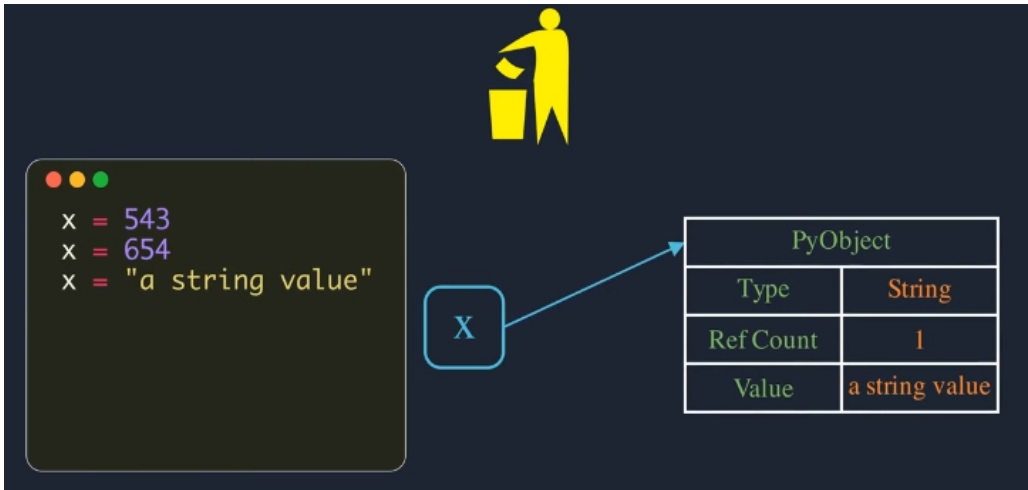
Storing Variables

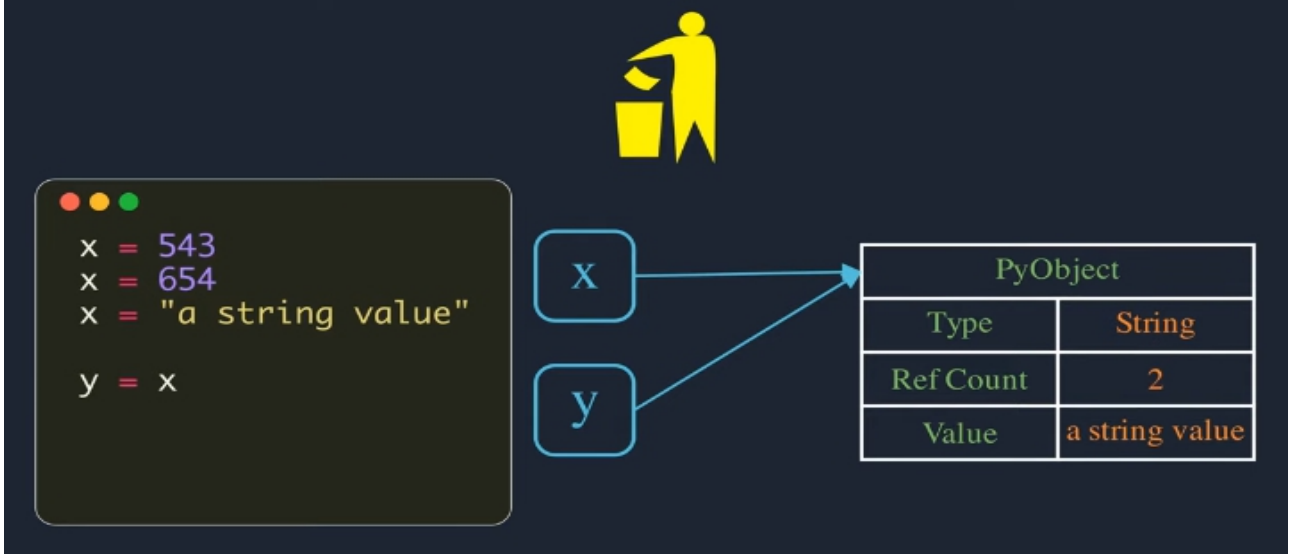

x = 543
X
PyObject
Type
Integer
Ref Count
0
Value
543

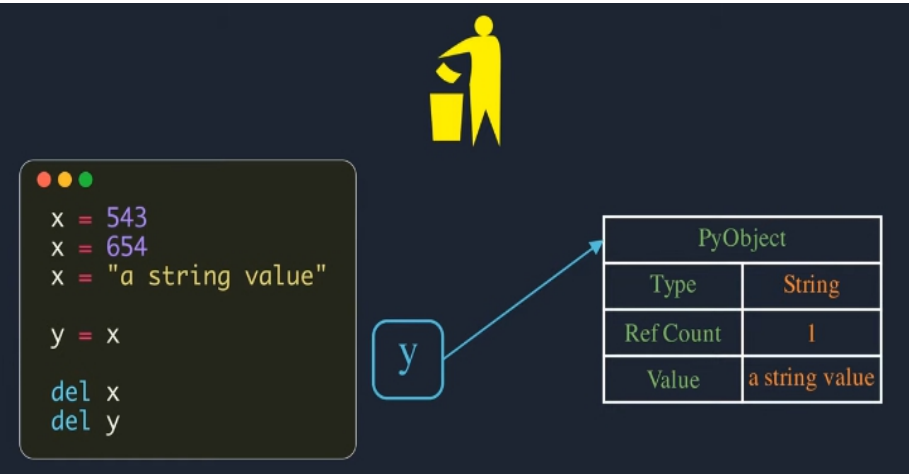

x = 543
x = 654
X
PyObject
Type
Integer
Ref Count
1
Value
543
PyObject
Type
Integer
Ref Count
1
Value
654

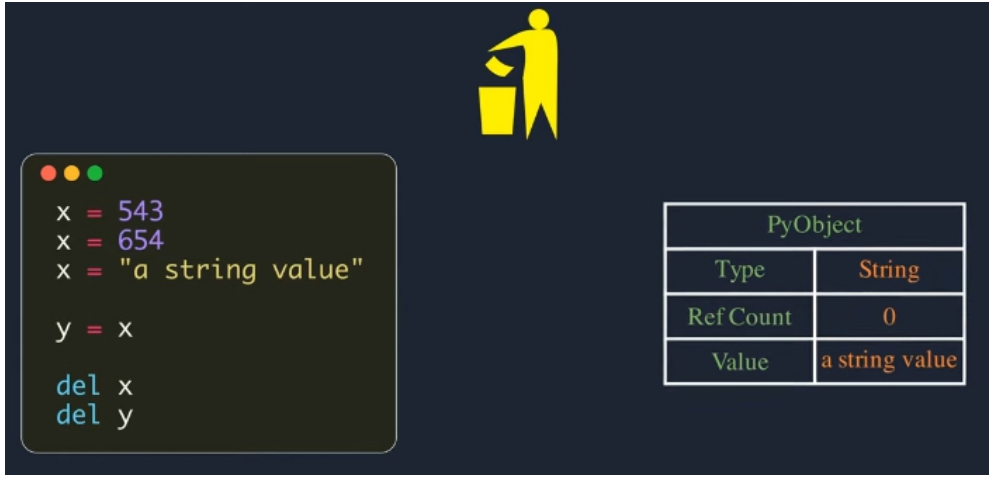

x = 543
x = 654
X
PyObject
Type
Integer
Ref Count
1
Value
654


x = 543
x = 654
x = "a string value"
X
PyObject
Type
String
Ref Count
0
Value
a string value
PyObject
Type
Integer
Ref Count
1
Value
654


x = 543
x = 654
x = "a string value"
X
PyObject
Type
String
Ref Count
1
Value
a string value


x = 543
x = 654
x = "a string value"
y = x
x
y
PyObject
Type
String
Ref Count
2
Value
a string value


x = 543
x = 654
x = "a string value"
y = x
del x
del y
y
PyObject
Type
String
Ref Count
1
Value
a string value


x = 543
x = 654
x = "a string value"
y = x
del x
del y
PyObject
Type
String
Ref Count
0
Value
a string value

type() function

Get the variable Type

- You can get the data type of a variable with the type() function.

```
[12]: r='hello'

[13]: type(r)

[13]: str

[14]: m=23.53

[15]: type(m)

[15]: float

[16]: i=56544
      type(i)

[16]: int
```

Python Casting (Type Conversion)

You can explicitly cast, or convert, a variable from one type to another.

Type casting: int()

To explicitly convert a float number, a string, or boolean value to an integer, cast the number using int() function.

- Cast float to int

[2]:
```
f=0.2658
x=int(f)
print(x)
```

```
0
```

- Cast string to int

[3]:
```
print(int("hello"))   # cannot cast string to int
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-3-09db908c8b1b> in <module>
----> 1 print(int("hello"))

ValueError: invalid literal for int() with base 10: 'hello'
```

Type Casting: float()

The float() function returns a floating point number constructed from a number, string, or boolean value

- Cast int to float

```
[4]: print(float(234))
```

```
234.0
```

- Cast string to float

```
[8]: print(float("hi"))        # cannot cast string to float
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-8-1880ba6f98ec> in <module>
----> 1 print(float("hi"))        # cannot cast string to float

ValueError: could not convert string to float: 'hi'
```

Type Casting: str()

The str() function returns a string

- Cast int to string

```
[12]: str(234)
[12]: '234'
```

- Cast float to string

```
[13]: str(23.904)
[13]: '23.904'
```

Type Casting: chr()

- Convert an integer to a string of one character whose ASCII code is same as the integer using chr() function.
- The integer value should be in the range of 0–255.

```
1 ascii_to_char = chr(100)
2 print('Equivalent Character for ASCII value of 100 is ', ascii_to_char)
```

```
Equivalent Character for ASCII value of 100 is  d
```

Boolean Casting

- Any string is True, except empty strings.
- Any number is True, except 0.
- Any list, tuple, set, and dictionary are True, except empty ones.

Type Casting

- Cast bool to int

```
[21]: int(True)
[21]: 1
[22]: int(False)
[22]: 0
```

- Cast bool to float

```
[23]: float(True)
[23]: 1.0
[24]: float(False)
[24]: 0.0
```

- Cast bool to string

```
[25]: str(True)
[25]: 'True'
[26]: str(False)
[26]: 'False'
```

Type Casting: bool()

- Cast int to bool

```
[8]: a=34
     b=bool(a)

[9]: b

[9]: True

[10]: a=0
      b=bool(a)

[11]: b

[11]: False
```

- Cast float to bool

```
[12]: a=-0.5454
      b=bool(a)

[13]: b

[13]: True

[14]: a=-0.0
      b=bool(a)

[15]: b

[15]: False
```

Type Casting: bool()...

- Cast float to bool

```
[16]: a='hi'
      b=bool(a)

[17]: b

[17]: True

[19]: a=""
      b=bool(a)

[20]: b

[20]: False
```

input() function

input() function

- take string from user

```
[152]: input("Please enter your name: ")

       Please enter your name:  noha
[152]: 'noha'
```

```
In [2]: 1 name=input("enter your name: ")

        enter your name: Mona
```

- take number as string from user

```
[151]: input("Please enter an integer: ")

       Please enter an integer:  56
[151]: '56'
```

- take number as integer from user (casting)

```
[150]: int(input("Please enter an integer: "))

       Please enter an integer:  56
[150]: 56
```

print() function

print() function

[112]:
```
i = 256*256
print('The value of i is', i)
```

```
The value of i is 65536
```

[108]:
```
a, b=0, 1  #multiple assignment: the variables a and b simultaneously get the new values 0 and 1.
print('a value= ', a, '\nb value =', b)
```

```
a value=  0
b value = 1
```

[109]:
```
a, b = b, a+b+3
```

[110]:
```
print('a value= ', a, '\nb value =', b)
```

```
a value=  1
b value = 4
```

Excercise

- Given the Radius, Write Python Program to Find the Area (πr^2) and Circumference of a Circle ($2\pi r$).

In [3]:
```
radius = int(input("Enter the radius of a circle"))
area_of_a_circle = 3.1415 * radius * radius
circumference_of_a_circle = 2 * 3.1415 * radius
print(f"Area={area_of_a_circle} andCircumference={circumference_of_a_circle}")
```

```
Enter the radius of a circle5
Area=78.53750000000001 andCircumference=31.415000000000003
```

Excercise

- Write Python Program to Convert the Given Number of Days to a Measure of Time Given in Years, Weeks and Days. For Example, 375 Days Is Equal to 1 Year, 1 Week and 3 Days (Ignore Leap Year).

```
1 number_of_days = int(input("Enter number of days"))
2 number_of_years = int(number_of_days/365)
3 number_of_weeks = int(number_of_days % 365 / 7)
4 remaining_number_of_days = int(number_of_days % 365 % 7)
5 print(f"Years = {number_of_years}, Weeks = {number_of_weeks}, Days = {remaining_number_of_days}")
```

```
Enter number of days375
Years = 1, Weeks = 1, Days = 3
```

Excercises

- Write a python program that read three numbers from the user, then it prints their maimum, mimimum, summation, average, product.

- Write a python program that takes from the user the size of his harddisk in GB to prints the size in MB.

Note: MB=2^{10} * GB

- Solve the book excercise (chapter 2)

Operations

Operators

1. Arithmetic Operators

2. Assignment Operators

3. Comparison Operators

4. Logical Operators

Arithmetic Operators

List of Arithmetic Operators

Operator	Operator Name	Description	Example
+	Addition operator	Adds two operands, producing their sum.	p + q = 5
–	Subtraction operator	Subtracts the two operands, producing their difference.	p – q = –1
*	Multiplication operator	Produces the product of the operands.	p * q = 6
/	Division operator	Produces the quotient of its operands where the left operand is the dividend and the right operand is the divisor.	q / p = 1.5
%	Modulus operator	Divides left hand operand by right hand operand and returns a remainder.	q % p = 1
**	Exponent operator	Performs exponential (power) calculation on operators.	p**q = 8
//	Floor division operator	Returns the integral part of the quotient.	9//2 = 4 and 9.0//2.0 = 4.0

Note: The value of p is 2 and q is 3.

For example,

1. >>> 10+35
   45
2. >>> –10+35
   25
3. >>> 4*2
   8
4. >>> 4**2
   16

Assignment operators

List of Assignment Operators

Operator	Operator Name	Description	Example
=	Assignment	Assigns values from right side operands to left side operand.	z = p + q assigns value of p + q to z
+=	Addition Assignment	Adds the value of right operand to the left operand and assigns the result to left operand.	z += p is equivalent to z = z + p
−=	Subtraction Assignment	Subtracts the value of right operand from the left operand and assigns the result to left operand.	z −= p is equivalent to z = z − p
*=	Multiplication Assignment	Multiplies the value of right operand with the left operand and assigns the result to left operand.	z *= p is equivalent to z = z * p
/=	Division Assignment	Divides the value of right operand with the left operand and assigns the result to left operand.	z /= p is equivalent to z = z / p
**=	Exponentiation Assignment	Evaluates to the result of raising the first operand to the power of the second operand.	z**= p is equivalent to z = z ** p
//=	Floor Division Assignment	Produces the integral part of the quotient of its operands where the left operand is the dividend and the right operand is the divisor.	z //= p is equivalent to z = z // p
%=	Remainder Assignment	Computes the remainder after division and assigns the value to the left operand.	z %= p is equivalent to z = z % p

For example,

1. >>> p = 10
2. >>> q = 12
3. >>> q += p
4. >>> q
   22
5. >>> q *= p
6. >>> q
   220

Assignment operators

If you try to update a variable which doesn't contain any value, you get an error.

```
1. >>> z = z + 1
   NameError: name 'z' is not defined
```

Trying to update variable z which doesn't contain any value results in an error because Python evaluates the right side before it assigns a value to z ①.

```
1. >>> z = 0
2. >>> x = z + 1
```

Comparison Operators

List of Comparison Operators

Operator	Operator Name	Description	Example
==	Equal to	If the values of two operands are equal, then the condition becomes True.	(p == q) is not True.
!=	Not Equal to	If values of two operands are not equal, then the condition becomes True.	(p != q) is True
>	Greater than	If the value of left operand is greater than the value of right operand, then condition becomes True.	(p > q) is not True.
<	Lesser than	If the value of left operand is less than the value of right operand, then condition becomes True.	(p < q) is True.
>=	Greater than or equal to	If the value of left operand is greater than or equal to the value of right operand, then condition becomes True.	(p >= q) is not True.
<=	Lesser than or equal to	If the value of left operand is less than or equal to the value of right operand, then condition becomes True.	(p <= q) is True.

Note: The value of p is 10 and q is 20.

For example,

1. >>>10 == 12
   False
2. >>>10 != 12
   True
3. >>>10 < 12
   True
4. >>>10 > 12
   False
5. >>>10 <= 12
   True

Logical Operator

List of Logical Operators

Operator	Operator Name	Description	Example
and	Logical AND	Performs AND operation and the result is True when both operands are True	p and q results in False
or	Logical OR	Performs OR operation and the result is True when any one of both operand is True	p or q results in True
not	Logical NOT	Reverses the operand state	not p results in False

Note: The Boolean value of p is True and q is False.

Boolean Logic Truth Table

P	Q	P and Q	P or Q	Not P
True	True	True	True	False
True	False	False	True	
False	True	False	True	True
False	False	False	False	

For example,

1. >>> True and False
   False
2. >>> True or False
   True
3. >>> not(True) and False
   False
4. >>> not(True and False)
   True
5. >>> (10 < 0) and (10 > 2)
   False
6. >>> (10 < 0) or (10 > 2)
   True
7. >>> not(10 < 0) or (10 > 2)
   True
8. >>> not(10 < 0 or 10 > 2)
   False

Operator Precedence

Operator Precedence in Python

Operators	Meaning
()	Parentheses
**	Exponent
+x, −x, ~x	Unary plus, Unary minus, Bitwise NOT
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, −	Addition, Subtraction
<<, >>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
==, !=, >, >=, <, <=, is, is not, in, not in	Comparisons, Identity, Membership operators
not	Logical NOT
and	Logical AND
or	Logical OR

Consider the following code,

1. >>> 2 + 3 * 6
   20
2. >>> (2 + 3) * 6
   30
3. >>> 6 * 4 / 2
   12.0

identity operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

- Operator **is** evaluates to True if the values of operands on either side of the operator point to the same object and False otherwise.
- The operator **is not** evaluates to False if the values of operands on either side of the operator point to the same object and True otherwise.

Membership operator

Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y