Numbering and Operators

Numbers

- There are three numeric types in Python:

  1. int

     Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

```
x = 1
y = 35656222554887711
z = -3255522
```

  2. float

     - Float, or "floating point number" is a number, positive or negative, containing one or more decimals.

```
x = 1.10
y = 1.0
z = -35.59
```

Numbers cont.

- Float can also be scientific numbers with an "e" to indicate the power of 10.

```
x = 35e3
y = 12E4
z = -87.7e100
```

3. complex

Complex numbers are written with a "j" as the imaginary part.

```
x = 3+5j
y = 5j
z = -5j
```

Random Number

- Python does not have a random() function to make a random number, but Python has a built-in module called random that can be used to make random numbers.

```
import random

x = random.randrange(1, 10)
print(x)
```

```
In [7]: runfile('C:/Users/Computer Market/.spyder-py3/temp.py', wdir='C:/Users/Computer
Market/.spyder-py3')
5
```

Operators

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Arithmetic operators

Arithmetic operators are used with numeric values

Operator	Name	Description	Example
+	Addition		x + y
-	Subscription		x – y
*	Multiplication		x * y
/	Division		x / y
%	Modulus	Return the reminder	x % y
**	Exponentiation	Return the power result	x ** y
//	Floor division	rounds the result down to the nearest whole number	x // y

Assignment operators

Operator	Example	Same As
=	x = 4	x = 4
+=	x += 4	x = x + 4
-=	x -= 4	x = x – 4
*=	x *= 4	x = x * 4
/=	x /= 4	x = x / 4
%=	x %= 4	x = x % 4
//=	x =// 4	x = x // 4
**=	x **= 4	x = x ** 4
&=	x &= 4	x = x & 4
|=	x |= 4	x = x | 4
^=	x ^= 4	x = x ^ 4
>>=	x >>= 4	x = x >> 4
<<=	x <<= 4	x = x << 4

Comparison operators

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical operators

- Logical operators are used to combine conditional statements.

Operator	Description	Example
and	Returns True if both statements are true.	x < 5 and x < 10
or	Returns True if one of the statements is true.	x < 5 or x < 10
not	Reverse the result, returns False if the result is true.	not(x < 5 and x < 10)

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
is	Returns True if both variables are the same object.
is not	Returns True if both variables are not the same object.

Identity example..

```
1  x = ["apple", "banana"]
2  y = ["apple", "banana"]
3  z = x
4
5  print(x is z)
6
7  # returns True because z is the same object as x
8
9  print(x is y)
10
11 """
12 returns False because x is not the same object as y,
13 even if they have the same content
14 """
15
16 print(x == y)
17
18 """
19 to demonstrate the difference betweeen "is" and "==":
20 this comparison returns True because x is equal to y
21 """
```

Membership operators

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

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

Membership example..

```
x = ["apple", "banana"]

print("banana" in x)
"""
returns True because a sequence with the value "banana"
is in the list
"""
y = ["apple", "banana"]
print("pineapple" not in y)
"""
returns True because a sequence with the value "pineapple"
is not in the list
"""
```

Bitwise operators

- used to compare (binary) numbers.

Operators	Name	Description
&	And	Sets each bit to 1 if both bits are 1.
|	Or	Sets each bit to 1 if one of two bits is 1.
^	Xor	Sets each bit to 1 if only one of two bits is 1.
~	Not	Inverts all the bits.
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off.
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off.

Problems

- Ask the user for a number. Depending on whether the number is even or odd, print out an appropriate message to the user. *Hint: how does an even / odd number react differently when divided by 2?*
- Ask user for 2 numbers and do four different Arithmetic operation to it and display the result.
- Ask user for 2 numbers "*num1 , num2*" and compare them, if num1 is greater than num2, display the appropriate message otherwise display another one.
- assigned to variable and do left shift and right shift and display the results.

  (*Hint: apply this operation on each of decimal and binary*)

Any Question…?