Python Data Type And Operators

資料型態與運算子

Example: Get an Odd Integer

Jiann-Ming Wu, Sep 23, 2024

Python Tutorial

Release 3.7.0

Guido van Rossum
and the Python development team

Press right key and run

Choose one file to run

ex1_2021 〉 ex2.py

ex2

Project

ex1_2021 ~/Desktop/py_

venv

abc.png

ex1.py

ex2.py

External Libraries

Scratches and Consoles

ex1.py ex2.py

```
1  ss = "123"
2  num = int(ss)
3  print(num)
```

Editor

Run: ex2

```
/Users/apple/Desktop/py_code_2020/ex1_2021/venv/bin/
123

Process finished with exit code 0
```

Run window output window

1: Project

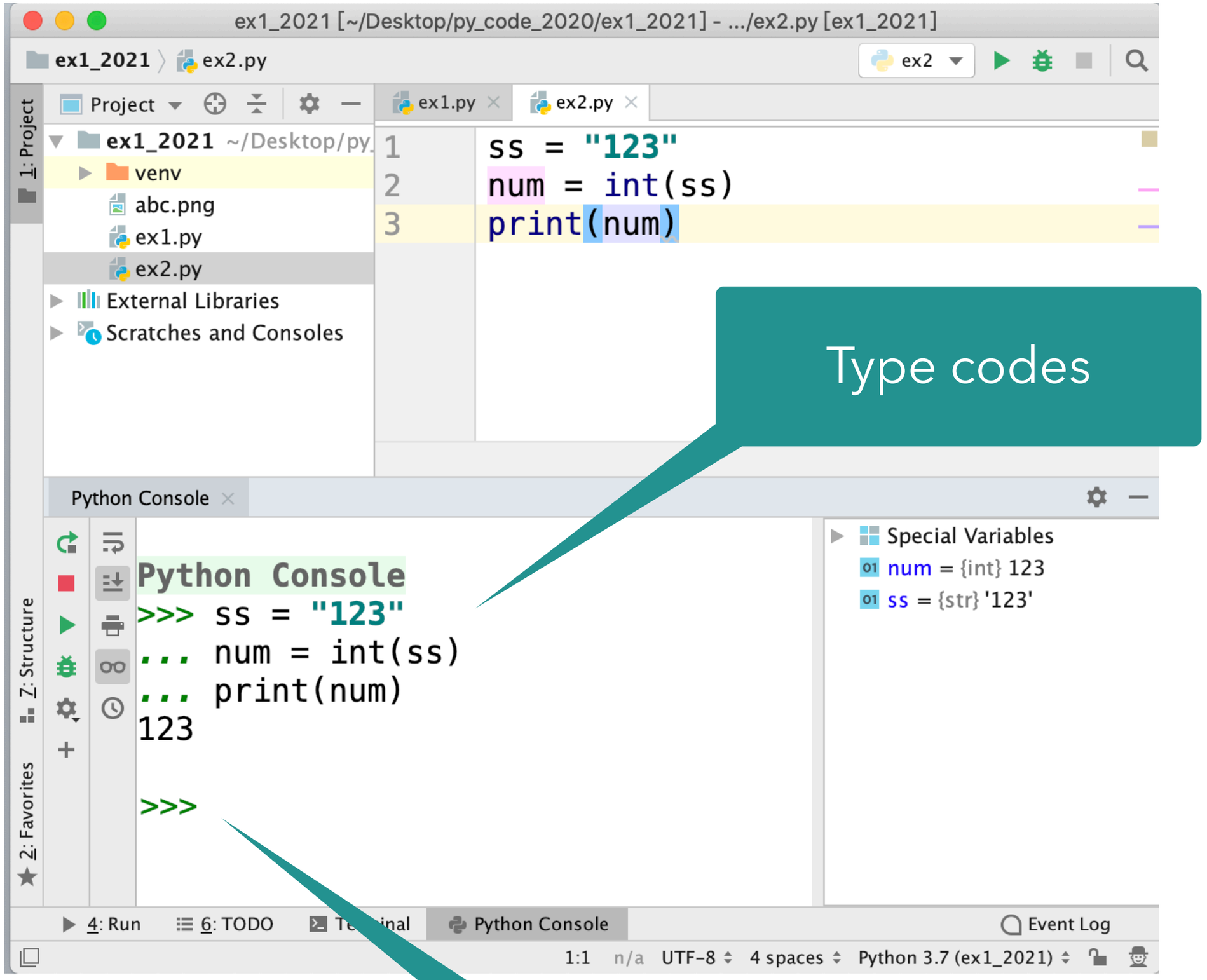
7: Structure

2: Favorites

4: Run 6: TODO Terminal Python Console Event Log

4 spaces Python 3.7 (ex1_2021)

Python Console


ex1_2021 [~/Desktop/py_code_2020/ex1_2021] - .../ex2.py [ex1_2021]
ex1_2021
ex2.py
ex2
Project
ex1_2021 ~/Desktop/py
venv
abc.png
ex1.py
ex2.py
External Libraries
Scratches and Consoles
ex1.py
ex2.py
1 ss = "123"
2 num = int(ss)
3 print(num)
Type codes
Python Console
Python Console
>>> ss = "123"
... num = int(ss)
... print(num)
123
>>>
Special Variables
num = {int} 123
ss = {str} '123'
1: Project
2: Favorites
7: Structure
4: Run
6: TODO
Python Console
Event Log
1:1
n/a
UTF-8
4 spaces
Python 3.7 (ex1_2021)
Python Console

Python Data Type

Data Type

- int —-> whole numbers —-> 3, 300, 200
- float —-> Numbers with decimal point —-> 2.3, 4.6, 100.0
- str —-> ordered sequence of characters —-> “hello”, “concepts” , “2000”, “_P*#@$!”
- dict —-> unordered key : value pairs —-> {“key1” : “value1”, “name” : “crock concepts”}
- tup —-> ordered immutable sequence of objects —-> (10, “hello”. 200.3)
- set ——> unordered collection of unique objects —-> {“a”, “b”}

```
>>> x = 5
>>> type(x)
<class 'int'>
>>> type(-6)
<class 'int'>
```

Integers:

Positive or negative whole numbers; without a decimal point

```
>>> x2 = 4.75
>>> type(x2)
<class 'float'>
>>> int(x2)
4
>>> float(5)
5.0
```

Floating points (floats):

Real numbers; with a decimal point

```
>>> x3 = True
>>> type(x3)
<class 'bool'>
>>> x4 = true
Traceback (most recent call last):
  File "<input>", line 1, in <module>
NameError: name 'true' is not defined
```

True;
False

Boolean values:

A "True" or "False" value

```
>>> 'George'
'George'
>>> "George"
'George'
>>> print('George')
George
>>> print("George")
George
```

Strings:

Text values composed of a sequence of characters

```
>>> y = 10
>>> print(str(y) + " dolars")
10 dolars
>>> 'I'm fine'
  File "<input>", line 1
    'I'm fine'
       ^
SyntaxError: invalid syntax
>>> "I'm fine"
"I'm fine"
>>> 'I\' fine'
"I' fine"
```

```
>>> 'press "Enter"'
'press "Enter"'
>>> 'red' 'car'
'redcar'
>>> 'red ' 'car'
'red car'
>>> 'red '+'car'
'red car'
>>> print('red '+'car')
red car
>>> print('red ','car')
red  car
```

```
>>> print(3,5)
3 5
>>> type((3,5,6.9,7.0,'car'))
<class 'tuple'>
```

Python Operator

Numerical Operator

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

```
Python Console
>>> 5 % 3
2
>>> 5**3
125
>>> 5//3
1
```

Python Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

Operator	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

```
>>> True and True
True
>>> True and False
False
>>> False or False
False
>>> False or True
True
>>> True or True
True
>>> True & True
True
>>> False | False
False
```

```
>>> True ^ True
False
>>> True ^ False
True
>>> False ^ True
True
>>> False ^ False
False
```

```
>>> 0b101 << 2
20
>>> bin(0b101 << 2)
'0b10100'
>>> 0b101 >> 2
1
>>> 0b111101 >> 2
15
>>> bin(15)
'0b1111'
```

Python Membership Operators

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

```
>>> x = 5
>>> y = [ 5, 3, 2, 1]
>>> x in y
True
>>> 6 not in y
True
```

Python Comparison Operators

Comparison operators are used to compare two values:

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

```
>>> 5 == 6
False
>>> 5 != 6
True
>>> "a" < "b"
True
>>> "aa" > "ab"
False
```

Python 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 < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Operator	Example	Same As
=	x = 5	x = 5
+=	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
&=	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

習題：

Get an odd number

本例題的while迴圈的進入條件永遠為真

```
while True:
    ...
    continue
    ...
    break
    ...
```

執行到continue忽略continue後面的指令，繼續進行迴圈執行

執行到break忽略break後面的指令，結束迴圈執行

```
>>> ss = input("please input an odd number (between 0-100):")
... print(ss)
please input an odd number (between 0-100):>? abc
abc
```

列印結果

輸入

```
>>> ss = input("please input an odd number (between 0-100):")
... print(ss)
please input an odd number (between 0-100):>? 1234
1234
```

使用者有輸入字串：

可轉換數整數或不可轉換為整數

將ss的內容轉換為整數，如果轉換失敗，num的內容無定義

當int轉換失敗執行except：指令

```
try:
        num = int(ss)
except:
        print("Invalid number, please input again.")
```

```
>>> try:
...     print(int("abc"))
... except:
...     print("invalid")
...
invalid
```

轉換失敗

```
>>> try:
...     print(int("123"))
... except:
...     print("invalid")
...
123
```

轉換成功

num除2餘數等於1

```
if num%2 == 1:
        print(ss+" is odd")
        break
else:
        print(ss+" is even. Input again")
```

中斷while回圈執行

否則

Get an odd number

```
while True:
    ss = input("please input an odd number (between 0-100):")
    try:
        num = int(ss)
    except:
        print("Invalid number, please input again.")
        continue
    if num%2 == 1:
        print(ss+" is odd")
        break
    else:
        print(ss+" is even. Input again")
```

執行直到迴圈進入條件不成立

將字串轉為整數

except代表轉換失敗

continue繼續到while迴圈的進入指令執行

如果餘數是1，代表奇數，break會中斷迴圈

輸入的字串轉換為整數，失敗

輸入的字串

```
please input an odd number (between 0-100):we2
Invalid number, please input again.
please input an odd number (between 0-100):22
22 is even. Input again
please input an odd number (between 0-100):21
21 is odd
```

輸入的字串轉換為整數，判斷為偶數

輸入的字串轉換為整數，是奇數

Example:

請使用者輸入1-12中一個整數，輸出對應的十二生肖名稱與圖片

錯誤處理：

1. 如果使用輸入非整數，重新輸入
2. 如果使用者輸入的整數超出1-12的範圍，重新輸入