VScode and Python

Day2: Basic Syntax Structure of Python

Dr. Byoung-gyu Gong

Assistant professor

Information System

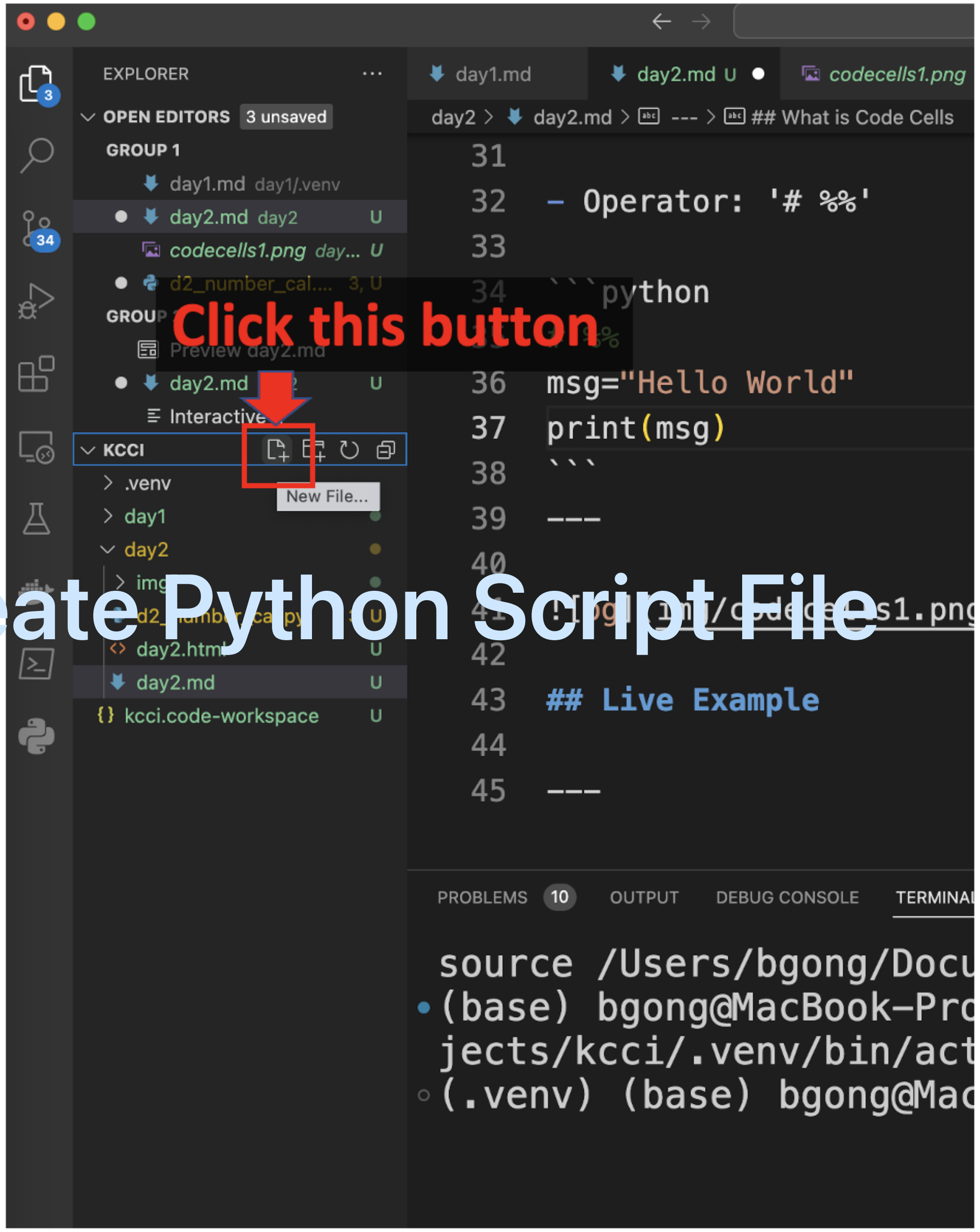
Davide Eccles Business School

University of Utah Asia Campus

Contents

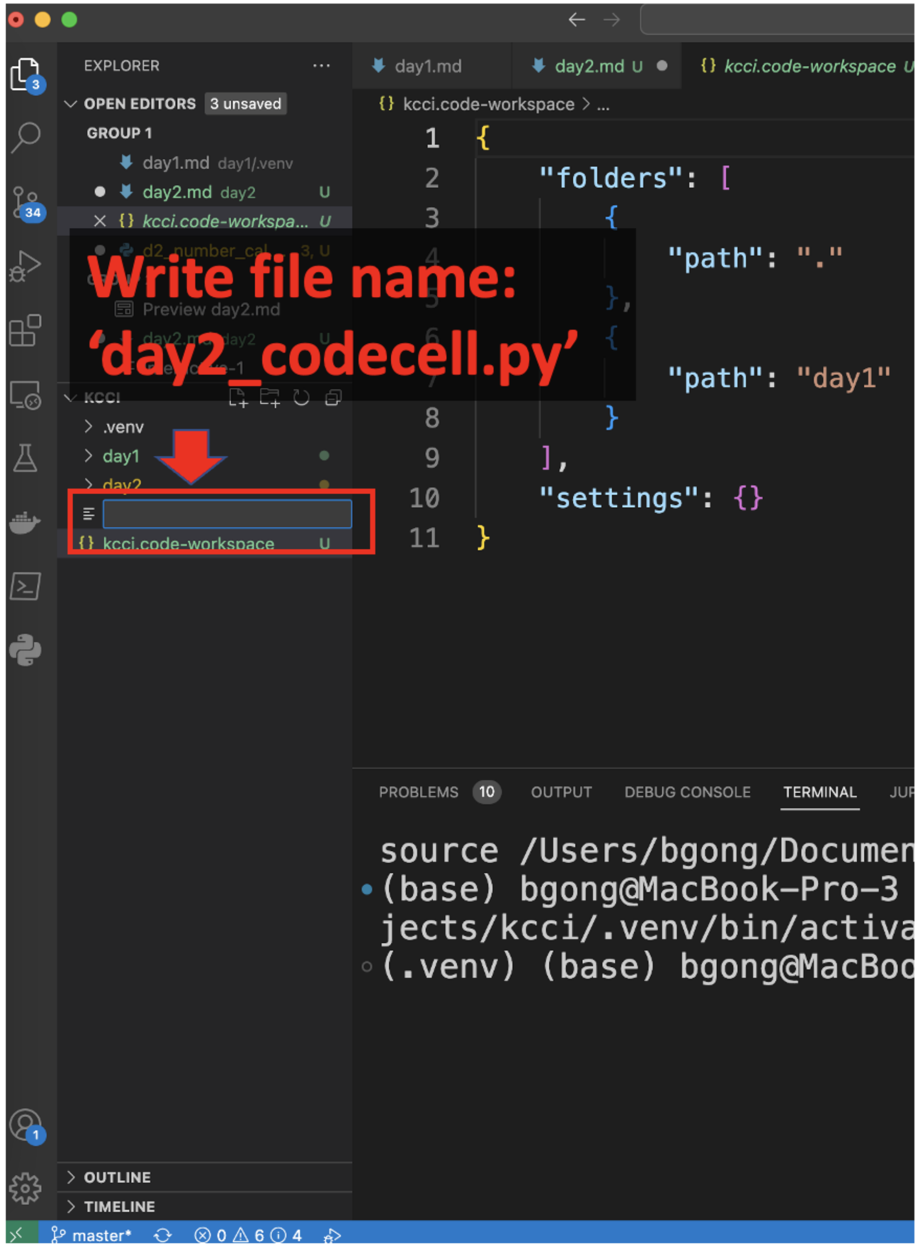
First Project

Create Python Script File



~~~
- Operator: '# %%'

```python
# %%
msg="Hello World"
print(msg)
```

---

![png](img/codecells1.png

## Live Example

---
~~~

```
source /Users/bgong/Docu
(base) bgong@MacBook-Pro
jects/kcci/.venv/bin/act
(.venv) (base) bgong@Mac
```

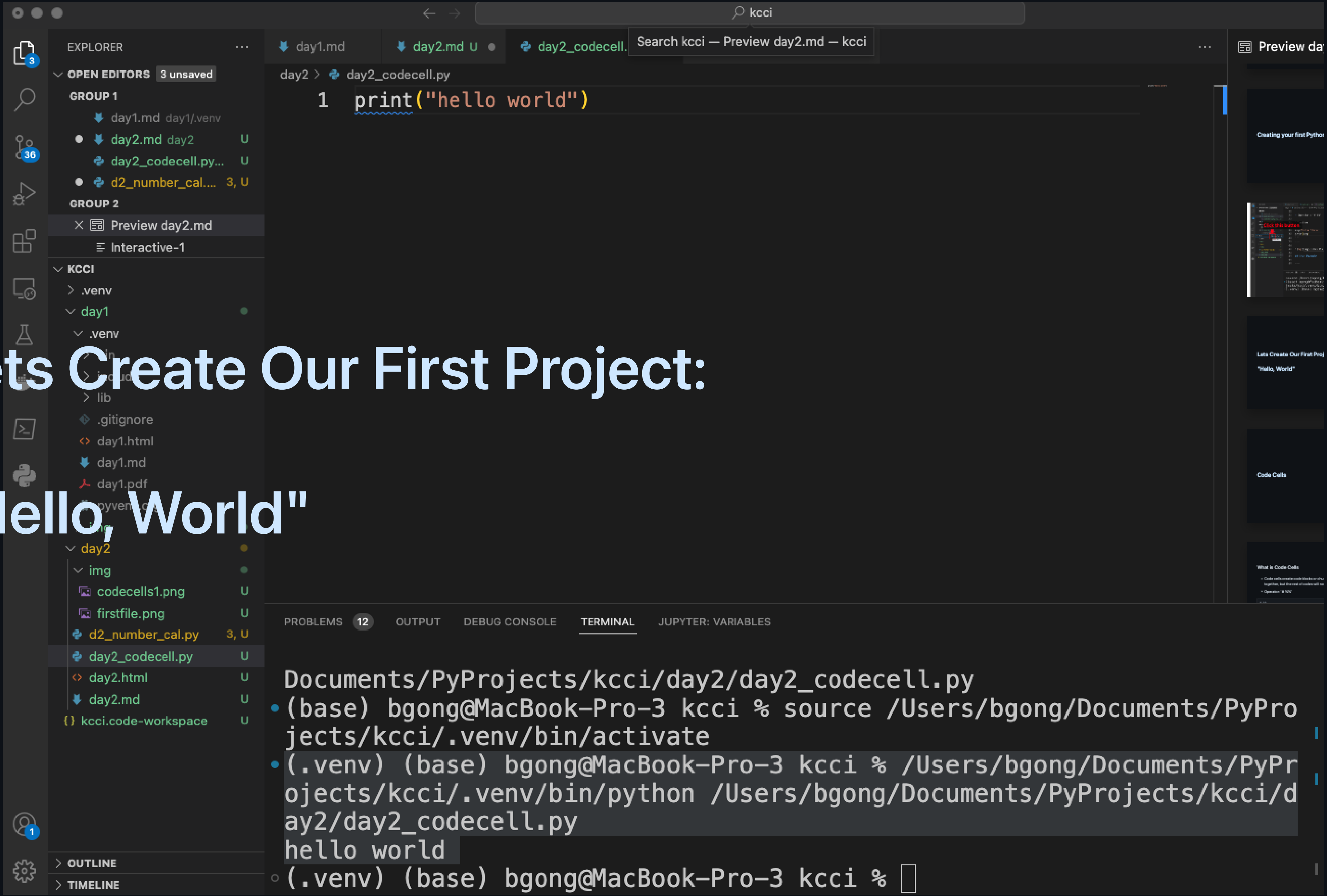


```
{
    "folders": [
        {
            "path": "."
        },
        {
            "path": "day1"
        }
    ],
    "settings": {}
}
```

```
source /Users/bgong/Documen
(base) bgong@MacBook-Pro-3
jects/kcci/.venv/bin/activa
(.venv) (base) bgong@MacBoo
```

Lets Create Our First Project:

"Hello, World"



```
print("hello world")
```

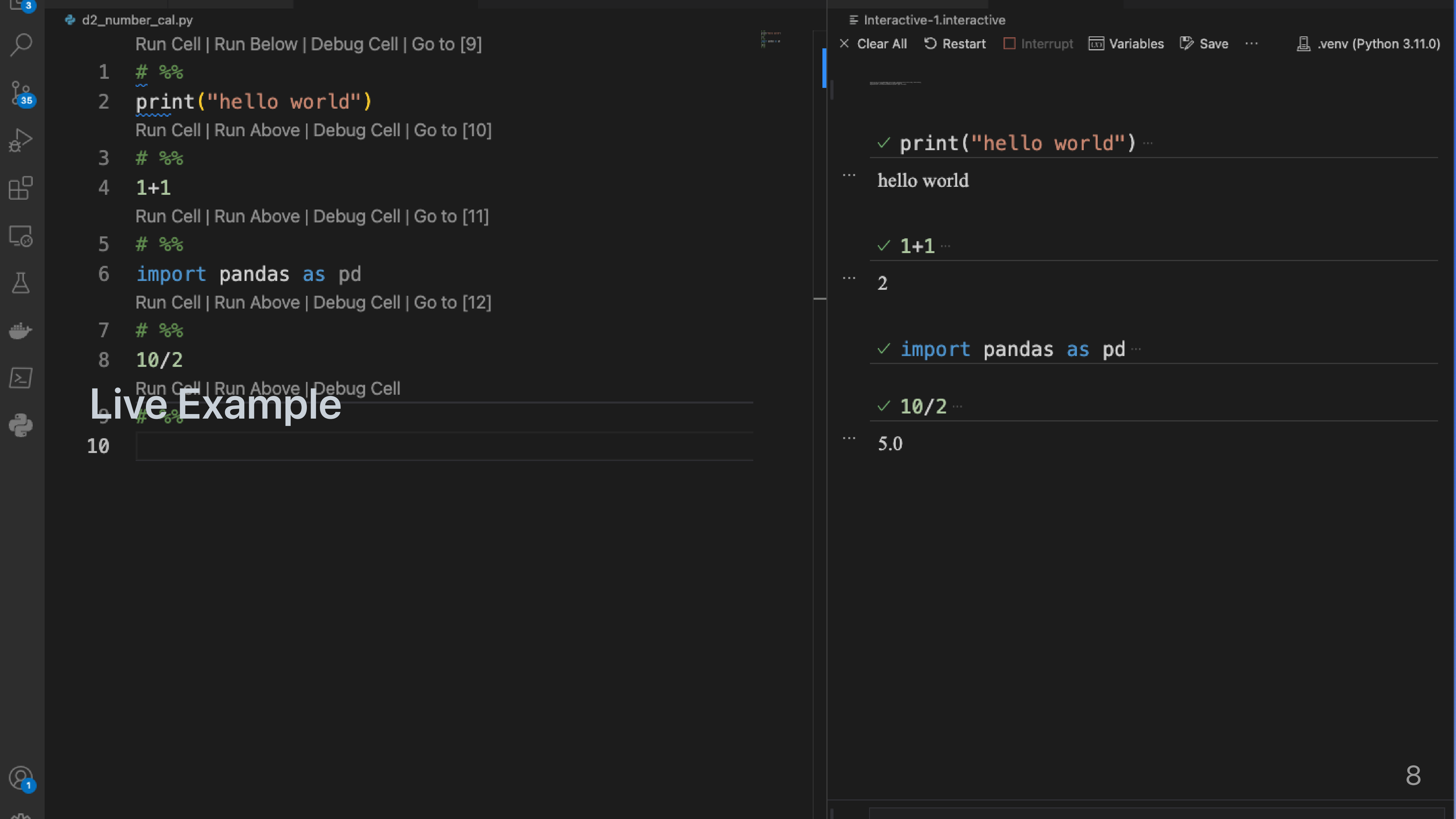
```
Documents/PyProjects/kcci/day2/day2_codecell.py
(base) bgong@MacBook-Pro-3 kcci % source /Users/bgong/Documents/PyPro
jects/kcci/.venv/bin/activate
(.venv) (base) bgong@MacBook-Pro-3 kcci % /Users/bgong/Documents/PyPr
ojects/kcci/.venv/bin/python /Users/bgong/Documents/PyProjects/kcci/d
ay2/day2_codecell.py
hello world
(.venv) (base) bgong@MacBook-Pro-3 kcci %
```

Code Cells

What is Code Cells

- Code cells create code blocks or chunks that are bind together. They are executed all together, but the rest of codes will not be executed.
- Operator: '# %%'

```
# Run Cell|Run Below|Debug Cell
# %%
msg="Hello World"
print(msg)
```



Live Example

d2_number_cal.py

```
Run Cell | Run Below | Debug Cell | Go to [9]
# %%
print("hello world")
Run Cell | Run Above | Debug Cell | Go to [10]
# %%
1+1
Run Cell | Run Above | Debug Cell | Go to [11]
# %%
import pandas as pd
Run Cell | Run Above | Debug Cell | Go to [12]
# %%
10/2
Run Cell | Run Above | Debug Cell
# %%
```

Interactive-1.interactive

Clear All | Restart | Interrupt | Variables | Save | .venv (Python 3.11.0)

```
✓ print("hello world")
hello world

✓ 1+1
2

✓ import pandas as pd

✓ 10/2
5.0
```

Basic Manipulations in Python

1. Number Calculations

```
# %%
2+2
```

```
4
```

```
# %%
100*2
```

```
200
```

```
# %%
100/4
```

```
25
```

```python
# %%
# Squared calculation
2 ** 2
```

```
4
```

```python
# %%
# Calculating remainder of division
15 % 2
```

```
1
```

2. String Manipulations

2.1. Printing Format

```
# String without quotation mark
fall
```

NameError: name 'fall' is not defined

```
# String with quotation mark
'fall'
```

'fall'

```
# String with quotation mark + print()
print('fall')
```

fall

- So you should put qutation mark('') for the string values.
- By using print() you can create more readable output.

2.2. Concatenation

- We can manipulate strings using arithmatic operators(+,-,*)

```
print('K'+2*'C'+'I')
```

KCCI

```
print('py''thon')
```

python

Excercise-Question

Your monthly salary after the tax is $1000. This month your spending looks like this:

- Rent: $300
- Grocery: $300
- Others: $100

Please write down your formula in Python code.

1. How much is your total monthly spending?
2. How much is the remaining after spending?

Excercise -Answer

Your monthly salary after the tax is $1000. This month your spending looks like this:

```
# Salary:1000
# Rent:300
# Grocery:300
# Others:100
```

```
#Question1.Total monthly spending
300+300+100
```

700

```
#Question2.Remaining balance
1000-700
```

300

Variables

Basics

Variables indicates objects in the Python programming. They should be defined and declared to have specific value or function in them using operator '='.

For instance, we can insert specific value into the string name variables:

```
a=1
b=2
a,b
```

(1,2)

```
# Even you can calculate variables having numeric values in it.
a+b
```

3

Variable Manipulation: Syntax

'object.function(conditions)'

```
c="John"
b="10 years old"
print(c,":",b)
```

1. replace()

```
# replace(old,new)
b.replace('10','12')
```

```
'12 years old'
```

2. split()

```
# split(delimiter)
c=b.split(' ')
```

['10', 'years', 'old']

3. 'delimiter'.join()

```
d=' '.join(c)
d
```

'10 years old'

3. strip()

```python
# split(delimiter)
txt = "     banana     "
txt.strip()
```

```
'banana'
```

4. rstrip(),lstrip()

```python
txt = ' John '
print('[' + txt.rstrip() + ']')
print('[' + txt.lstrip() + ']')
print('[' + txt.strip() + ']')
```

```
[ John]
[John ]
[John]
```

Exercise-Question

Please convert the following phone numbers into pure numbers - delete the hyphen('-') in the string.

```
phone1='800-294-2934'
phone2='800-293-4920'
phone3='602-493-2999'
```

Exercise-Answer

Please convert the following phone numbers into pure numbers - delete the hyphen('-') in the string.

```
phone1='800-294-2934'
phone2='800-293-4920'
phone3='602-493-2999'
```

```
a=phone1.replace('-','')
b=phone2.replace('-','')
c=phone3.replace('-','')
a,b,c
```

('8002942934', '8002934920', '6024932999')

Indexing

Indexing is very critical function to process and manipulate the variables in python because it provides you a sophisticated method to clean or modify your data.

```
word='Python'
word[0]
```

'P

```
word[:2]
```

'Py'

```
word[:6]
```

'Python'

```
x='selflearning'
'-'.join([x[:4], x[4:12]])
```

```
'self-learning'
```

Exercise - Question

Please convert the following phone numbers into the number with the hyphen('-') in the string.

```
phone1='8002942934'
phone2='8002934920'
phone3='6024932999'
```

Exercise - Answer

```
phone1='8002942934'
phone2='8002934920'
phone3='6024932999'

'-'.join([phone1[:3], phone1[3:6], phone1[6:]])
'-'.join([phone2[:3], phone2[3:6], phone1[6:]])
'-'.join([phone3[:3], phone3[3:6], phone1[6:]])
```

```
'800-294-2934'
'800-293-4920'
'602-493-2999'
```