

## **A. Task được giao về nhà:**

### **I. Data types:**

Có 8 loại:

1. **Text Type:** str => string

2. **Numeric Types:** int, float, complex:

- Int => dùng cho số nguyên.
- Float => dùng cho số thực có dấu thập phân hoặc module thập phân cho tính toán cần độ chính xác cao.
- Complex => số phức (có thể khai báo kiểu gán thông thường hoặc dùng hàm `complex()` ).

\*Kiểm tra xem 1 số x thuộc kiểu dữ liệu dạng số nào => `print (type(x))`

\*Có thể chuyển đổi các kiểu dữ liệu số qua lại lẫn nhau (int qua float và complex, float qua int và complex nhưng complex không thể chuyển qua int và float):

Ví dụ có số 5 là kiểu int => Ta có thể chuyển qua số thực dấu phẩy động và số phức như sau:

`X = float(5)` => kết quả là 5.0

`X = complex(5)` => kết quả là 5 + 0j

⇒ Tránh chuyển đổi số thực dấu phẩy động qua số nguyên để tránh mất mát dữ liệu (ví dụ `int(5.4) = 5`).

3. **Sequence Types:** list, tuple, range:

#### **3.1. List:**

- Lưu trữ nhiều phần tử vào 1 biến.
- Có thể thay đổi các phần tử trong mảng, thông qua index, range, `listName.insert(index, value)`.
- Các phần tử được chứa trong dấu ngoặc vuông, các phần tử trong mảng có thể thuộc cùng kiểu hoặc khác kiểu dữ liệu.
- Các phần tử được xếp thứ tự và được đánh index, truy cập vào các phần tử trong mảng theo index: nếu theo thứ tự từ trái sang thì phần tử đầu tiên có index=0, nếu từ phải sang thì phần tử đầu tiên có index=-1. Cũng có thể truy cập và in ra các phần tử trong mảng theo range: ví dụ `print(list[a:b])` thì kết quả thu được là các phần tử có index từ a cho đến b-1 trong mảng ; Ngoài ra range cũng có thể chỉ có 1 chỉ số, ví dụ như `[1:]`=> từ phần tử có index bằng 1 đến phần tử cuối trong mảng, `[:4]`=> từ phần tử đầu tiên đến phần tử có index=3 trong mảng.
- Thêm phần tử vào cuối mảng bằng cách `listName.append(value)`.
- Nối 2 list với nhau bằng cách: `list1.extend(list2)` => list2 sẽ được nối vào list1 và kết quả là list2 không thay đổi, còn list1 mới sẽ bằng list1 cũ nối list2; cũng có thể dùng `extend` để nối list với tuple.
- Xóa phần tử trong list có 4 cách:
  - + `listName.remove(value)`
  - + `listName.pop(index)` => Nếu không có index thì mặc định xóa phần tử cuối cùng.
  - + `del listName` => xóa toàn bộ list.

+ `listName.clear()` => xoá toàn bộ phần tử và đưa về list rỗng.

### 3.2. **Tuple:**

- Về cơ bản giống list nhưng giá trị phần tử không thể thay đổi, các phần tử được chứa trong ngoặc tròn.
- Để có thể thay đổi giá trị phần tử trong tuple thì cần chuyển tuple về list rồi thực hiện thay đổi như với list, sau đó chuyển list về lại tuple.

### 4. **Mapping Type: dict:**

- Chứa các phần tử theo cặp key-value.
- Các phần tử có thể thay đổi giá trị value.
- Value có thể là bất kỳ kiểu dữ liệu nào.
- Truy cập vào dict:
  - + `dictName.get(key)`.
  - + `dictName.keys()` => trả về tất cả key.
  - + `dictName.values()` => trả về tất cả value.
  - + `dictName.items()` => trả ra tất cả item trong dict theo cặp key-value, mỗi cặp được chứa trong 1 tuple.
- Thêm phần tử vào dict: `dictName.update(cặp key-value)` (cũng có thể dùng để đổi giá trị của 1 key).
- Xoá phần tử khỏi dict:
  - + `dictName.pop(key)`
  - + `dictName.popitem()` => xoá cặp key-value cuối dict
  - + `del dictName[key]` => xoá cặp key-value theo key (không thể `del dictName`)
  - + `dictName.clear()` => xoá tất cả cặp key-value và trả về dict rỗng

### 5. **Set Types: set, frozenset.**

- Set về cơ bản giống list và tuple nhưng không có thứ tự giữa các phần tử, không thể thay đổi phần tử, không được sắp xếp.
- Các phần tử được chứa trong ngoặc nhọn và không có phần tử có giá trị bằng nhau.
- Không thể thay đổi phần tử nhưng có thể thêm và xoá.
- Không thể truy cập vào phần tử trong set theo index nên phải dùng vòng lặp hoặc truy cập qua giá trị.
- Nối set dùng `set1.update(set2)`.
- Xoá phần tử khỏi set:
  - + `setName.remove(value)`
  - + `setName.discard(value)` (nếu value không tồn tại thì không báo lỗi, còn `remove` có).
  - + `setName.pop()` => random loại 1 phần tử.
  - + `setName.clear()` => xoá toàn bộ phần tử, đưa về set rỗng.
  - + `del setName` => xoá hoàn toàn set.

⇒ Phân biệt cách dùng list, tuple và set:

- List: các phần tử có thứ tự và có thể thay đổi.
- Tuple: các phần tử có thứ tự nhưng không cần thay đổi.
- Set: tập hợp các phần tử duy nhất và không cần thứ tự.

6. **Boolean Type:** bool => true/false.

7. **Binary Types**

8. **None Type**

II. **Step để viết 1 function chuẩn cho một logic:**

- Xác định các yêu cầu của logic.
- Thiết kế hàm với tên hàm, tham số đầu vào, kết quả mà hàm trả về nhằm xử lý logic, sau đó thực hiện viết hàm.
- Thiết kế các test cases để test các trường hợp có thể xảy ra của logic đó, nhất là các trường hợp đặc biệt để đảm bảo sự tối ưu và xử lý mọi tình huống của hàm.
- Tối ưu hàm bằng cách check lại xem code đã rõ ràng và dễ hiểu chưa để sửa lại nếu cần, hoặc xem xét thêm những giải pháp, hướng đi khác có thể giúp hàm trở nên tối ưu hơn, sử dụng comment để note lại những ý còn khó hiểu trong hàm hoặc để hàm trở nên rõ ràng hơn.

III. **Bài thực hành tuần 1:**

**Bài 1:** Cài đặt Python và viết chương trình “Hello World”:

- Cài đặt Python:

```
[(base) macbook@HuyTuan ~ % python --version  
Python 3.12.4
```

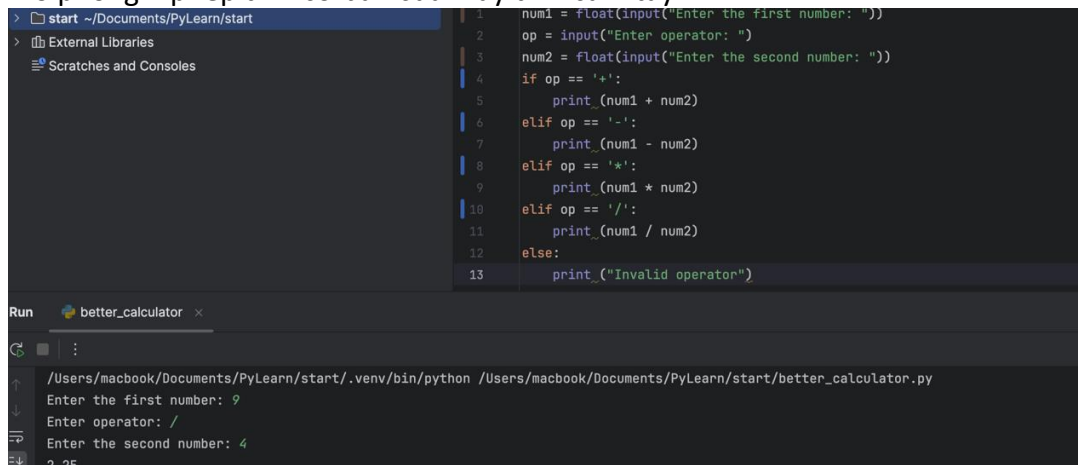
- Viết chương trình “Hello World”:



```
1 print("Hello World")  
  
Run Hello_World x  
/Users/macbook/Documents/PyLearn/start/.venv/bin/python /Users/macbook/Documents/PyLearn/start/Hello_World.py  
Hello World
```

**Bài 2:** Cài đặt và cấu hình 1 IDE, viết chương trình nhập dữ liệu từ người dùng và hiển thị kết quả:

- IDE cài đặt: Pycharm.
- Chương trình nhận dữ liệu từ người dùng và hiển thị kết quả: Chương trình mô phỏng 4 phép tính cơ bản của máy tính cầm tay:



```
1 num1 = float(input("Enter the first number: "))  
2 op = input("Enter operator: ")  
3 num2 = float(input("Enter the second number: "))  
4 if op == '+':  
5     print(num1 + num2)  
6 elif op == '-':  
7     print(num1 - num2)  
8 elif op == '*':  
9     print(num1 * num2)  
10 elif op == '/':  
11     print(num1 / num2)  
12 else:  
13     print("Invalid operator")  
  
Run better_calculator x  
/Users/macbook/Documents/PyLearn/start/.venv/bin/python /Users/macbook/Documents/PyLearn/start/better_calculator.py  
Enter the first number: 9  
Enter operator: /  
Enter the second number: 4  
2.25
```

### **Bài 3:**

- Tạo môi trường ảo với virtualenv:

```
[(base) macbook@HuyTuan ~ % pip install virtualenv
Requirement already satisfied: virtualenv in /opt/anaconda3/lib/python3.12/site-packages (20.26.3)
Requirement already satisfied: distlib<1,>=0.3.7 in /opt/anaconda3/lib/python3.12/site-packages (from virtualenv) (0.3.8)
Requirement already satisfied: filelock<4,>=3.12.2 in /opt/anaconda3/lib/python3.12/site-packages (from virtualenv) (3.13.1)
Requirement already satisfied: platformdirs<5,>=3.9.1 in /opt/anaconda3/lib/python3.12/site-packages (from virtualenv) (3.10.0)
[(base) macbook@HuyTuan ~ % virtualenv myenv
created virtual environment CPython3.12.4.final.0-64 in 231ms
  creator CPython3Posix(dest=/Users/macbook/myenv, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, via=copy, app_data_dir=/Users/macbook/Library/Application Support/virtualenv)
    added seed packages: pip==24.1
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
[(base) macbook@HuyTuan ~ % source myenv/bin/activate
(myenv) (base) macbook@HuyTuan ~ %
```

- Cài đặt Numpy và Pandas:

```
[(myenv) (base) macbook@HuyTuan ~ % python
Python 3.12.4 | packaged by Anaconda, Inc. | (main, Jun 18 2024, 10:07:17) [Clang 14.0.6 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
[print]>>> print(np.__version__)
2.0.1
>>> print(pd.__version__)
2.2.2
>>>
```

- Viết 1 script sử dụng Numpy và Pandas:

```
>>> data = np.arange(5)
>>> df = pd.DataFrame(data, column = ['Numbers'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: DataFrame.__init__() got an unexpected keyword argument 'column'
>>> df = pd.DataFrame(data, columns = ['Numbers'])
>>> df['Squared']=df['Numbers']**2
>>> print(df)
   Numbers  Squared
0         0         0
1         1         1
2         2         4
3         3         9
4         4        16
```

**Bài 4:** Cài đặt Jupyter Notebook và tạo một notebook mới, thực hiện thao tác đơn giản với Numpy và Pandas:

```
[25]: import numpy as np
import sys
import pandas as pd

array = np.random.randint(0, 101, size=(10, 5))

means = np.mean(array, axis=1)
medians = np.median(array, axis=1)
std_devs = np.std(array,axis=1)
mins = np.min(array,axis=1)
maxs = np.max(array, axis=1)

stats_table = pd.DataFrame({
    'Mean': means,
    'Median': medians,
    'Standard Deviation': std_devs,
    'Minimum': mins,
    'Maximum': maxs
})

print(array)
print(stats_table)

print (np.__version__)
print (sys.version)
```

```
[[72  6 49  0  2]
 [ 6 87 12 87 88]
 [38 50 16  4 28]
 [98  4 35 38 43]
 [33  6 82 88 56]
 [96 74 52 60 82]
 [95 52 18 45 21]
 [50 96 64 48 51]
 [51 39 82 32 23]
 [58 26 60 77 76]]
```

	Mean	Median	Standard Deviation	Minimum	Maximum
0	25.8	6.0	29.314843	0	72
1	56.0	87.0	38.423951	6	88
2	27.2	28.0	16.129476	4	50
3	43.6	38.0	30.440762	4	98
4	53.0	56.0	30.607189	6	88
5	72.8	74.0	15.625620	52	96
6	46.2	45.0	27.737339	18	95
7	61.8	51.0	18.004444	48	96
8	45.4	39.0	20.460694	23	82
9	59.4	60.0	18.456435	26	77

```
1.26.4
```

```
3.12.4 | packaged by Anaconda, Inc. | (main, Jun 18 2024, 10:07:17) [Clang 14.0.6 ]
```

**Bài 5:** Cài đặt Anaconda, tạo môi trường bằng conda, cài đặt thư viện và viết 1 chương trình đơn giản sử dụng môi trường conda:

The image shows the Anaconda Navigator application window. The top bar includes the Anaconda Navigator logo and an 'Update Now' button. The left sidebar contains navigation links: Home, Environments, Learning, and Community. The main area displays a grid of application tiles, each with an icon, name, version, description, and a button to either 'Install' or 'Launch' the application. The tiles include:

- PyCharm Professional**: The Python IDE for data science. It combines the interactivity of Jupyter notebooks with intelligent Python coding assistance, Anaconda support, and scientific libraries. (Install)
- Anaconda AI Navigator**: Access various large language models (LLMs) curated by Anaconda, and start leveraging secure local AI today. (Install)
- Anaconda Toolbox** 4.0.15: Anaconda Assistant. JupyterLab supercharged with a suite of Anaconda extensions, starting with the Anaconda Assistant AI chatbot. (Launch)
- Anaconda Cloud Notebooks**: Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage. (Launch)
- JupyterLab** 4.0.11: An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. (Launch)
- Notebook** 7.0.8: Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. (Launch)
- PyCharm Community** 2024.1.4: An IDE by JetBrains for pure Python development. Supports code completion, listing, and debugging. (Launch)
- Qt Console** 5.5.1: PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. (Launch)

Below the grid, there are logos for Watsonx and Oracle. At the bottom, a terminal window is open, showing the execution of a Python script named `guessing_game.py`. The script is a simple word-guessing game where the user has 3 attempts to guess a secret word. The terminal output shows the user guessing 'dog' and 'cat', and the program correctly identifying 'cat' as the secret word.

```

guessing_game.py > [e] secret_word
1 secret_word = "cat"
2 guess = ""
3 guess_limit = 3
4 guess_count = 0
5 out_of_guesses = False
6
7 while guess != secret_word and not(out_of_guesses):
8     if guess_count < guess_limit:
9         guess = input("Enter your guess: ")
10        guess_count+=1
11    else:
12        out_of_guesses = True
13
14 if out_of_guesses:
15     print ("You lose")
16 else:
17     print ("You win!")
18
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
/opt/anaconda3/bin/python /Users/macbook/Documents/Py_mini_prj/guessing_game.py
(base) macbook@HuyTuan Py_mini_prj % /opt/anaconda3/bin/python /Users/macbook/
Documents/Py_mini_prj/guessing_game.py
Enter your guess: dog
Enter your guess: cat
You win!
(base) macbook@HuyTuan Py_mini_prj %

```

## B. Kết quả đạt được trong tuần 1:

- Ở công ty:

- + Làm quen với công ty và thống nhất lịch làm việc cố định tại văn phòng trong giai đoạn hè và vào năm học.
- + Thống nhất lịch báo cáo hàng tuần và nội dung báo cáo.
- + củng cố, nâng cao kiến thức về Python và giải quyết được những bài toán được giao, được chữa thêm những cách tối ưu hơn cho 1 bài toán.
- + Đảm bảo tiến độ tuần 1 trong kế hoạch training.
- Ở nhà:
  - + Tổng hợp và củng cố kiến thức về Python.
  - + Làm quen với 2 thư viện Numpy và Pandas.
  - + Cơ bản hoàn thành phần bài tập về nhà về 2 thư viện Numpy và Pandas mà anh Vỹ giao.

**C. Kế hoạch học tập và nghiên cứu trong tuần 2:**

- Đảm bảo tiến độ tuần 2 trong kế hoạch training.
- Tìm hiểu và hoàn thành thêm 1 khoá học về Python (Coursera) để vừa ôn lại và nắm vững hơn kiến thức về cả Python và 2 thư viện đã được làm quen ở tuần 1 là Numpy và Pandas.
- Chủ động tìm hiểu thêm syntax, mục đích sử dụng, thực hành các thư viện khác của Python như: Matplotlib, Seaborn, Opencv, Scikit learn.