Web Programming

Python

Prof. Josué Obregón

Department of Industrial Engineering- ITM

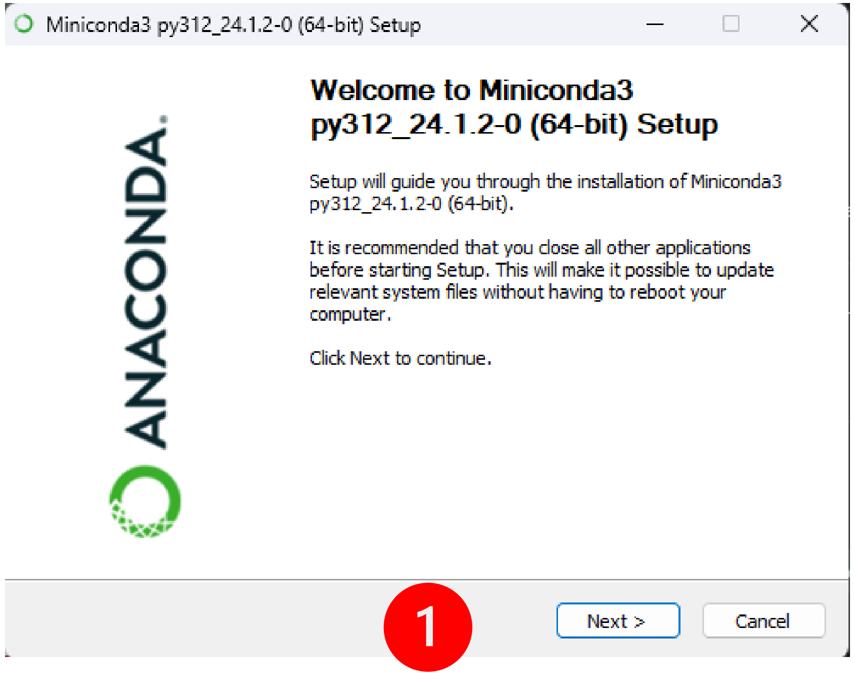
Seoul National University of Science and Technology

Python

Preparing our environment

1. Install miniconda

- Download and install miniconda



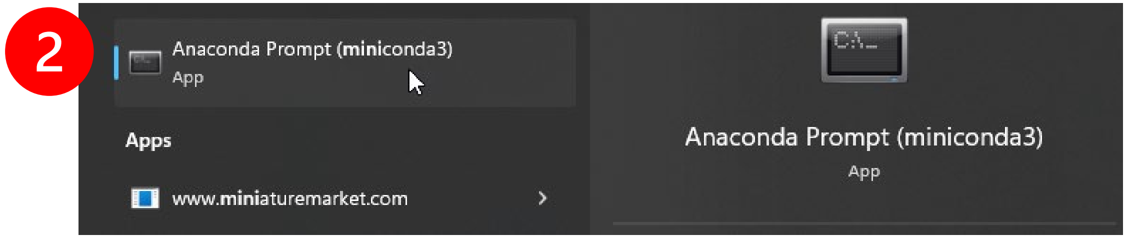
Miniconda3 py312_24.1.2-0 (64-bit) Setup

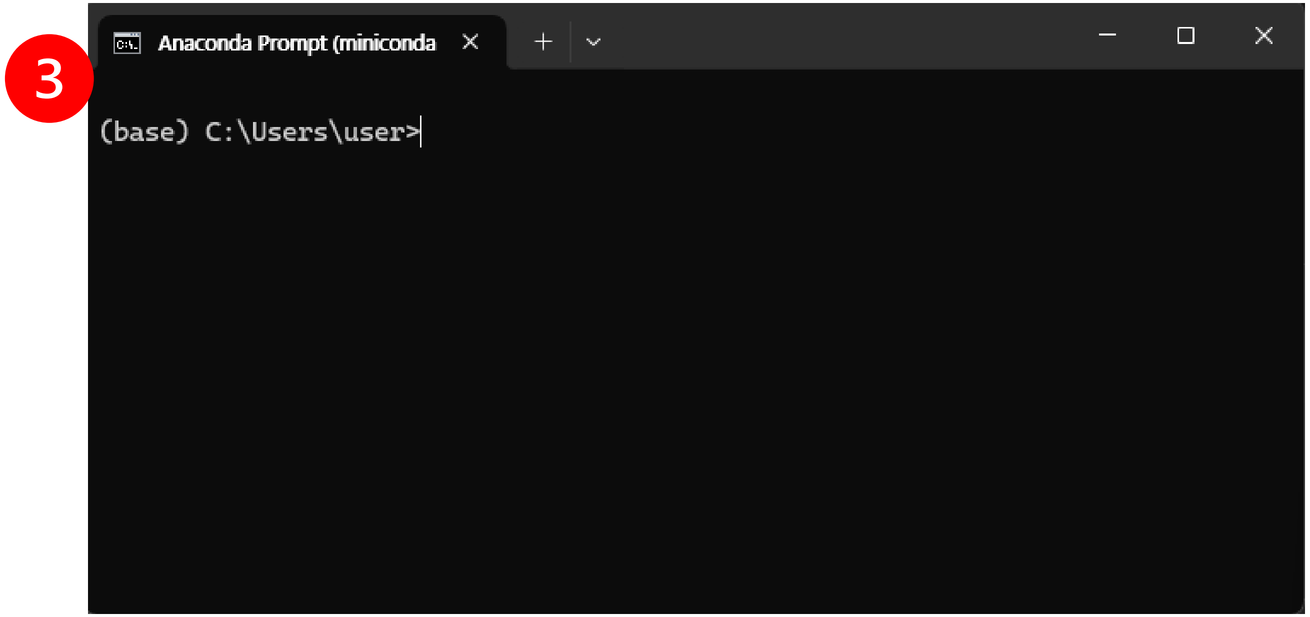
Welcome to Miniconda3 py312_24.1.2-0 (64-bit) Setup

Setup will guide you through the installation of Miniconda3 py312_24.1.2-0 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.





```
(base) C:\Users\user>
```

https://docs.anaconda.com/free/miniconda/index.html

Create a virtual environment using conda

1. Create a new virtual environment

```
$ conda create -n webprog python=3.11
```

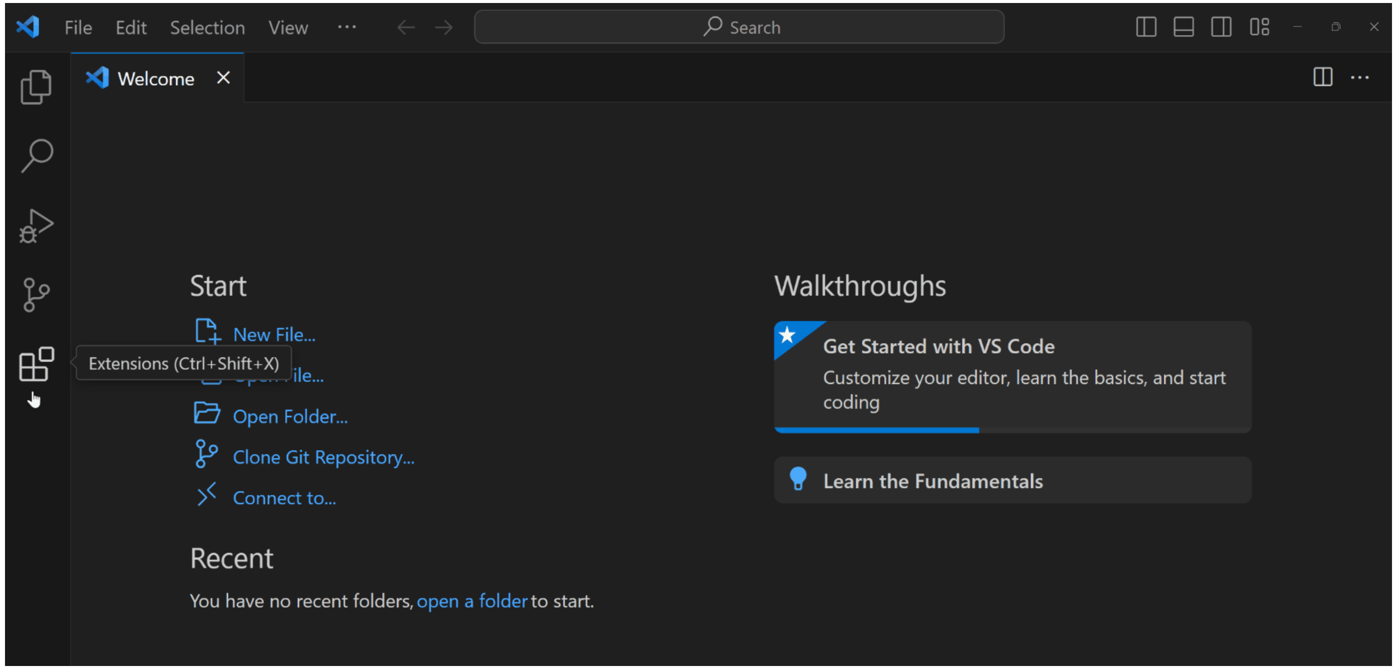
2. Activate the new virtual environment

```
$ conda activate webprog
```

Prepare VS Code for running Python

1. Install VS Code Python extension
2. Activate the Python profile of VS Code

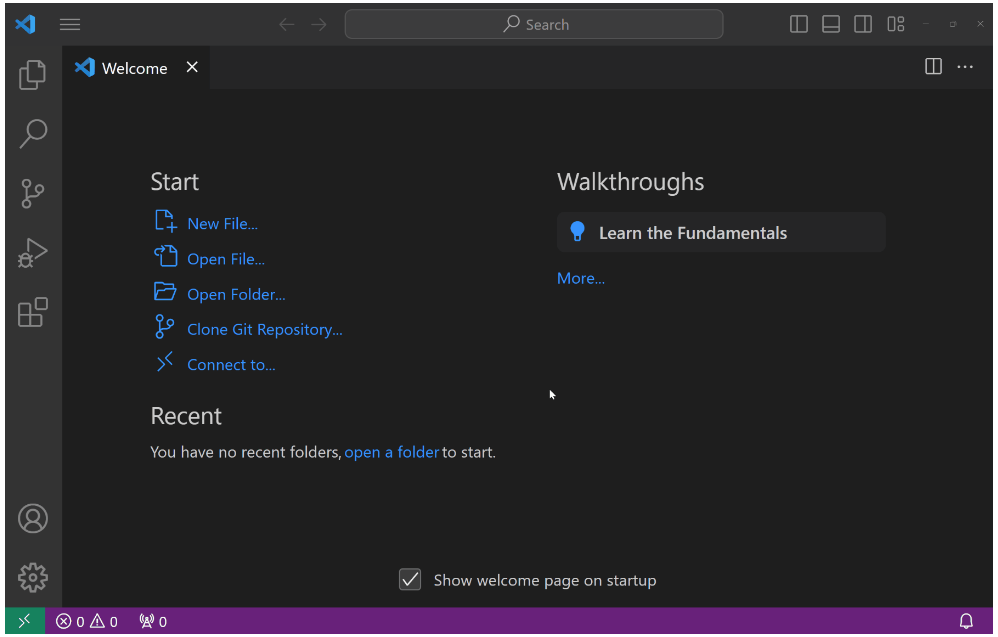
1



Customize your editor, learn the basics, and start coding

You have no recent folders, open a folder to start.

2



You have no recent folders, open a folder to start.

☑ Show welcome page on startup

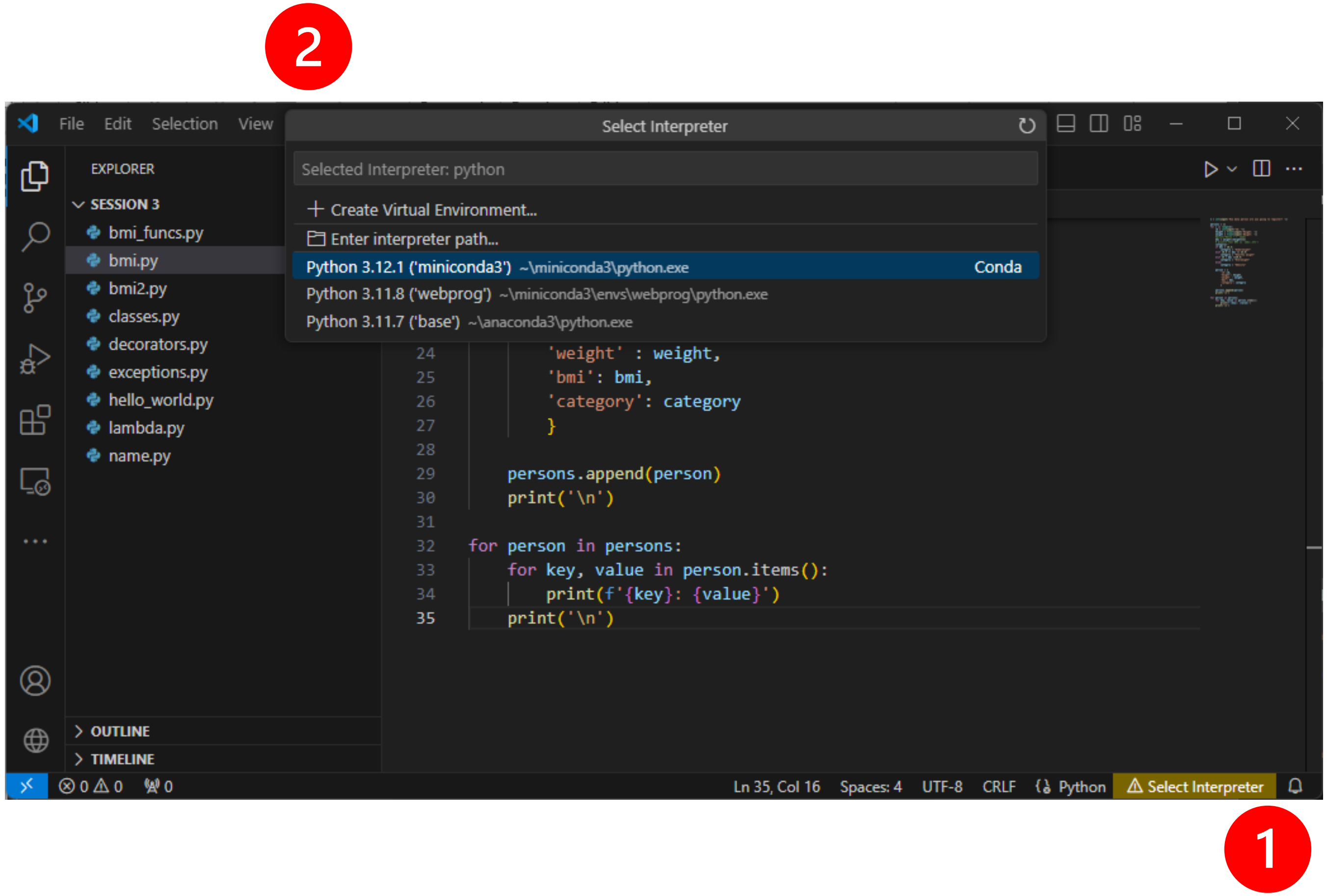
https://code.visualstudio.com/docs/python/python-quick-start

Prepare VS Code for running Python

- One of two ways

1. Click the select interpreter on the right bottom of the VS Code GUI
2. Access the Command Palette (*Ctrl+Shift+P*). and then write: "*Python: Select Interpreter*"

- Select the interpreter from the virtual environment that we created before



Selected Interpreter: python

+ Create Virtual Environment...

Enter interpreter path...

Python 3.12.1 ('miniconda3') ~\miniconda3\python.exe Conda

Python 3.11.8 ('webprog') ~\miniconda3\envs\webprog\python.exe

Python 3.11.7 ('base') ~\anaconda3\python.exe

```
            'weight' : weight,
            'bmi': bmi,
            'category': category
            }

        persons.append(person)
        print('\n')

    for person in persons:
        for key, value in person.items():
            print(f'{key}: {value}')
        print('\n')
```

Hello, World

```
print("Hello, world!")
```

Variables

```
a =   28
b =   1.5
c =   "Hello!"
d =   True
e =   None
```

Types

```
a =   28            # int
b =   1.5           # float
c =   "Hello!"      # str
d =   True          # bool
e =   None          # NoneType
```

Input

```
name = input("Name: ")
print(f"Hello, {name}") #fstrings
```

https://docs.python.org/3/tutorial/inputoutput.html#formatted-string-literals

Operators

Arithmetic	+	-	*	/	%	**	//
Assignment	=	+=	-=	*=	/=	%=	//=
Comparison	==	!=	>	<	>=	<=	
Logical	and	or	not				
Identity	is	is not					
Membership	in	not in					

Exercise 1

- Body Mass Index is a measure of body fat based on height and weight.

$$BMI = \frac{weight\ (kg)}{height^2(m)}$$

- Write a small program that:
  - Prompt the user to input their weight and height
  - Compute BMI
  - Display a message with the BMI of the user

Conditions

```
if x > 0:
    print("x is positive")
elif x < 0:
    print("x is negative")
else:
    print("x is 0")
```

Exercise 2

- For adults 20 years old and older, BMI is interpreted using standard weight status categories. These categories are the same for men and women of all body types and ages.

BMI	**Weight Status**
Below 18.5	Underweight
18.5 – 24.9	Healthy Weight
25.0 – 29.9	Overweight
30.0 and Above	Obesity

- Extend your previous program as follows :
  - Compare the computed BMI with the provided values and categories
  - Store the category in a String type variable
  - Display an additional message with the category of the input data

Data Structures

- `list` - sequence of mutable values → [1, 2, 3]
- `tuple` - sequence of immutable values → (1,2,3)
- `set` - collection of unique values → {1,2,3}
- `dict` - collection of key-value pairs {'a': 1, 'b': 2, 'c': 3}
- `...`

Loops

```
for iterator_var in sequence:
    statements(s)

for i in [0, 1, 2, 3, 4, 5]:
    print(i)

for i in range(5):
    print(i)

names = ['Leona', 'Annie', 'Diana', 'Mundo']
for name in names:
    print(name)
```

Functions and modules

functions.py

```
def addition(x, y, z):
  return x + y + z
```

program.py

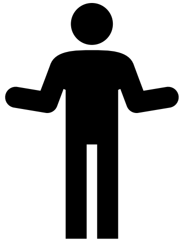
```
from functions import addition

a = 1
b = 2
c = 3

print(f'Result {addition(a,b,c)}')
```

Exercise 3

- Extend the BMI application by enabling the user to input more than one person information.
- Perform the following tasks:
    - Prompt the user to input the number *n* of people to record
    - Create loop that request the user to enter the following information:
        - id, weight and height
    - For each person, you should compute the BMI and decide a category according to the BMI
    - Store that information in a proper data structure
    - use functions and modules whenever possible(e.g., compute_bmi, get_category)
    - after the user enters the information of all persons, display the recorded information in the selected data structure

Object-Oriented Programming

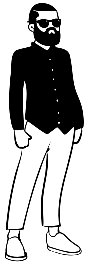
Class: template, blueprint



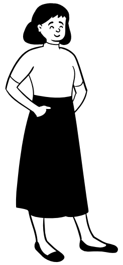
```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def greet(self):
    print(f"Hello, my name is {self.name}.")
```

Objects: concrete instances or examples of classes

```
person = Person("Frank", 28)
person.greet()
>>> Hello, my name is Frank
```



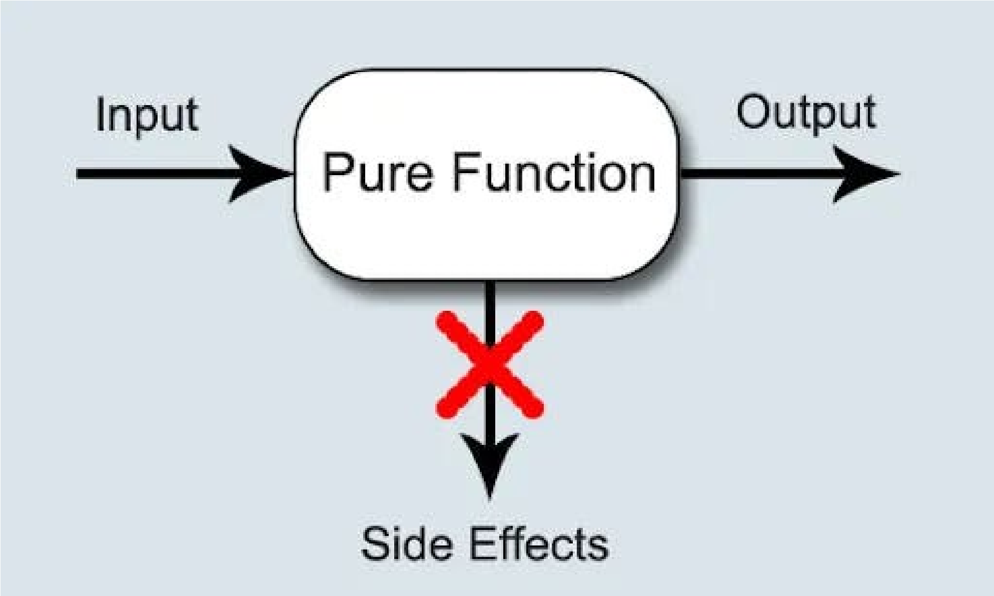
```
person = Person("Matt", 30)
person.greet()
>>> Hello, my name is Matt
```



```
person = Person("Alice", 30)
person.greet()
>>> Hello, my name is Alice
```

Functional programming

- Functional programming is a declarative programming paradigm style where one applies pure functions in sequence to solve complex problems.
- First-class functions in functional programming are treated as data type variables and can be used like any other variables.
- These first-class variables can be passed to functions as parameters, or stored in data structures.



Decorators

```
def announce(f):
   def wrapper():
      print("About to run the function...")
      f()
      print("Done with the function.")
   return wrapper

@announce
def hello():
    print("Hello world!")
```

Lambda Functions

A lambda function is a small anonymous function.

```
x = lambda a : a + 10
print(x(5))
```

Exceptions

```
try:
    x = int(input("Please enter a number: "))
except ValueError:
    print("Oops!  That was no valid number.")
```

Exercise 4

- Add the object-oriented design to your BMI program
- Perform the following tasks:
  - Define your Patient class
    - Attributes: id, weight, height, bmi and weight_category
    - Behavior: constructor, compute bmi, get weight category, display patient information)
    - Create an @log decorator that print in the console "Function FUNCTION_NAME was executed" (hint: you can get the function name accessing the __name__ special variable)
    - Decorate your compute bmi and get weight category functions
  - After creating your class, prompt the user to input the number *n* of patients to record
  - Create loop that request the user to enter the following information for each patient:
    - id, weight and height
    - Handle ValueError exception
  - Store that information in a Patient object
  - after the user enters the information of all persons, display the recorded information of the patients

Web Programming

with Python and JavaScript