COMPUTER PROGRAMMING LABORATORY
Experiment # 1:
Introduction

OBJECTIVES

The main purpose of this experiment is to introduce you to Integrated Development Environments (IDE). In this experiment, steps for downloading and installation of an IDE are given.

INFORMATION

IDEs are used to develop, compile and run a computer program. Spyder is a well-known, free, fast, and simple IDE for developing Python codes. In order to use the IDE, you must follow the steps that are given below.

Step 1: Download Python 3.9.10. You can use the following website

https://www.python.org/downloads/windows/

Step 2: Then, install the downloaded file.

Step 3: Download WinPython 3.9.10 You can use the following website

https://winpython.github.io/

Step 4: After installation, run the Spyder.

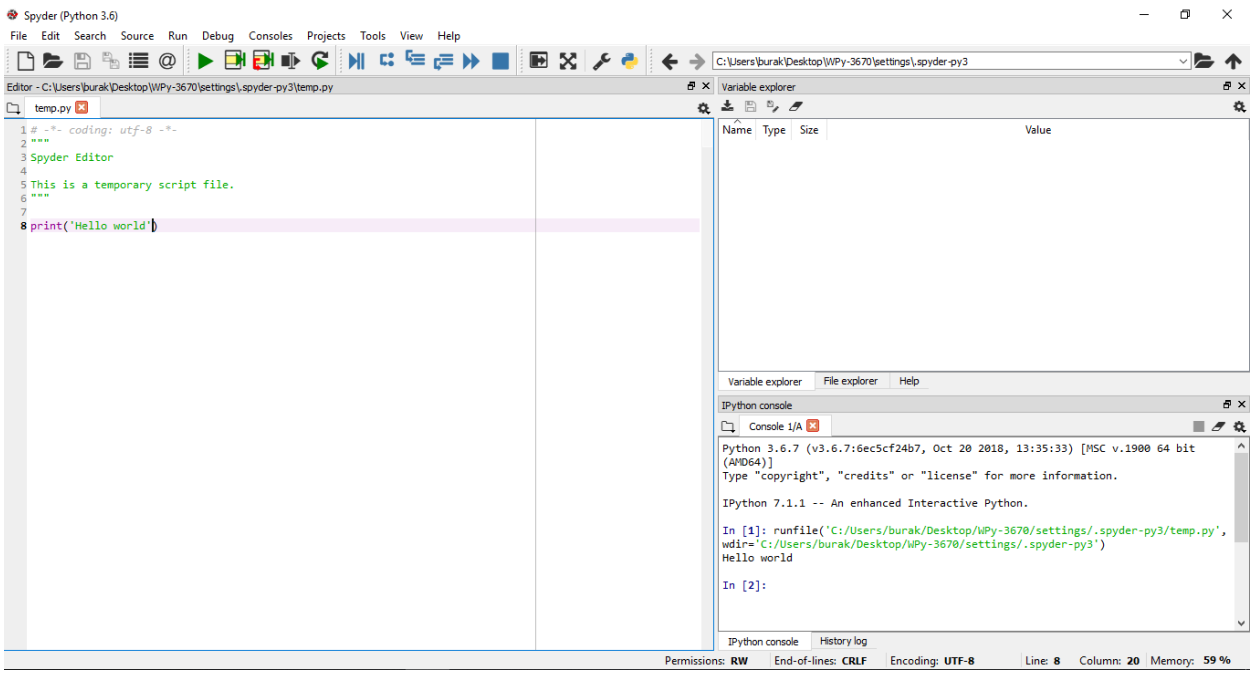


Fig. 1 Spyder IDE

E-mail : burakaleci@gmail.com

TOPICS

- **Introduction**

```
IPython console
Console 1/A
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.1900 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.1.1 -- An enhanced Interactive Python.

In [1]: print('Python programming is fun!')
Python programming is fun!

In [2]:
```

Variable explorer

Name	Type	Size	Value
lenght	int	1	5
width	int	1	10

Variable explorer | File explorer | Help

```
IPython console
Console 1/A
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.1900 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.1.1 -- An enhanced Interactive Python.

In [1]: width=10

In [2]: lenght=5

In [3]: print(width)
10

In [4]: print(lenght)
5

In [5]:
```

- **Decision and Repetition statements**

```python
# This program determines whether a bank customer
# qualifies for a loan.

# Constants for minimum salary and minimum
# years on the job
MIN_SALARY = 30000.0
MIN_YEARS = 2


# Get the customer's annual salary.
salary = float(input('Enter your annual salary: '))

# Get the number of years on the current job.
years_on_job = int(input('Enter years employed: '))


# Determine whether the customer qualifies.
if salary >= MIN_SALARY:
    if years_on_job >= MIN_YEARS:
        print('You qualify for the loan.')
    else:
        print('You must have been employed','for at least', MIN_YEARS,'years to qualify.')
else:
    print('You must earn at least $',format(MIN_SALARY, ',.2f'),' per year to qualify.', sep='')

```

```python
# Get a test score.
score = int(input('Enter a test score: '))

# Make sure it is not less than 0.
while score < 0:
    print('ERROR: The score cannot be negative.')
    score = int(input('Enter the correct score: '))
```

```python
# This program uses a loop to display a
# table showing the numbers 1 through 10
# and their squares.

# Print the table headings.
print('Number\tSquare')
print('--------------')

# Print the numbers 1 through 10
# and their squares.
for number in range(1, 11):
    square = number**2
    print(number, '\t', square)
```

Program Output

```
Number  Square
--------------
1        1
2        4
3        9
4        16
5        25
6        36
7        49
8        64
9        81
10       100
```

- **Functions**

```python
# This program demonstrates a function that accepts
# two arguments.

def main():
    print('The sum of 12 and 45 is')
    show_sum(12, 45)

# The show_sum function accepts two arguments
# and displays their sum.
def show_sum(num1, num2):
    result = num1 + num2
    print(result)

# Call the main function.
main()
```

- **Python built-in data types**

```python
# The get_scores function gets a series of test
# scores from the user and stores them in a list.
# A reference to the list is returned.
def get_scores():
    # Create an empty list.
    test_scores = []

    # Create a variable to control the loop.
    again = 'y'

    # Get the scores from the user and add them to
    # the list.
    while again == 'y':
        # Get a score and add it to the list.
        value = float(input('Enter a test score: '))
        test_scores.append(value)

        # Want to do this again?
        print('Do you want to add another score?')
        again = input('y = yes, anything else = no: ')
        print()

    # Return the list.
    return test_scores
```

```
>>> test_scores = { 'Kayla' : [88, 92, 100], Enter
                    'Luis' : [95, 74, 81], Enter
                    'Sophie' : [72, 88, 91], Enter
                    'Ethan' : [70, 75, 78] } Enter
>>> test_scores Enter
{'Kayla': [88, 92, 100], 'Sophie': [72, 88, 91], 'Ethan': [70, 75, 78],
'Luis': [95, 74, 81]}
>>> test_scores['Sophie'] Enter
[72, 88, 91]
>>> kayla_scores = test_scores['Kayla'] Enter
>>> print(kayla_scores) Enter
[88, 92, 100]
```

- **Libraries**

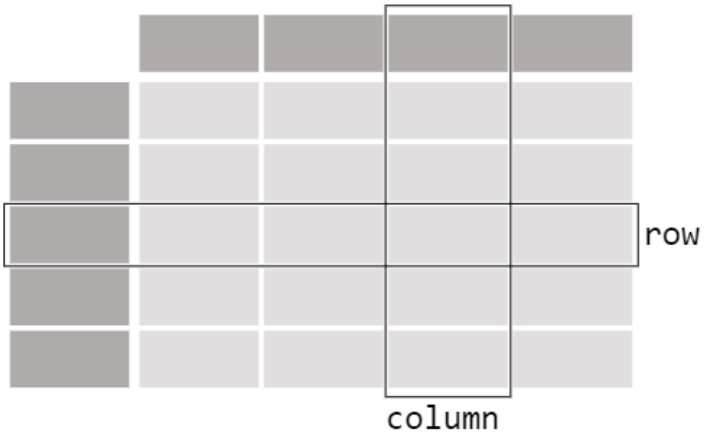
```
In [1]: import numpy as np

In [2]: a = np.array([1, 2, 3, 4, 5, 6])

In [3]: a.dtype
Out[3]: dtype('int32')

In [4]: a.shape
Out[4]: (6,)

In [5]: b=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])

In [6]: b.dtype
Out[6]: dtype('int32')

In [7]: b.shape
Out[7]: (3, 4)

In [8]: print(b[0])
[1 2 3 4]

In [9]: print(a[4])
5
```

```
In [1]: import numpy as np

In [2]: b=np.array([[7,1,4,12],[2,9,8,3]])

In [3]: b.shape
Out[3]: (2, 4)

In [4]: np.sort(b,axis=0,kind='mergesort')
Out[4]:
array([[ 2,  1,  4,  3],
       [ 7,  9,  8, 12]])

In [5]: np.sort(b,axis=1,kind='heapsort')
Out[5]:
array([[ 1,  4,  7, 12],
       [ 2,  3,  8,  9]])

In [6]: np.sort(b,axis=0,kind='quicksort')[::-1]
Out[6]:
array([[ 7,  9,  8, 12],
       [ 2,  1,  4,  3]])

In [7]: np.sort(b,axis=1,kind='quicksort')[::-1]
Out[7]:
array([[ 2,  3,  8,  9],
       [ 1,  4,  7, 12]])
```

```
In [25]: divisible_by_2 = a[a%2==0]

In [26]: print(divisible_by_2)
[ 2  4  6  8 10 12]

In [27]: c = a[(a > 2) & (a < 11)]

In [28]: print(c)
[ 3  4  5  6  7  8  9 10]

In [29]: five_up = (a > 5) | (a == 5)

In [30]: print(five_up)
[[False False False False]
 [ True  True  True  True]
 [ True  True  True  True]]
```

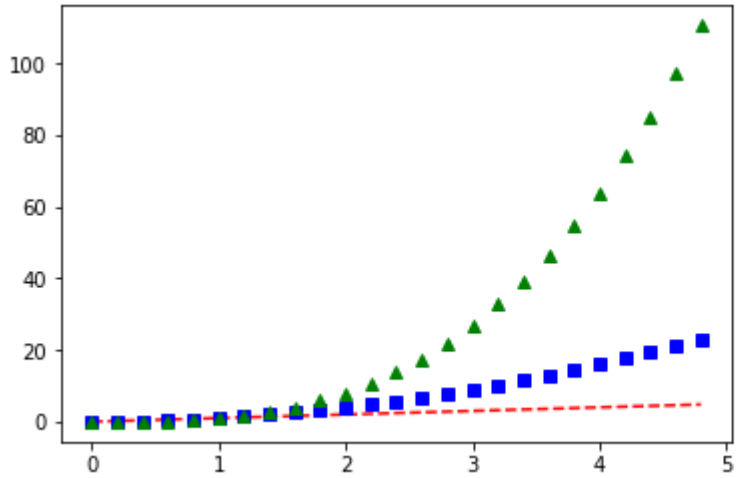
```
In [97]: x=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])

In [98]: a1 = x.flatten()

In [99]: a1[0] = 99

In [100]: print(x)
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]

In [101]: print(a1)
[99  2  3  4  5  6  7  8  9 10 11 12]

In [102]: a2 = x.ravel()

In [103]: a2[0] = 98

In [104]: print(x)
[[98  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]

In [105]: print(a2)
[98  2  3  4  5  6  7  8  9 10 11 12]
```

DataFrame

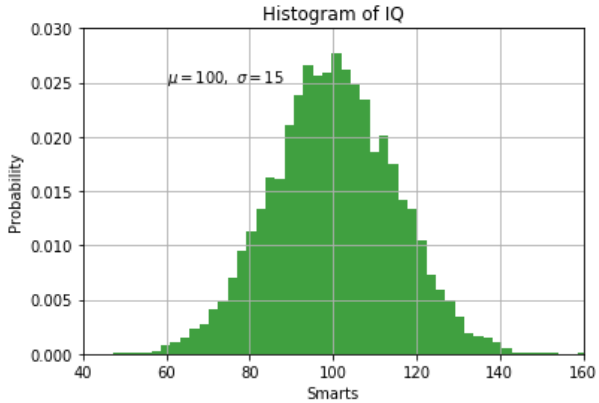



```
In [1]: import pandas as pd

In [2]: df=pd.DataFrame({"Name": ["Braund, Mr. Owen Harris",
   ...: "Allen, Mr. William Henry",
   ...: "Bonnell, Miss. Elizabeth"],
   ...: "Age": [22, 35, 58],
   ...: "Sex": ["male", "male", "female"]}
   ...: )

In [3]: df
Out[3]:
                        Name  Age     Sex
0   Braund, Mr. Owen Harris   22    male
1  Allen, Mr. William Henry   35    male
2  Bonnell, Miss. Elizabeth   58  female
```

```
In [17]: import numpy as np
    ...:
    ...: # evenly sampled time at 200ms intervals
    ...: t = np.arange(0., 5., 0.2)
    ...:
    ...: # red dashes, blue squares and green triangles
    ...: plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
    ...: plt.show()
```




```
In [25]: mu, sigma = 100, 15
    ...: x = mu + sigma * np.random.randn(10000)
    ...: # the histogram of the data
    ...: n, bins, patches = plt.hist(x, 50, density=1, facecolor='g',
alpha=0.75)
    ...: plt.xlabel('Smarts')
    ...: plt.ylabel('Probability')
    ...: plt.title('Histogram of IQ')
    ...: plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
    ...: plt.axis([40, 160, 0, 0.03])
    ...: plt.grid(True)
    ...: plt.show()
```



- **Classes**

```python
import random

# The Coin class simulates a coin that can be flipped.
class Coin:

    # The __init__ method initializes the __sideup data attribute with 'Heads'.
    def __init__(self):
        self.__sideup = 'Heads'

    # The toss method generates a random number in the range of 0 through 1.
    # If the number is 0, then sideup is set to 'Heads'.
    # Otherwise, sideup is set to 'Tails'.
    def toss(self):
        if random.randint(0, 1) == 0:
            self.__sideup = 'Heads'
        else:
            self.__sideup = 'Tails'

    # The get_sideup method returns the value referenced by sideup.
    def get_sideup(self):
        return self.__sideup

# The main function.
def main():
    # Create an object from the Coin class.
    my_coin = Coin()

    # Display the side of the coin that is facing up.
    print('This side is up:', my_coin.get_sideup())

    # Toss the coin.
    print('I am going to toss the coin ten times:')
    for count in range(10):
        my_coin.toss()
        print(my_coin.get_sideup())

# Call the main function.
main()
```

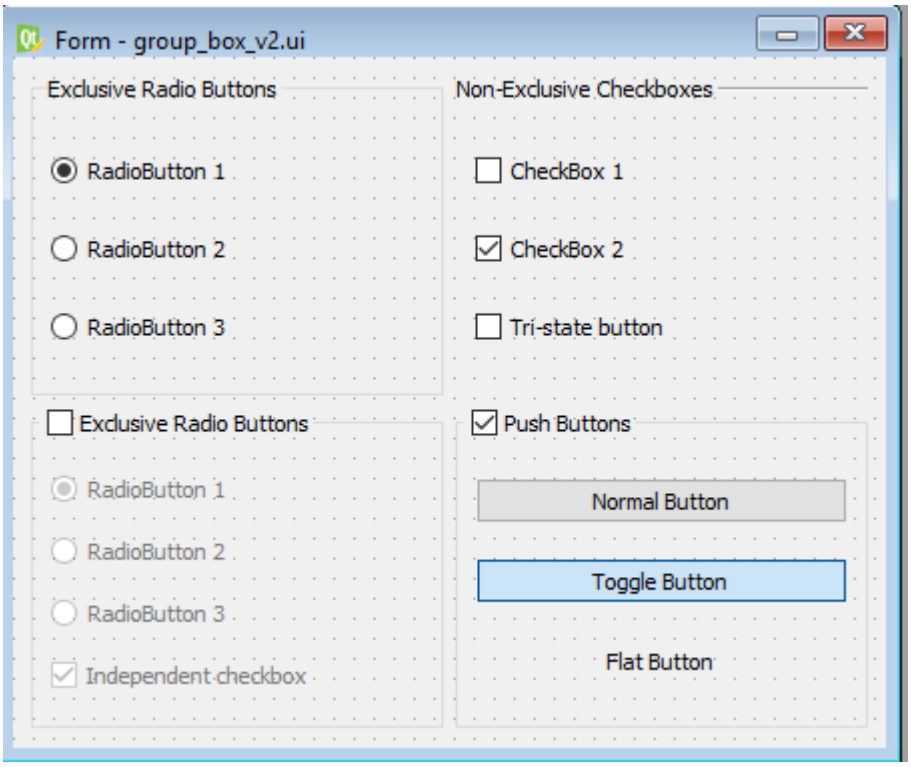
- **Inheritance and Polymorphism**

```python
class Automobile:

    def __init__(self,make, model, mileage, price):
        self.__make = make
        self.__model = model
        self.__mileage = mileage
        self.__price = price

    def set_make(self,make):
        self.__make = make

    def set_model(self,model):
        self.__model = model

    def set_mileage(self,mileage):
        self.__mileage = mileage

    def set_price(self,price):
        self.__price = price

    def get_make(self):
        return self.__make

    def get_model(self):
        return self.__model

    def get_mileage(self):
        return self.__mileage

    def get_price(self):
        return self.__price
```

```python
class Car(Automobile):

    def __init__(self,make, model, mileage, price,doors):

        Automobile.__init__(self,make,model,mileage,price)

        self.__doors = doors

    def set_doors(self,doors):
        self.__doors = doors

    def get_doors(self):
        return self.__doors
```

```python
class Truck(Automobile):

    def __init__(self,make, model, mileage, price,drive_type):

        Automobile.__init__(self,make,model,mileage,price)

        self.__drive_type = drive_type

    def set_drive_type(self,drive_type):
        self.__drive_type = drive_type

    def get_drive_type(self):
        return self.__drive_type
```

- **QT**



- **File Processing and Exception Handling**

```python
# This program allows the user to search the coffee.txt file for records matching a description.

def main():
    # Create a bool variable to use as a flag.
    found = False

    # Get the search value.
    search = input('Enter a description to search for: ')

    # Open the coffee.txt file.
    coffee_file = open('coffee.txt', 'r')

    # Read the first record's description field.
    descr = coffee_file.readline()

    # Read the rest of the file.
    while descr != '':
        # Read the quantity field.
        qty = float(coffee_file.readline())

        # Strip the \n from the description.
        descr = descr.rstrip('\n')

        # Determine whether this record matches the search value.
        if descr == search:
            # Display the record.
            print('Description:', descr)
            print('Quantity:', qty)
            print()
            # Set the found flag to True.
            found = True

        # Read the next description.
        descr = coffee_file.readline()

    # Close the file.
    coffee_file.close()

    # If the search value was not found in the file display a message.
    if not found:
        print('That item was not found in the file.')

# Call the main function.
main()
```

```python
# This program displays the contents
# of a file.

def main():
    # Get the name of a file.
    filename = input('Enter a filename: ')

    try:
        # Open the file.
        infile = open(filename, 'r')

        # Read the file's contents.
        contents = infile.read()

        # Display the file's contents.
        print(contents)

        # Close the file.
        infile.close()
    except IOError:
        print('An error occurred trying to read')
        print('the file', filename)

# Call the main function.
main()
```