HW 1: Basic Python Programming

CPE232 Data Models

WISIT SUWANNAO 67070501042

Description : For this homework, please write your code in the section we prepared for you. If you need to leave a comment or message for the TA, use the # symbol in the code section. You may also create a text cell in the Jupyter Notebook to provide clarification.

Good luck with Python!

1. Basic usage

> John Doe is a 35 years-old software engineer who earns ฿35000.00 a month.

Create and assign variables to store this person's information (name, age, position and salary).

In [1]:
```
name = "John Doe"
age = 35
position = "software engineer"
salary = 35000.00
```

What is the type of each variables?

In [2]:
```
print(f"Type of name: {type(name)}")
print(f"Type of age: {type(age)}")
print(f"Type of position: {type(position)}")
print(f"Type of salary: {type(salary)}")
```

```
Type of name: <class 'str'>
Type of age: <class 'int'>
Type of position: <class 'str'>
Type of salary: <class 'float'>
```

The manager decides to give John a 7% raise. Update his salary.

In [3]:
```
salary = salary * 1.07
```

Prints his information again with his new salary.

In [4]:
```
print(f"{name} is a {age} years-old {position} who earns ฿{salary:.2f} a month.")
```

```
John Doe is a 35 years-old software engineer who earns ฿37450.00 a month.
```

2. Variable and Expression

2.1 Write a code to convert temperature unit from celcius to other units and then prints out

In [5]:
```
C = 34.5
```

Fahrenheit

$$\frac{C}{5} = \frac{F-32}{9}$$

In [6]:
```
F = (C / 5) * 9 + 32
print(f'{C}°C is {F}°F')
```

```
34.5°C is 94.1°F
```

Kelvin

$$K = C + 273.15$$

In [7]:
```
K = C + 273.15
print(f'{C}°C is {K}K')
```

```
34.5°C is 307.65K
```

Rømer

$$Ro = \frac{C \times 21}{40} + 7.5$$

In [8]:
```
Ro = (C * 21) / 40 + 7.5
print(f'{C}°C is {Ro}°Ro')
```

```
34.5°C is 25.6125°Ro
```

2.2 Write code to read the input for weight (kg) and height (cm), then print out the BMI (Body Mass Index).

$$BMI = \frac{kg}{m^2}$$

In [9]:
```
weight = int(input("Weight (kg):"))
height = int(input("Height (cm):"))

height_in_meters = height / 100
BMI = weight / (height_in_meters ** 2)
print(f"Your BMI is: {BMI:.2f}")
```

```
Weight (kg):73
Height (cm):176
Your BMI is: 23.57
```

3. Multi-item variables

List

In [10]:
```
names = ['Thomas', 'Kate', 'Mike', 'Amelia', 'James', 'Megan']
```

Create new variable call `new_name` which takes input name of the user.

In [11]:
```
new_name = input('Enter your name: ')
```

```
Enter your name: Palapluem
```

Insert `new_name` into `names` list.

In [12]:
```
names.append(new_name)
```

Print your name from the list `by index`.

In [13]:
```
print(names[-1])
```

```
Palapluem
```

Merge `another_names` into `names`.

In [14]:
```
another_names = ['Peter', 'Steve', 'Sam', 'Charlotte']
```

In [15]:
```
names.extend(another_names)
print(names)
```

```
['Thomas', 'Kate', 'Mike', 'Amelia', 'James', 'Megan', 'Palapluem', 'Peter', 'Steve', 'Sam', 'Charlotte']
```

Change `Amelia`'s name to `Amy`

In [16]:
```
index = names.index('Amelia')
names[index] = 'Amy'
print(names)
```

```
['Thomas', 'Kate', 'Mike', 'Amy', 'James', 'Megan', 'Palapluem', 'Peter', 'Steve', 'Sam', 'Charlotte']
```

Dictionary

In [17]:
```
capital_city = {'England':'London',
                'Spain':'Madrid',
                'Japan':'Tokyo',
                'Australia':'Sydney',
                'Germany':'Berlin',
               }
```

Add a record `Thailand` and it's capital city to this dictionary

In [18]:
```
capital_city['Thailand'] = 'Bangkok'
print(capital_city)
```

```
{'England': 'London', 'Spain': 'Madrid', 'Japan': 'Tokyo', 'Australia': 'Sydney', 'Germany': 'Berlin', 'Thailand': 'Bangkok'}
```

You may notice that the capital city of `Australia` is wrong. It should be `Canberra`. Correct this mistake.

In [19]:
```
capital_city['Australia'] = 'Canberra'
print(capital_city)
```

```
{'England': 'London', 'Spain': 'Madrid', 'Japan': 'Tokyo', 'Australia': 'Canberra', 'Germany': 'Berlin', 'Thailand': 'Bangkok'}
```

4. Control Flows and conditional statements

if...elif...else

1. Write a program that takes two numbers as input from the user, called A and B. Then, print the number that is greater.

Example:

- If `A = 25` and `B = 15`, the program should print: "A = 25"
- If `A = 10` and `B = 22`, the program should print: "B = 22"

In [20]:
```
A = int(input("A: "))
B = int(input("B: "))
# Write your code here
if A > B:
```

```
    print(f"A = {A}")
elif B > A:
    print(f"B = {B}")
else:
    print(f"A = {A} and B = {B}. They are equal.")
```

```
A: 25
B: 15
A = 25
```

2. Define a variable to get input age from user

In [21]:
```
age = int(input("Enter age: "))
```

```
Enter age: 20
```

Write a series of if...elif...else statement that categorize input age into following groups:

> Babies: 0-2 years old
> Children: 3-12 years old
> Teenager: 13-19 years old
> Young Adults: 20-29 years old
> Middle-aged Adults: 30-45 years old
> Old Adult: 46-59 years old
> Elderly: Above 60 years old

In [22]:
```
# Write your code here
if 0 <= age <= 2:
    print("Babies")
elif 3 <= age <= 12:
    print("Children")
elif 13 <= age <= 19:
    print("Teenager")
elif 20 <= age <= 29:
    print("Young Adults")
elif 30 <= age <= 45:
    print("Middle-aged Adults")
elif 46 <= age <= 59:
```

```
    print("Old Adult")
elif age >= 60:
    print("Elderly")
else:
    print("Invalid age")
```

```
Young Adults
```

Looping

1. Write a code to create a multiplication table of an input number (multiplier from 1-12).

In [23]:
```
num = int(input("Enter a number for the multiplication table: "))
for i in range(1, 13):
    print(f"{num} x {i} = {num * i}")
```

```
Enter a number for the multiplication table: 12
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
12 x 11 = 132
12 x 12 = 144
```

2. Write a code that construct the following pattern.

```
input: 5
output:
*
**
***
```

```
****
*****
```

In [24]:
```
input_num = int(input("Enter a number for the pattern: "))
for i in range(1, input_num + 1):
    print("*".ljust(i, '*'))
```

```
Enter a number for the pattern: 10
*
**
***
****
*****
******
*******
********
*********
**********
```

3. Creates a loop to print `I love <programming language>!` except for Assembly, print `Not you, Assembly`.

In [25]:
```
languages = ['C/C++', 'Python', 'R', 'Java', 'SQLs', 'Assembly', 'Go', 'Rust', 'Kotlin']
```

In [26]:
```
for lang in languages:
    if lang == 'Assembly':
        print('Not you, Assembly')
    else:
        print(f'I love {lang}!')
```

```
I love C/C++!
I love Python!
I love R!
I love Java!
I love SQLs!
Not you, Assembly
I love Go!
I love Rust!
I love Kotlin!
```

4. Write a code to print every number from 1 to 25 except the one that is divisible by 3.

In [27]:
```
for num in range(1, 26):
    if num % 3 != 0:
        print(num)
```

```
1
2
4
5
7
8
10
11
13
14
16
17
19
20
22
23
25
```

5. Write a code that finds the number that is divisible by 7 in a given range.

In [28]:
```
lower_bound = 1
upper_bound = 100
divisor = 7

result = []
```

In [29]:
```
for num in range(lower_bound, upper_bound + 1):
    if num % divisor == 0:
        result.append(num)
print(result)
```

```
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98]
```

In [30]:
```
# Write your code here
```

6. Write a code that construct the following pattern.

```
input: 5
output:
*#####
**####
***###
****##
*****#

input: 10
output:
*##########
**#########
***########
****#######
*****######
******#####
*******####
********###
*********##
**********#
```

In [31]:
```
input_num = int(input("Enter a number for the pattern: "))
for i in range(1, input_num + 1):
    print('*' * i + '#' * (input_num - i))
```

```
Enter a number for the pattern: 10
*#########
**########
***#######
****######
*****#####
******####
*******###
********##
*********#
**********
```

5. Functions

1. Define a function `average` that takes *list of numbers* and calculate the mean of input. It should look like this:

> average([1,2,3,4]) output: 2.5

In [32]:
```
def average(numbers):
    return sum(numbers) / len(numbers)
```

2. Define a function `sumproduct` that takes 2 *equal-sized* lists and calculate sum of the products of two lists.
It should look like this:

> sumproduct([1,2,3],[4,5,6])
> output: 32

(1 * 4) + (2 * 5) + (3 * 6) = 32

In [33]:
```
def sumproduct(list1, list2):
    if len(list1) != len(list2):
        raise ValueError("Lists must be of equal size")
    total_sum = 0
    for i in range(len(list1)):
        total_sum += list1[i] * list2[i]
    return total_sum
```

3. Define a function `fibonacci` that returns Fibonacci number at `n` position.
A Fibonacci number at position `n` is defined by `F(n) = F(n-1) + F(n-2)`. Where `F(0) = 0` and `F(1) = 1`
Example: `0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...`

In [34]:
```
def fibonacci(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
```

```
    else:
        a, b = 0, 1
        for _ in range(2, n + 1):
            a, b = b, a + b
        return b
```

4. Define a function `is_palindrome` that takes input string and check whether it is a palindrome or not.
A string is a palindrome if it reads the same forward and backwards.

Example: `madam` , `race car` , `borrow or rob` , `amore roma` , `never odd or even`
Do not consider whitespace. Use `str.replace(' ', '')` to remove whitespace from your string.
Case-insensitive. You can turn everything into lower or uppercase using `str.lower()` or `str.upper()`

Hint: you can reverse the string using `[::-1]` slice.

In [35]:
```
str1 = "radar" # palindrome
str2 = "rotator" # palindrome
str3 = "lemon" # not palindrome
```

In [36]:
```
def is_palindrome(text):
    processed_text = text.replace(' ', '').lower()
    return processed_text == processed_text[::-1]

print(f"'{str1}' is a palindrome: {is_palindrome(str1)}")
print(f"'{str2}' is a palindrome: {is_palindrome(str2)}")
print(f"'{str3}' is a palindrome: {is_palindrome(str3)}")
```

```
'radar' is a palindrome: True
'rotator' is a palindrome: True
'lemon' is a palindrome: False
```

5. An `anagram` is a word or phrase formed by rearranging the letters of a different word or phrase.
Define a function `is_anagram` that takes in 2 strings and check whether it is possible to compose a second string using letters in the first string or not.

Example: `Tom Marvolo Riddle` can be rearraged into `I am Lord Voldermort`
`Meaning of Life` can be rearranged into `Engine of a Film`

Do not consider whitespace. Use `str.replace(' ', '')` to remove whitespace from your string.
Case-insensitive. You can turn everything into lower or uppercase using `str.lower()` or `str.upper()`
Returns only `True` of `False`

In [37]:
```python
def is_anagram(str1, str2):
    str1 = str1.replace(' ', '').lower()
    str2 = str2.replace(' ', '').lower()
    return sorted(str1) == sorted(str2)

str1_anagram = "Tom Marrvolo Riddle"
str2_anagram = "I am Lord Voldemort"
print(f"'{str1_anagram}' and '{str2_anagram}' are anagrams: {is_anagram(str1_anagram, str2_anagram)}")

str1_anagram_2 = "Meaning of Life"
str2_anagram_2 = "Engine of a Film"
print(f"'{str1_anagram_2}' and '{str2_anagram_2}' are anagrams: {is_anagram(str1_anagram_2, str2_anagram_2)}")
```

```
'Tom Marrvolo Riddle' and 'I am Lord Voldemort' are anagrams: False
'Meaning of Life' and 'Engine of a Film' are anagrams: True
```
