HW 1: Basic Python Programming

CPE232 Data Models

---

1. Basic usage

John Doe is a 29 years-old system engineer who earns ฿41500.00 a month.

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

In [120…
```python
# Write your code here


tuple = ("John Doe", 29, "system engineer", 41500.00)
print(tuple)
```
```
('John Doe', 29, 'system engineer', 41500.0)
```

What is the type of each variables?

In [121…
```python
# Write your code here
# Type of each element in the tuple
for i in range(len(tuple)):
    print(f"{tuple[i]} is {type(tuple[i])}")
```
```
John Doe is <class 'str'>
29 is <class 'int'>
system engineer is <class 'str'>
41500.0 is <class 'float'>
```

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

In [122…
```python
# Write your code here
tuple = (tuple[0], tuple[1], tuple[2], tuple[3] * 107 / 100)
```

Prints his information again with his new salary.

In [123…
```python
# Write your code here
print(tuple)
```
```
('John Doe', 29, 'system engineer', 44405.0)
```

Now, he decides to resign. Delete his information from the system.

In [124…
```python
# Write your code here
del tuple
print(tuple)
```

```
<class 'tuple'>
```

2. Variable and Expression

2.1 Write a code to convert temperature unit from celcius to other units

In [125…

```
C = 34.5
```

Fahrenheit

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

In [126…

```
F = C * 9 / 5 + 32
print(F)
```

```
94.1
```

Kelvin

$K = C + 273.15$

In [127…

```
K = C + 273.15
print(K)
```

```
307.65
```

Rømer

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

In [128…

```
Ro = C * 21 / 40 + 7.5
print(Ro)
```

```
25.6125
```

3. Multi-item variables

List

In [13]:

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

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

In [14]:

```
new_name = input('Enter your name: ')
```

Insert `new_name` into `names` list.

In [15]:

```
# Write your code here
names.append(new_name)
print(names)
```

```
['Thomas', 'Kate', 'Mike', 'Amelia', 'James', 'Megan', 'Kamin']
```

Select your name from the list

In [18]:
```
# Write your code here
print(names[len(names) - 1])
```

```
Kamin
```

Merge `another_names` into `names`.

In [133…
```
another_names = ['Peter', 'Steve', 'Sam', 'Charlotte']
```

In [134…
```
# Write your code here
names.extend(another_names)
print(names)
```

```
['Thomas', 'Kate', 'Mike', 'Amelia', 'James', 'Megan', 'Kamin', 'Peter', 'St
eve', 'Sam', 'Charlotte']
```

Change `Amelia`'s name to `Amy`

In [135…
```
# Write your code here
# Change the name of the person in the list
target_name = "Amelia"
new_name = "Amy"
if target_name in names:
    index = names.index(target_name)
    names[index] = new_name

print(names)
```

```
['Thomas', 'Kate', 'Mike', 'Amy', 'James', 'Megan', 'Kamin', 'Peter', 'Stev
e', 'Sam', 'Charlotte']
```

Dictionary

In [136…
```
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 [137…
```
# Write your code here
# Add a new record to the dictionary
capital_city['Thailand'] = 'Bangkok'
print(capital_city)
```

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

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

In [138…

```
# Write your code here
# Change the capital city of Australia to Canberra
capital_city['Australia'] = 'Canberra'
print(capital_city)
```

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

4. Control Flows and conditional statements

if...elif...else

1. Define a variable to get input age from user.

In [139…

```
age = int(input('Enter your age: '))
print(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 [155…

```
# Write your code here
if age >= 0 and age <= 2:
    print("Babies")
elif age >= 3 and age <= 12:
    print("Children")
elif age >= 13 and age <= 19:
    print("Teenager")
elif age >= 20 and age <= 29:
    print("Young Adults")
elif age >= 30 and age <= 45:
    print("Middle-aged Adults")
elif age >= 46 and age <= 59:
    print("Old Adult")
else:
    print("Elderly")
```

```
Young Adults
```

Looping

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

In [141…

```
# Write your code here
for i in range(1, 13):
    for j in range(1, 13):
        print(f"{i} x {j} = {i * j}")
    print()
```

```
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
1 x 11 = 11
1 x 12 = 12

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
2 x 11 = 22
2 x 12 = 24

3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
3 x 11 = 33
3 x 12 = 36

4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
4 x 11 = 44
4 x 12 = 48

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
```

```
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
5 x 12 = 60

6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
6 x 11 = 66
6 x 12 = 72

7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
7 x 11 = 77
7 x 12 = 84

8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80
8 x 11 = 88
8 x 12 = 96

9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
```

```
9 x 9 = 81
9 x 10 = 90
9 x 11 = 99
9 x 12 = 108

10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
10 x 11 = 110
10 x 12 = 120

11 x 1 = 11
11 x 2 = 22
11 x 3 = 33
11 x 4 = 44
11 x 5 = 55
11 x 6 = 66
11 x 7 = 77
11 x 8 = 88
11 x 9 = 99
11 x 10 = 110
11 x 11 = 121
11 x 12 = 132

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 [142…

```
# Write your code here
level = int(input("Enter your level: "))
for i in range(1, level + 1):
    for j in range(1, i + 1):
        print("*", end="")
    print()
```

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

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

In [143…

```
languages = ['C/C++', 'Python', 'R', 'Java', 'SQLs', 'Assembly', 'Go', 'Rust
```

In [144…

```
# Write your code here
for i in range(len(languages)):
    if languages[i] == 'Assembly':
        print("Not you, " + languages[i])
    else:
        print("I love " + languages[i])
```

```
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 [145…

```
# Write your code here
for i in range(1,26):
    if i % 3 != 0:
        print(i)
```

```
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 [146…
```python
lower_bound = 1
upper_bound = 100
divisor = 7

result = []
```

In [147…
```python
# Write your code here
for i in range(lower_bound, upper_bound + 1):
    if i % divisor == 0:
        print(i)
```

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

6. Write a code that construct the following pattern.

input: 5 output: *##### **#### ***### ****## *****# input: 10 output: *########## **######### ***######## ****####### *****###### ******##### *******#### ********### *********## **********#

In [148…
```python
# Write your code here
level = int(input("Enter your level: "))
for i in range(1, level + 1):
    for j in range(1, i + 1):
        print("*", end="")
    for k in range(1, (level - i + 2)):
        print("#", end="")
    print()
```

```
*##########
**#########
***########
****#######
*****######
******#####
*******####
********###
*********##
**********#
```

5. Functions

1. Define a function `average` that takes arbitrary number of arguments and calculate the mean of input.

In [149…

```
# Write your code here
def average(*args):
    return sum(args) / len(args)

# test the function
print(average(1, 2, 3, 4, 5))
```

```
3.0
```

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 [150…

```
# Write your code here
def sumproduct(list1, list2):
    return sum(x * y for x, y in zip(list1, list2))

# test the function
print(sumproduct([1, 2, 3], [4, 5, 6]))
```

```
32
```

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 [151…

```
# Write your code here
def fibonacci(n):
    fib_sequence = []
    for i in range(n + 1):
        if i == 0:
            fib_sequence.append(0)
        elif i == 1:
            fib_sequence.append(1)
        else:
            fib_sequence.append(fib_sequence[-1] + fib_sequence[-2])
    return fib_sequence

# test the function
print(fibonacci(10))
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

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 [152…

```
str1 = "radar" # palindrome
str2 = "rotator" # palindrome
str3 = "lemon" # not palindrome
```

In [153…

```
# Write your code here
def is_palindrome(s):
    return s == s[::-1]

# test the function
print(is_palindrome(str1))
print(is_palindrome(str2))
print(is_palindrome(str3))
```

```
True
True
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 Marrvolo 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 [154…

```
# Write your code here
str1 = "Meaning of Life"
str2 = "Engine of a Film"

def is_anagram(s1, s2):
    return sorted(s1.lower().replace(" ", "")) == sorted(s2.lower().replace(
# test the function
print(is_anagram(str1, str2))
```

```
True
```