hw1-basicpythonprogramming-2

January 16, 2024

Ratchanon Tarawan 65070503464

1 Lab 1: Basic Python Programming

1.1 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).

[ ]:
```
# Write your code here
name = "John Doe"
age=29
position='system engineer'
salary=41500
```

What is the type of each variables?

[ ]:
```
print(type(name))
print(type(age))
print(type(position))
print(type(salary))
```

```
<class 'str'>
<class 'int'>
<class 'str'>
<class 'int'>
```

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

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

Prints his information again with his new salary.

[ ]:
```
# Write your code here
print(salary)
```

```
44405.0
```

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

[ ]:
```python
# Write your code here
```

1.2 2. Variable and Expression

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

[ ]:
```python
C = 34.5
```

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

[ ]:
```python
F = C*9/5 + 32
print(F)
```

```
94.1
```

Kelvin
$K = C + 273.15$

[ ]:
```python
K = C +273.15
print(K)
```

```
307.65
```

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

[ ]:
```python
Ro = C*21/40 + 7.5
print(Ro)
```

```
25.6125
```

1.3 3. Multi-item variables

List

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

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

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

```
Enter your name: Billy191
```

Insert `new_name` into `names` list.

[ ]:
```python
names.insert(len(names), new_name)
names
```

[ ]:
```
['Thomas', 'Kate', 'Mike', 'Amelia', 'James', 'Megan', 'Billy191']
```

Select your name from the list

```
[ ]: print(names[6])
```

```
Billy191
```

Merge `another_names` into `names`.

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

```
[ ]: names = names + another_names
     names
```

```
[ ]: ['Thomas',
      'Kate',
      'Mike',
      'Amelia',
      'James',
      'Megan',
      'Billy191',
      'Peter',
      'Steve',
      'Sam',
      'Charlotte']
```

Change `Amelia`'s name to `Amy`

```
[ ]: names[3] = 'Amy'
     names
```

```
[ ]: ['Thomas',
      'Kate',
      'Mike',
      'Amy',
      'James',
      'Megan',
      'Billy191',
      'Peter',
      'Steve',
      'Sam',
      'Charlotte']
```

Dictionary

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

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

```
[ ]: 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.

```
[ ]: capital_city.update({'Australia' : 'Canberra'})
     print(capital_city)
```

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

1.4 4. Control Flows and conditional statements

1.4.1 if...elif...else

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

```
[ ]: age = int(input('Type your age'))
```

```
Type your age18
```

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

```
[ ]: 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')
     elif age >= 60:
         print('Elderly')
```

```
Teenager
```

1.4.2 Looping

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

[ ]:
```python
# Write your code here
userInput = int(input('Input Number'))
for i in range (1,13):
  print(f"{userInput} * {i} = {userInput*i}")
```

```
Input Number20
20 * 1 = 20
20 * 2 = 40
20 * 3 = 60
20 * 4 = 80
20 * 5 = 100
20 * 6 = 120
20 * 7 = 140
20 * 8 = 160
20 * 9 = 180
20 * 10 = 200
20 * 11 = 220
20 * 12 = 240
```

2. Write a code that construct the following pattern. input: 5 output: * ** *** **** *****

[32]:
```python
num1 = int(input('Number: '))
for i in range(1, num1 + 1):
    for j in range(i):
        print('*', end='')
    print()
```

```
Number: 10
*
**
***
****
*****
******
*******
********
*********
**********
```

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

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

```
[ ]: for i in languages:
       if i == 'Assembly':
         print('Not you, Assembly.')
       else:
         print(f"I love {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.

```
[ ]: 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.

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

```
result = []
```

[ ]:
```
for i in range (lower_bound, upper_bound):
  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:
*##########
**#########
***########
****#######
*****######
******#####
*******####
********###
*********##
**********#
```

```
  File "<ipython-input-67-6183328cb379>", line 2
    output:
          ^
SyntaxError: invalid syntax
```

```
[ ]: num1 = int(input('Number: '))
     for i in range(1, num1 + 1):
         for j in range(i):
             print('*', end=' ')
         for k in range(num1 - i):
             print('#', end = ' ')
         print()
```

```
Number: 3
* # #
* * #
* * *
```

1.5 5. Functions

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

```
[ ]: def average(number1):
       sum =0
       for i in number1:
         sum += i
       return sum/len(number1)


     myNum = [1,2,3]
     print(average(myNum))
```

```
2.0
```

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

```
[ ]: def sumproduct(list1, list2):
       temp = 0
       for i in range (0, len(list1)):
         temp += list1[i] * list2[i]
       return temp

     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, ...`

[ ]:
```
def fibonacci(n):
    if n < 0:
        return "Invalid input."
    elif n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

x = int(input('Input number'))
print(fibonacci(x))
```

```
Input number9
34
```

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.

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

[ ]:
```
def is_palindrome(string):
    string = string.lower()
    string_reverse = string[::-1]
    flag = True
    if string != string_reverse:
        flag = False
    else:
      flag = True
    return flag

input_str = input('Word: ')
print(is_palindrome(input_str))
```

```
Word: nut
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`

[31]:
```
def is_anagram(str1, str2):
    str1 = sorted(str1.lower().replace(' ', ''))
    str2 = sorted(str2.lower().replace(' ', ''))

    if(str1 == str2):
      return True
    return False

result = is_anagram('Tom Marrvolo Riddle', 'I am Lord Voldermort')
result2 = is_anagram('Meaning of Life', 'Engine of a Film')
print(result)
print(result2)
```

```
True
True
```
