py101-py109_sp_easy_2

December 4, 2024

1 PY 101 - PY 109

1.1 Small Problems

1.1.1 Easy 2

1.2 Welcome Stranger

Create a function that takes 2 arguments, a list and a dictionary. The list will contain 2 or more elements that, when joined with spaces, will produce a person's name. The dictionary will contain two keys, `"title"` and `"occupation"`, and the appropriate values. Your function should return a greeting that uses the person's full name, and mentions the person's title.

```
greeting = greetings(
    ["John", "Q", "Doe"],
    {"title": "Master", "occupation": "Plumber"},
)
print(greeting)
# Hello, John Q Doe! Nice to have a Master Plumber around.
```

P: Create a function that takes a related list and dictionary. The list has 'at least two' elements that make a full name when joined with spaces. The dictionary has that person's title and occupation. {title: x, occupation: y}

E: The example shows the function taking both arguments and returning them, usable in an f string

D: I'll be creating a string

A: START SET list, dictionary FOR items in list if list.index('item') = -1 name_maker.append(item) else name_maker.append(item + " ") SET return_string f"Hello {name_maker}! Nice to have a {dictionary.get("title")} {dictionary.get("occupation")} around.

[3]:
```
def greetings(lst, dct):
    name_string = " ".join(lst)
    return (f'Hello {name_string}! Nice to have a {dct.get("title")} {dct.
 ↪get("occupation")} around.')

greeting = greetings(
    ["John", "Q", "Doe"],
    {"title": "Master", "occupation": "Plumber"},
)
```

```
print(greeting)
```

```
Hello John Q Doe! Nice to have a Master Plumber around.
```

The suggested solution put everything on the return line, which is simple.

status is interesting. I don't see it in the documentation under dictionaries.

1.3 Greeting a user

Write a program that asks for a user's name, then greets the user. If the user appends a ! to their name, the computer will yell the greeting (print it using all uppercase).

P. The problem is well-defined- take a name, print a greeting. If the name has !, print the greetiung in all caps

E: The examples show a prompt asking for a name, the name, and a greeting.

D: This is just strings

A: Prompt for name

Does name have '!"?

If no: nice quiet hello

If yes: LOUD HELLO

[3]:
```
name = input('What is your name? ')
if name[-1] != '!':
    print(f'Hello {name}.')
else:
    print('HELLO ' + name.upper())
```

```
What is your name?  matt!
```

```
HELLO MATT!
```

1.4 Multiplying Two Numbers

Create a function that takes to arguments, multiplies them together, and returns the result.

P. Simple enough to grasp - two numbers, number * number, return

E. Exaple shows exactly what to expect

D. JUst using integers

A: RUn function with two parameters, assign product of parameters to variable, return variable

[5]:
```
def multiply(x,y):
    return x * y

multiply(2,4)
```

[5]:
```
8
```

1.5 Squaring an argument

Using the multiply function from the "Multiplying Two Numbers" exercise, write a function that computes the square of its argument (the square is the result of multiplying a number by itself).

[6]:
```
def multiply(x,y):
    return x * y

def square(a):
    return multiply(a,a)

print(square(5))
```

25

[31]:
```
def multiply(x,y):
    return x * y

def power(a, n):
    '''n=exponent'''
    return_value = 1
    for i in range(n):
        return_value = multiply(return_value, a)
    return return_value
print(power(2,3))
```

8

1.6 Floating Point Arithmetic

Write a program that prompts the user for two positive numbers (floating-point), then prints the results of the following operations on those two numbers: addition, subtraction, product, quotient, floor quotient, remainder, and power. Do not worry about validating the input.

[35]:
```
float_1 = float(input('Write a floating point number: '))
float_2 = float(input('Write a floating point number: '))

return_dict = {}
return_dict["add"] = float_1 + float_2
return_dict["sub"] = float_1 - float_2
return_dict["multiply"] = float_1 * float_2
return_dict["divide"] = float_1 / float_2
return_dict["floor_div"] = float_1 // float_2
return_dict["mod"] = float_1 % float_2
return_dict["exp"] = float_1 * float_2

for key in return_dict:
    print(return_dict[key])
```

```
Write a floating point number:  7
Write a floating point number:  3.5

10.5
3.5
24.5
2.0
2.0
0.0
24.5
```

1.7 The end is near but not here

Write a function that returns the next to last word in the string argument.

Words are any sequence of non-blank characters.

You may assume that the input string will always contain at least two words.

[40]:
```
def penultimate(string):
    lst= string.rsplit(" ", 1)
    return lst[-1]

print(penultimate("Hey hey doogie howser"))
```

```
howser
```

1.8 Exclusive Or

The or operator returns a truthy value if either or both of its operands are truthy, a falsy value if both operands are falsy. The and operator returns a truthy value if both of its operands are truthy, and a falsy value if either operand is falsy. This works great until you need only one of two conditions to be truthy, the so-called exclusive or, also known as xor (pronounced "ECKS-or").

In this exercise, you will write an xor function that takes two arguments, and returns True if exactly one of its arguments is truthy, False otherwise.

take two operands, compare them. match/case statement holding state? No, even simpler. just a and not b

[1]:
```
def xor(a, b):
    if (a and not b) or (b and not a):
        return True
    return False

print(xor(5, 0) == True)
print(xor(False, True) == True)
print(xor(1, 1) == False)
print(xor(True, True) == False)
```

```
True
True
```

```
True
True
```

1.9 Odd Lists

Write a function that returns a list that contains every other element of a list that is passed in as an argument. The values in the returned list should be those values that are in the 1st, 3rd, 5th, and so on elements of the argument list.

Just a slice

[9]:
```
def oddities(lst):
    return lst[0::2]

print(oddities([2, 3, 4, 5, 6]) == [2, 4, 6])  # True
print(oddities([1, 2, 3, 4]) == [1, 3])        # True
print(oddities(["abc", "def"]) == ['abc'])     # True
print(oddities([123]) == [123])                # True
print(oddities([]) == [])                      # True

print(oddities([2, 3, 4, 5, 6]))
```

```
True
True
True
True
True
[2, 4, 6]
```

1.10 How Old is Teddy

Build a program that randomly generates and prints Teddy's age. To get the age, you should generate a random number between 20 and 100, inclusive.

[12]:
```
import random

print(f'Teddy\'s age is {random.randint(20,100)}')
```

```
Teddy's age is 49
```

[15]:
```
import random

name = input('What\'s the name of the person whose age you need?')
if not name:
    name = 'Teddy'

print(f'{name} is {random.randint(20,100)} years old!')
```

```
What's the name of the person whose age you need?

Teddy is 33 years old!
```

1.11 When Will I Retire?

Build a program that displays when the user will retire and how many years she has to work till retirement.

[22]:
```
while True:
    current_age = int(input('What is your age?'))
    retirement_age = int(input('At what age would you like to retire?'))
    if type(current_age) == int or type(retirement_age) == int:
        break
print(f'It\'s 2024. You will retire in {2024+(retirement_age - current_age)} \n'
      f'You have only {(retirement_age - current_age)} years of work to go!')
```

```
What is your age? 21
At what age would you like to retire? 72

It's 2024. You will retire in 2075
You have only 51 years of work to go!
```

1.12 Get Middle Character

Write a function that takes a non-empty string argument and returns the middle character(s) of the string. If the string has an odd length, you should return exactly one character. If the string has an even length, you should return exactly two characters.

[26]:
```
def center_of(string):
    mid = len(string) // 2
    return string[mid - 1:mid + 1] if len(string) % 2 == 0 else string[mid]




print(center_of('I Love Python!!!') == "Py")     # True
print(center_of('Launch School') == " ")         # True
print(center_of('Launchschool') == "hs")         # True
print(center_of('Launch') == "un")               # True
print(center_of('Launch School is #1') == "h")   # True
print(center_of('x') == "x")                     # True
```

```
True
True
True
True
True
True
```

1.13 Always Return Negative

Write a function that takes a number as an argument. If the argument is a positive number, return the negative of that number. If the argument is a negative number, return it as-is.

[27]:
```
def negative(x):
    return (x * -1) if x > 0 else x

print(negative(5) == -5)      # True
print(negative(-3) == -3)     # True
print(negative(0) == 0)       # True
```

```
True
True
True
```

[ ]:
```

```