Assignment2

Part 1

Development:

1) Backward-engineer (using Python) the following screenshots:
2) The program should be organized with two modules (See Ch. 4):
   a) **functions.py** module contains the following functions:
      i) get_requirements()
      ii) calculate_payroll()
      iii) print_pay()
   b) **main.py** module imports the **functions.py** module, and calls the functions.
   c) **Note: *Always* run your program from main.py.**
3) Be sure to test your program using both **IDLE** and **Visual Studio Code**.
4) ***Be Sure*** to carefully review (**How to Write Python**):
   https://realpython.com/lessons/what-pep-8-and-why-you-need-it/

Part 2

README.md file should include the following items:

1. **Assignment requirements, as per A1.**
2. Screenshots as per examples below.
3. Upload A2 **.ipynb** file and create link in README.md;
   Note: *Before* uploading **.ipynb** file, ***be sure*** to do the following actions from **Kernal** menu:
   a. **Restart & Clear Output**
   b. **Restart & Run All**

Deliverables:

1. Provide **Bitbucket** read-only access to **lis4369** repo link.
2. **Also, *be sure* the assignment README.md** includes screenshots.

Payroll No Overtime

```
Payroll Calculator

Program Requirements:
1. Must use float data type for user input.
2. Overtime rate: 1.5 times hourly rate (hours over 40).
3. Holiday rate: 2.0 times hourly rate (all holiday hours).
4. Must format currency with dollar sign, and round to two decimal places.
5. Create at least three functions that are called by the program:
        a. main(): calls at least two other functions.
        b. get_requirements(): displays the program requirements.
        c. calculate_payroll(): calculates an individual one-week paycheck.

Input:
Enter hours worked: 40
Enter holiday hours: 10
Enter hourly pay rate: 10

Output:
Base:      $400.00
Overtime:  $0.00
Holiday:   $200.00
Gross:     $600.00
```

Payroll with Overtime

```
Payroll Calculator

Program Requirements:
1. Must use float data type for user input.
2. Overtime rate: 1.5 times hourly rate (hours over 40).
3. Holiday rate: 2.0 times hourly rate (all holiday hours).
4. Must format currency with dollar sign, and round to two decimal places.
5. Create at least three functions that are called by the program:
        a. main(): calls at least two other functions.
        b. get_requirements(): displays the program requirements.
        c. calculate_payroll(): calculates an individual one-week paycheck.

Input:
Enter hours worked: 50
Enter holiday hours: 10
Enter hourly pay rate: 10

Output:
Base:      $400.00
Overtime:  $150.00
Holiday:   $200.00
Gross:     $750.00
```

a2_payroll.ipynb

(***Be sure*** to include *your* name as "Developer"!)

In [1]:
```
def get_requirements():
    print("Payroll Calculator")
    print("\nProgram Requirements:\n"
          + "1. Must use float data type for user input.\n"
          + "2. Overtime rate: 1.5 times hourly rate (hours over 40).\n"
          + "3. Holiday rate: 2.0 times hourly rate (all holiday hours).\n"
          + "4. Must format currency with dollar sign, and round to two decimal places.\n"
          + "5. Create at least three functions that are called by the program:\n"
          + "\ta. main(): calls at least two other functions.\n"
          + "\tb. get_requirements(): displays the program requirements.\n"
          + "\tc. calculate_payroll(): calculates an individual one-week paycheck.\n")
```

In [2]:
```
def calculate_payroll():
    # constants to represent base hours,  overtime and holiday rates
    # Note: python doesn't provide true constants
    BASE_HOURS = 40     # base hours
    OT_RATE = 1.5        # overtime rate
    HOLIDAY_RATE = 2.0   # holiday rate

    # IPO: Input > Process > Output
    # get user data
    print("Input:")
    # get hours worked and hourly pay rate
    hours = float(input('Enter hours worked: '))
    holiday_hours = float(input('Enter holiday hours: '))
    pay_rate = float(input('Enter hourly pay rate: '))

    # Process:
    # calculations
    base_pay = BASE_HOURS * pay_rate
    overtime_hours = hours - BASE_HOURS

    # calculate and display gross pay
    if hours > BASE_HOURS:
        # calculate gross pay with overtime

        # calculate overtime pay
        overtime_pay = overtime_hours * pay_rate * OT_RATE

        # calculate holiday pay
        holiday_pay = holiday_hours * pay_rate * HOLIDAY_RATE

        # calculate gross pay
        gross_pay = BASE_HOURS * pay_rate + overtime_pay + holiday_pay
        print_pay(base_pay, overtime_pay, holiday_pay, gross_pay)
    else:
        # calculate gross pay without overtime, but include holiday pay
        overtime_pay = 0
        holiday_pay = holiday_hours * pay_rate * HOLIDAY_RATE
        gross_pay = hours * pay_rate + holiday_pay

        # display pay
        print_pay(base_pay, overtime_pay, holiday_pay, gross_pay)
        # https://docs.python.org/3.0/tutorial/inputoutput.html

        '''
        https://docs.python.org/3/library/string.html#format-specification-mini-language
        https://www.digitalocean.com/community/tutorials/how-to-use-string-formatters-in-python-3
        alignment specifiers:
        '<' Forces field to be left-aligned within available space (default for most objects).

        '>' Forces field to be right-aligned within available space (default for numbers).

        '=' Forces padding to be placed after sign (if any) but before digits.
        Used for printing fields: '+000000120'. Alignment option only valid for numeric types.

        '^' Forces field to be centered within available space.
        '''

        # two steps:
        # 1) right-align with spaces
        # 2) currency symbol, with thousand separator, and decimal places
        # print("\nBase: {:>15}".format('${:,.2f}'.format(base_pay)))

        # Passing integer after ':' causes field to be minimum number of characters wide

        # for x in range(1, 11):
        #    print('{0:2d} {1:3d} {2:4d}'.format(x, x * x, x * x * x))

        print()  # create blank line. DON'T use \n on first print will misalign first column!
        # # fill character (*)
        # print("{0:*<10} ${1:,.2f}".format('Base:', base_pay))

    # Output:
    # https://docs.python.org/3/library/string.html#format-specification-mini-language
    # https://docs.python.org/3/library/string.html#string-formatting
    # https://www.python-course.eu/python3_formatted_output.php
    # https://www.digitalocean.com/community/tutorials/how-to-use-string-formatters-in-python-3
```

In [3]:
```
def print_pay(base_pay, overtime_pay, holiday_pay, gross_pay):
    print("\nOutput:")
    print("{0:<10} ${1:,.2f}".format('Base:', base_pay))
    print("{0:<10} ${1:,.2f}".format('Overtime:', overtime_pay))
    print("{0:<10} ${1:,.2f}".format('Holiday:', holiday_pay))
    print("{0:<10} ${1:,.2f}".format('Gross:', gross_pay))
```

In [4]:
```
import functions as f

def main():
    f.get_requirements()
    f.calculate_payroll()

if __name__ == "__main__":
    main()
```

```
Payroll Calculator

Program Requirements:
1. Must use float data type for user input.
2. Overtime rate: 1.5 times hourly rate (hours over 40).
3. Holiday rate: 2.0 times hourly rate (all holiday hours).
4. Must format currency with dollar sign, and round to two decimal places.
5. Create at least three functions that are called by the program:
        a. main(): calls at least two other functions.
        b. get_requirements(): displays the program requirements.
        c. calculate_payroll(): calculates an individual one-week paycheck.

Input:
Enter hours worked: 40
Enter holiday hours: 10
Enter hourly pay rate: 10

Output:
Base:      $400.00
Overtime:  $0.00
Holiday:   $200.00
Gross:     $600.00
```

In [5]:
```
if __name__ == "__main__":
    main()
```

```
Payroll Calculator

Program Requirements:
1. Must use float data type for user input.
2. Overtime rate: 1.5 times hourly rate (hours over 40).
3. Holiday rate: 2.0 times hourly rate (all holiday hours).
4. Must format currency with dollar sign, and round to two decimal places.
5. Create at least three functions that are called by the program:
        a. main(): calls at least two other functions.
        b. get_requirements(): displays the program requirements.
        c. calculate_payroll(): calculates an individual one-week paycheck.

Input:
Enter hours worked: 50
Enter holiday hours: 10
Enter hourly pay rate: 10

Output:
Base:      $400.00
Overtime:  $150.00
Holiday:   $200.00
Gross:     $750.00
```

Part 3
Questions (Python: Ch. 4):

1. A file that contains reusable code is called a
   module
   hierarchy chart
   function
   namespace

2. A global variable
   is defined inside the main() function
   cannot be modified inside a function
   cannot be accessed from within a function
   is defined outside of all functions

3. A local variable is defined
   inside a function
   inside the main() function
   inside an if statement
   outside of all functions

4. A return statement
   must be coded within every function
   can be used to return a local variable to the calling function
   can be used to allow the function to modify the value of a global variable
   can only be used once in each function

5. Assuming the random module has been imported into its default namespace, which of the following could possibly result in a value of 0.94?
   number = random.randfloat()
   number = random.randint(0, 1)
   number = random.randint(0, 1) / 100
   number = random.random()

6. Assuming the random module has been imported into its default namespace, which of the following could be used to simulate a coin toss where 0 = heads and 1 = tails?
   number = random.coin()
   number = random.randint(0, 1)
   number = random.randint(0, 2)
   number = random.random()

7. Assuming the random module has been imported into its default namespace, which of the following could be used to generate a random even integer from 2 through 200?
   number = random(1, 100) * 2
   number = random.randint(2, 200, 2)
   number = random.randrange(2, 202, 2)
   number = random.randrange(2, 200, 2)

8. Before you can use a standard module like the random module, you need to
   import the module
   import the module into a custom namespace
   import the module into the global namespace
   import the module into its default namespace

9. Consider the following code:

```
def get_username(first, last):
    s = first + "." + last
    return s.lower()
def main():
    first_name = input("Enter your first name: ")
    last_name = input("Enter your last name: ")
    username = get_username(first_name, last_name)
    print("Your username is: " + username)

if __name__ == "__main__":
    main()
```

What function is called first when the program runs?

- get_username()
- main()
- if __name__ == "__main__"
- input("Enter your first name: ")

10. Consider the following code:

main program:

```
import arithmetic as a
def multiply(num1, num2):
    product = num1 * num2
    result = a.add(product, product)
    return result

def main():
    num1 = 4
    num2 = 3
    answer = multiply(num1, num2)
    print("The answer is", answer)
if __name__ == "__main__":
    main()
```

arithmetic module:

```
def add(x, y):
    z = x + y
    return z
```

What values are in x and y after the code runs?

- 4, 3
- 5, 6
- 12, 12
- 24, 24

11. Consider the following code:

```
main program:
import arithmetic as a
def multiply(num1, num2):
    product = num1 * num2
    result = a.add(product, product)
    return result

def main():
    num1 = 4
    num2 = 3
    answer = multiply(num1, num2)
    print("The answer is", answer)
if __name__ == "__main__":
    main()

arithmetic module:
def add(x, y):
    z = x + y
    return z
```

The add() function is called by
- the main() function
- the multiply() function
- the arithmetic module
- the result statement

12. Consider the following code:

```
main program:
import arithmetic as a
def multiply(num1, num2):
    product = num1 * num2
    result = a.add(product, product)
    return result

def main():
    num1 = 4
    num2 = 3
    answer = multiply(num1, num2)
    print("The answer is", answer)
if __name__ == "__main__":
    main()

arithmetic module:
def add(x, y):
    z = x + y
    return z
```

When this code runs, what does it print to the console?
- The answer is 28
- The answer is 12
- The answer is 7
- The answer is 24

13. Consider the following code:

```
def get_username(first, last):
    s = first + "." + last
    return s.lower()
def main():
    first_name = input("Enter your first name: ")
    last_name = input("Enter your last name: ")
    username = get_username(first_name, last_name)
    print("Your username is: " + username)

if __name__ == "__main__":
    main()
```

What arguments are defined by the get_username() function?

- first, last
- s, first, last
- first_name, last_name
- username

14. Consider the following code:

```
def get_username(first, last):
    s = first + "." + last
    return s.lower()
def main():
    first_name = input("Enter your first name: ")
    last_name = input("Enter your last name: ")
    username = get_username(first_name, last_name)
    print("Your username is: " + username)

if __name__ == "__main__":
    main()
```

If the user enters 'Lopez' for the first prompt in main() and 'Maria' for the second prompt, what will display?

- Maria.Lopez
- maria.lopez
- lopez.maria
- Lopez.Maria

15. Consider the following code:

```
def get_username(first, last):
    s = first + "." + last
    return s.lower()
def main():
    first_name = input("Enter your first name: ")
    last_name = input("Enter your last name: ")
    username = get_username(first_name, last_name)
    print("Your username is: " + username)

if __name__ == "__main__":
    main()
```

What is the scope of the variable named s?

- global
- local
- global in main() but local in get_username()
- local in main() but global in get_username()

16. Consider the following code:

```
def get_volume(width, height, length=2):
    volume = width * height * length
    return volume
def main():
    l = 3
    w = 4
    h = 5
    v = get_volume(l, w, h)
    print(v)

if __name__ == "__main__":
    main()
```

What value is passed to the height argument by the call to the get_volume() function?

- 2
- 3
- 4
- 5

17. Consider the following code:

```
def get_volume(width, height, length=2):
    volume = width * height * length
    return volume
def main():
    l = 3
    w = 4
    h = 5
    v = get_volume(l, w, h)
    print(v)

if __name__ == "__main__":
    main()
```

When this program runs, what does it print to the console?

24
40
60
v

18. Consider the following code:

```
def get_volume(width, height, length=2):
    volume = width * height * length
    return volume
def main():
    l = 3
    w = 4
    h = 5
    v = get_volume(l, w, h)
    print(v)

if __name__ == "__main__":
    main()
```

If you add the following code to the end of the main() method, what does it print to the console?

```
print(get_volume(10, 2))
```

20
40
60
Nothing, it causes an error

19. Consider the following code:

```
main program:
import arithmetic as a
def main():
    num1 = 5
    num2 = 6
    result = a.add(num1, num2)
    print("The sum is", result)

if __name__ == "__main__":
    main()

arithmetic module:
def add(x = 4, y = 2):
    z = x + y
    return z
```

What will be displayed after the code runs?

- The sum is 11
- The sum is 6
- The sum is 17
- Nothing, the code causes an error

20. Consider the following code:

```
main program:
import arithmetic as a
def main():
    num1 = 5
    num2 = 6
    result = a.add(num1, num2)
    print("The sum is", result)

if __name__ == "__main__":
    main()

arithmetic module:
def add(x = 4, y = 2):
    z = x + y
    return z
```

What values are in x and y after the code runs?

- 9, 8
- 5, 6
- 4, 2
- 20, 12

21. If you import two modules into the global namespace and each has a function named get_value(),

- an error occurs
- a name collision occurs
- an exception occurs
- the program crashes

22. The best way to call the main() function of a program is to code
    main()
    an if statement that calls the main() function only if the function exists
    an if statement that calls the main() function only if the current module is the main module
    a while statement that calls the main() function in each loop

23. The default namespace for a module is
    the global namespace
    the name of the module followed by _default
    the first letter of the module name
    the same as the name of the module

24. To assign a default value to an argument when you define a function, you
    code the default value instead of its name in the arguments list
    set the default value for the argument in the first line of code inside the function
    code the name of the argument, the assignment operator (=), and the default value
    code the name of the argument, the default operator (:), and the default value

25. To call a function, you code the function name and
    a set of parentheses
    a set of parentheses that contains zero or more arguments
    a set of parentheses that contains one or more arguments
    a set of parentheses that contains a list of the local variables

26. To call a function with named arguments, you code the
    name of each argument, an equals sign, and the value or variable that's being passed
    values that you're passing in the same sequence that the names are defined in the function
    values that you're passing at the beginning of the function call
    values that you're passing at the end of the function call, followed by the names that correspond with the values

27. To define a function, you code the def keyword and the name of the function followed by
    a set of parentheses
    a set of parentheses that contains zero or more arguments
    a set of parentheses that contains one or more arguments
    a set of parentheses that contains a list of the local variables

28. Which of the following statements is not true about the documentation of a module?
    The documentation can describe each function in the module.
    You can use regular Python comments to document the functions of the module.
    You can call the help() function from the interactive shell to view the documentation.
    You can use Python docstrings to document the functions of the module.

29. Which of the following statements imports a module into the default namespace?
    import temperature
    import temperature as temp
    from temperature import *
    import temperature as t

30. Which of the following statements imports a module into the global namespace?
    import temperature
    import temperature as temp
    from temperature import *
    import temperature as global

31. Which of the following is not true of hierarchy charts?
    Function names should start with a verb and indicate what the functions do.
    Each function should do only what is related to the function name.
    The top level box should be for the main() function.
    Related functions should be combined into a single function.

32. Which statement would you use to call the print_name() function from a module named address that has been imported with this statement?

import address as a
    address.print_name(name)
    a.print_name(name)
    print_name(name)
    global.print_name(name)