# Part 1: Functions

**Exercise 1:** What is the purpose of the “def” keyword in Python?

a) It is slang that means “the following code is really cool”
b) It indicates the start of a function
c) It indicates that the following indented section of code is to be stored for later
d) b and c are both true
e) None of the above

ANSWER: D

**Exercise 2:** What will the following Python program print out?

```
None
def fred():
   print("Zap")

def jane():
   print("ABC")

jane()
fred()
jane()
```

a) Zap ABC jane fred jane
b) Zap ABC Zap
c) ABC Zap jane
d) ABC Zap ABC
e) Zap Zap Zap

ANSWER: D

**Exercise 3:** Rewrite your pay computation with time-and-a-half for overtime and create a function called `computepay` which takes two parameters (`hours` and `rate`).

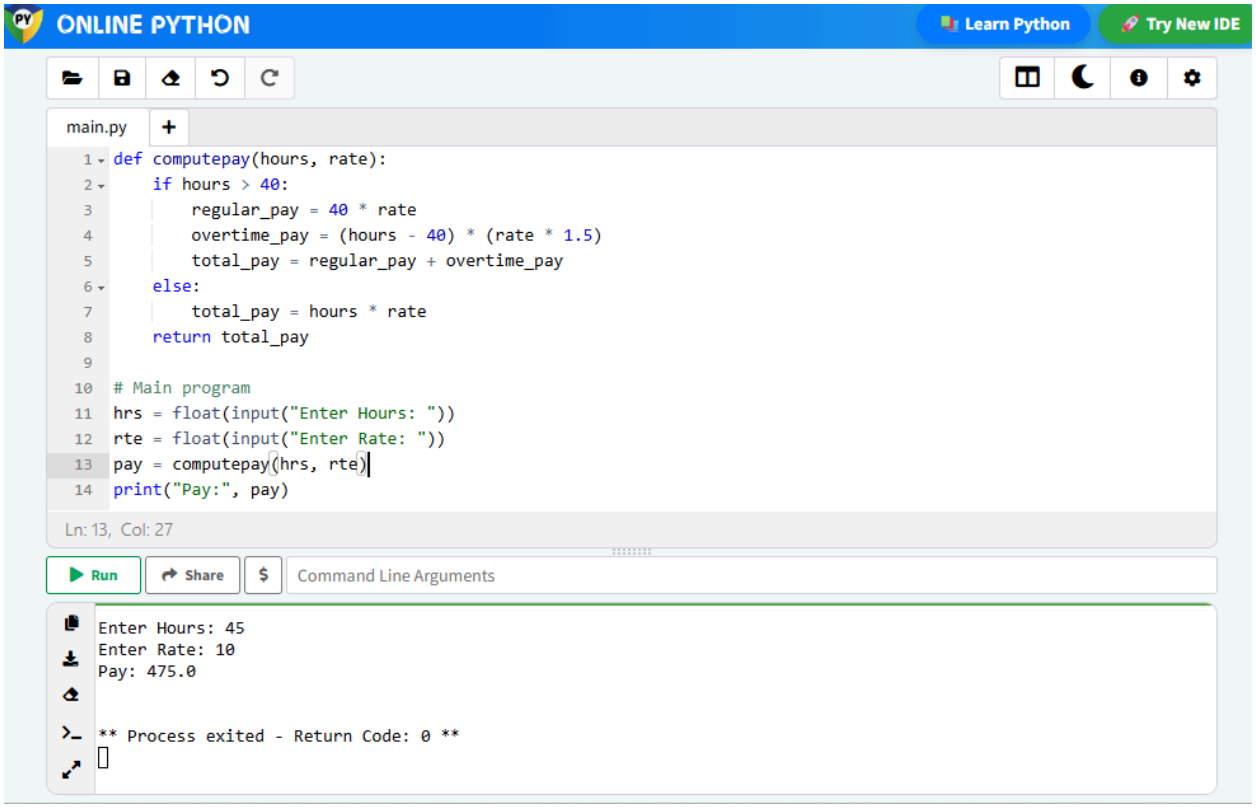
```
None
Enter Hours: 45
Enter Rate: 10
```

```
Pay: 475.0
```

ANSWER:

```
def computepay(hours, rate):
    if hours > 40:
        regular_pay = 40 * rate
        overtime_pay = (hours - 40) * (rate * 1.5)
        total_pay = regular_pay + overtime_pay
    else:
        total_pay = hours * rate
    return total_pay

# Main program
hrs = float(input("Enter Hours: "))
rte = float(input("Enter Rate: "))
pay = computepay(hrs, rte)
print("Pay:", pay)
```




**Exercise 4:** Rewrite the grade program from the previous chapter using a function called `computegrade` that takes a score as its parameter and returns a grade as a string.

None

```
 Score   Grade
>= 0.9     A
>= 0.8     B
>= 0.7     C
>= 0.6     D
 < 0.6     F
```

None

```
Enter score: 0.95
A
```

```
None
Enter score: perfect
Bad score
```

```
None
Enter score: 10.0
Bad score
```

```
None
Enter score: 0.75
C
```

```
None
Enter score: 0.5
F
```

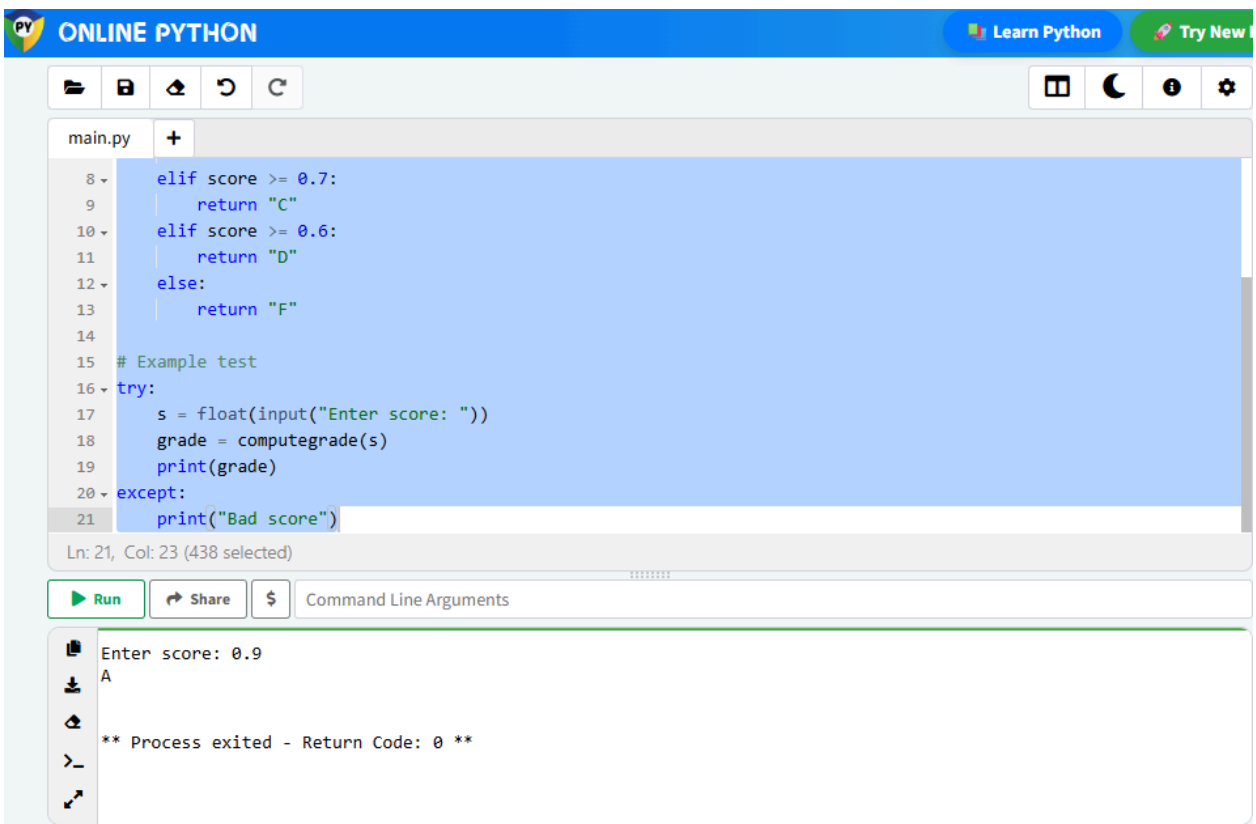
Run the program repeatedly to test the various different values for input.

ANSWER:

```
def computegrade(score):
    if score < 0.0 or score > 1.0:
        return "Bad score"
    elif score >= 0.9:
        return "A"
    elif score >= 0.8:
        return "B"
    elif score >= 0.7:
        return "C"
    elif score >= 0.6:
```

```
        return "D"
    else:
        return "F"


# Test
try:
    s = float(input("Enter score: "))
    grade = computegrade(s)
    print(grade)
except:
    print("Bad score")
```



ONLINE PYTHON

Learn Python

Try New I

main.py

```python
    elif score >= 0.7:
        return "C"
    elif score >= 0.6:
        return "D"
    else:
        return "F"

# Example test
try:
    s = float(input("Enter score: "))
    grade = computegrade(s)
    print(grade)
except:
    print("Bad score")
```

Ln: 15, Col: 15

Run | Share | $ | Command Line Arguments

```
Enter score: perfect
Bad score


** Process exited - Return Code: 0 **
```

```
Enter score: 10.0
Bad score


** Process exited - Return Code: 0 **
```

```
Enter score: 0.75
C


** Process exited - Return Code: 0 **
```

```
Enter score: 0.5
F


** Process exited - Return Code: 0 **
```

# Part 2: Loops and Iterations

**Exercise 1:** Write a program which repeatedly reads integers until the user enters “done”. Once “done” is entered, print out the total, count, and average of the integers. If the user enters anything other than a integers, detect their mistake using `try` and `except` and print an error message and skip to the next integers.

```
None
Enter a number: 4
Enter a number: 5
Enter a number: bad data
Invalid input
Enter a number: 7
Enter a number: done
16 3 5.333333333333333
```

ANSWER:

```
total = 0
count = 0


while True:
    num = input("Enter a number: ")
    if num == "done":
        break
    try:
        value = int(num)
        total += value
        count += 1
    except:
        print("Invalid input")
```

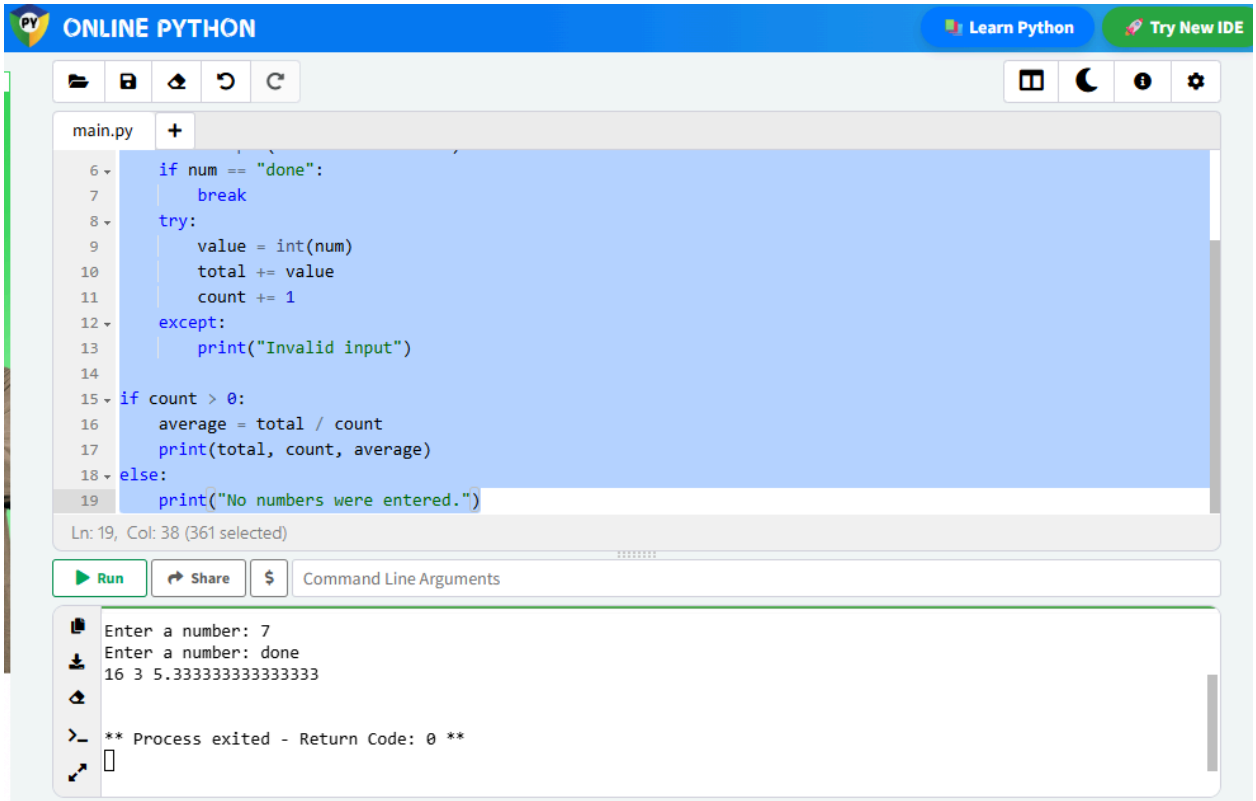
if count > 0:

    average = total / count

    print(total, count, average)

else:

    print("No numbers were entered.")




**Exercise 2:** Write another program that prompts for a list of numbers as above and at the end prints out both the maximum and minimum of the numbers instead of the average.

ANSWER:

def find_min_max():

    numbers = []

```
    while True:
        user_input = input("Enter a number: ")

        if user_input == "done":
            break

        try:
            number = float(user_input)  # Use float to allow non-integers
            numbers.append(number)
        except ValueError:
            print("Invalid input")
            continue

    if numbers:
        minimum = min(numbers)
        maximum = max(numbers)
        print("Minimum:", minimum)
        print("Maximum:", maximum)
    else:
        print("No numbers were entered.")

find_min_max()
```

ONLINE PYTHON

Learn Python | Try New IDE

main.py

```python
def find_min_max():
    numbers = []

    while True:
        user_input = input("Enter a number: ")

        if user_input == "done":
            break

        try:
            number = float(user_input)  # Use float to allow non-integers
            numbers.append(number)
        except ValueError:
            print("Invalid input")
```

Ln: 1, Col: 1

Run | Share | $ | Command Line Arguments

```
Enter a number: 4
Enter a number: 5
Enter a number: bad data
Invalid input
Enter a number: 7
Enter a number: done
Minimum: 4.0
```


Exercise 1.2 Functions | BigBlueButton - CPE 3 | BigBlueButton - R | Online Python - IDE, E | 1.3.3 Functions: CPE 3 | HIPOLITO_Exercise 1.2

online-python.com

ONLINE PYTHON

Learn Python | Try New IDE

main.py

```python
def find_min_max():
    numbers = []

    while True:
        user_input = input("Enter a number: ")

        if user_input == "done":
            break

        try:
            number = float(user_input)  # Use float to allow non-integers
            numbers.append(number)
        except ValueError:
            print("Invalid input")
```

Ln: 1, Col: 1

Run | Share | $ | Command Line Arguments

```
Enter a number: 4
Enter a number: 5
Enter a number: bad data
Invalid input
Enter a number: 7
Enter a number: done
Minimum: 4.0
```

31° Search